কম্পিউটার প্রোগ্রামিং

(সি প্রোগ্রামিং - পিডিএফ সংস্করণ ১.০)

তামিম শাহ্‌রিয়ার সুবিন

কম্পিউটার প্রোগ্রামিং
তামিম শাহ্‌রিয়ার সুবিন

গ্রন্থস্বত্ব : তামিম শাহ্‌রিয়ার সুবিন

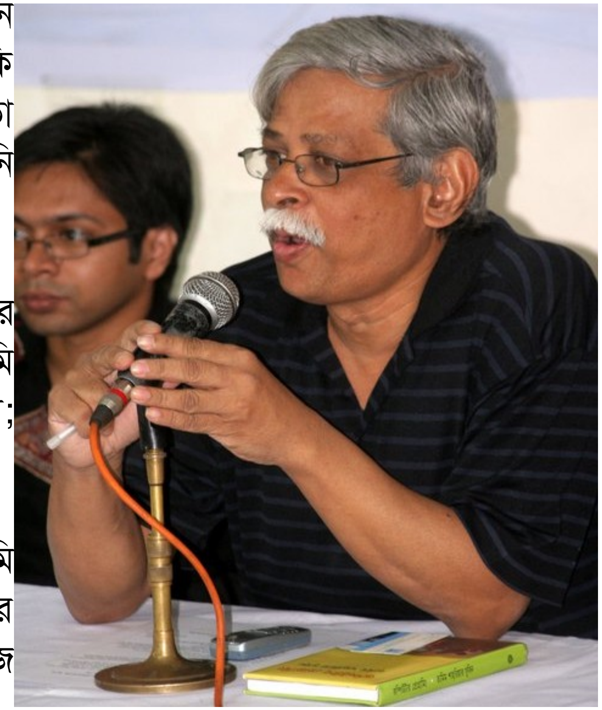
প্রথম প্রকাশ : ফেব্রুয়ারি ২০১১

সর্বশেষ হালনাগাদ : ফেব্রুয়ারি ২০১৪

উৎসর্গ

আমার মা ও বাবাকে

ভূমিকা


বইয়ের প্রকাশনা উৎসবে মুহম্মদ জাফর ইকবাল

কম্পিউটারের জন্ম হয়েছিল কম্পিউট বা হিসাব করার জন্য। এখন কম্পিউটারে মানুষ গান শোনে, সিনেমা দেখে, চিঠি লেখে, ফেসবুক করে, ইন্টারনেট ঘাঁটাঘাঁটি করে এমনকি চুরিচামারি পর্যন্ত করে কিন্তু হিসাব করে না! অথচ কম্পিউটারে কম্পিউট করার মতো আনন্দ আর কিছুতে নয়, সেটি করার জন্য যেটি জানা দরকার, সেটি হচ্ছে একটুখানি প্রোগ্রামিং।

ইউনিভার্সিটিতে বা বড় বড় শিক্ষাপ্রতিষ্ঠানে প্রোগ্রামিং শেখানো হয় কিন্তু স্কুল-কলেজের ছেলেমেয়েরাও যে খুব সহজে প্রোগ্রামিং করতে পারে, সেটি অনেকেই জানে না। আমি অনেক দিন থেকেই ভাবছিলাম, স্কুলের ছেলেমেয়েদের জন্য এরকম একটি বই লিখি; কিন্তু কিছুতেই সময় করে উঠতে পারছিলাম না।

ঠিক এরকম সময় আমার ছাত্র সুবিনের এই পাণ্ডুলিপিটি আমার চোখে পড়েছে। আমি অবাক হয়ে লক্ষ করলাম, আমি যে জিনিসটি করতে চেয়েছিলাম সুবিন ঠিক সেটিই করে রেখেছে! স্কুল-কলেজের ছেলেমেয়েদের জন্য একটি প্রোগ্রামিংয়ের বই লিখেছে, খুব সহজ ভাষায়, খুব সুন্দর করে গুছিয়ে।

আমি তার এই চমৎকার বইটির সাফল্য কামনা করি। ছেলেমেয়েরা গান শোনা, সিনেমা দেখা, চিঠি লেখা, ফেসবুক করা, ইন্টারনেট ঘাঁটাঘাঁটি করার পাশাপাশি আবার কম্পিউটারের মূল জায়গায় ফিরে আসুক – সেই প্রত্যাশায় থাকলাম।

মুহম্মদ জাফর ইকবাল

লেখক পরিচিতি

তামিম শাহরিয়ার (সুবিন)-এর জন্ম ১৯৮২ সালের ৭ই নভেম্বর ময়মনসিংহে। তাঁর বাবা মোঃ মোজাম্মেল হক ছিলেন সরকারি কর্মকর্তা এবং মা ফেরদৌসি বেগম গৃহিনী। গ্রামের বাড়ি কুমিল্লা জেলার চান্দিনা উপজেলার হারং গ্রামে।

লেখাপড়া করেছেন হোমনা সরকারি প্রাথমিক বিদ্যালয়, এ কে উচ্চ বিদ্যালয়, নটরডেম কলেজ এবং শাহজালাল বিজ্ঞান ও প্রযুক্তি বিশ্ববিদ্যালয়ে। ২০০৬ সালে শাহজালাল বিজ্ঞান ও প্রযুক্তি বিশ্ববিদ্যালয় এর কম্পিউটার সায়েন্স ও ইঞ্জিনিয়ারিং বিভাগ থেকে পাশ করেছেন। বিশ্ববিদ্যালয়ে থাকাকালীন বিভিন্ন প্রোগ্রামিং প্রতিযোগিতায় অংশগ্রহন করেছেন। পরবর্তী সময়ে (২০০৭ ও ২০০৮ সালে) তিনি এসিএম আইসিপিসি ঢাকা রিজিওনাল-এর বিচারক ছিলেন।

একটি বেসরকারি বিশ্ববিদ্যালয়ে শিক্ষকতা দিয়ে কর্মজীবন শুরু করলেও পরে একটি দেশি সফটওয়্যার নির্মাতা প্রতিষ্ঠানে কাজ করেন। তারপর যুক্তরাষ্ট্রভিত্তিক আরেকটি সফটওয়্যার তৈরির প্রতিষ্ঠানে কাজ করার পর নিজেই প্রতিষ্ঠা করেন মুক্ত সফটওয়্যার লিমিটেড নামক একটি সফটওয়্যার তৈরির প্রতিষ্ঠান। বর্তমানে সেখানেই তিনি ম্যানেজিং ডিরেক্টরের দায়িত্ব পালন করছেন। আর সম্প্রতি অনলাইন কোর্সের মাধ্যমে প্রোগ্রামিং ও সফটওয়্যার তৈরির নানা বিষয় শিক্ষাদানের জন্য তৈরি করেছেন দ্বিমিক কম্পিউটিং স্কুল। এছাড়া তিনি বাংলাদেশ গণিত অলিম্পিয়াড-এর একজন একাডেমিক কাউন্সিলর।

লেখকের কথা

গণিত অলিম্পিয়াডের ভলান্টিয়ার হবার কারণে স্কুল-কলেজের অনেক ছেলেমেয়ের সাথে আমার পরিচয় আছে এবং তারা প্রায়ই আমার কাছে জানতে চায় যে প্রোগ্রামিং শেখার জন্য কোন বই পড়বে? তাদের জন্যই বইটি লেখা। তবে ভার্সিটিতেও অনেককে আমি দেখেছি, প্রোগ্রামিংয়ের ইংরেজি বই পড়তে স্বাচ্ছন্দ্য বোধ করে না, তাই প্রথম বর্ষে প্রোগ্রামিংয়ে পিছিয়ে পড়ে এবং সেই দুর্বলতা কাটিয়ে ওঠার সুযোগ পায় না। আমি নিশ্চিত যে তারা যদি কোন বাংলা বই দিয়ে শুরু করত তবে আরো ভালো প্রোগ্রামার হতে পারত। যারা জীবনে প্রথমবারের মতো প্রোগ্রামিং শিখবে তাদের জন্য বইটি উপযোগী। ক্লাশ নাইন ও তার ওপরের ক্লাসের ছাত্রছাত্রীদের বইটি পড়তে কোন সমস্যা হওয়ার কথা নয়। বইতে 'সি' প্রোগ্রামিং ল্যাঙ্গুয়েজ ব্যবহার করা হয়েছে। তবে এটি আসলে প্রোগ্রামিং ল্যাঙ্গুয়েজের কোনো বই নয়, বরং প্রোগ্রামিংয়ের মৌলিক ধারনাগুলোর ওপর জোর দেওয়া হয়েছে যাতে ভালো প্রোগ্রামার হওয়ার জন্য শক্ত ভিত্তি তৈরি হয়।

বইটি লেখায় অনেকেই আমাকে নানাভাবে সাহায্য করেছেন তাদের গুরুত্বপূর্ণ মতামত ও উৎসাহ দিয়ে এবং বিভিন্ন ধরনের ভুল ধরিয়ে দিয়ে। যাদের নাম এ মুহূর্তে না বললেই নয়, তাঁরা হচ্ছেন, জাফর ইকবাল স্যার, মুনির হাসান, শাহ্‌রিয়ার মঞ্জুর, মনিকা আকবর, মোস্তাফিজুর রহমান, ওমর শেহাব, তৌহিদুল ইসলাম, রুহুল আমিন, মাহমুদুর রহমান, সাব্বির ইউসুফ, তানভীরুল ইসলাম, মানযুরুর রহমান খান, তাসকিনুর হাসান, রাইয়ান কামাল, আসিফ সালেকিন, আনা ফারিহা, নাফিজ ইশতিয়াক ও আবিরুল ইসলাম।

বইটি ইন্টারনেটে ফ্রি পড়া যায় http://cpbook.subeen.com ওয়েবসাইটে। এছাড়া এই পিডিএফটিও খুব স্বল্পমূল্যে ছেড়ে দেওয়া হচ্ছে যাতে অনেক বেশি সংখ্যক মানুষের কাছে বইটি পোঁছায়। এটি একটি এক্সপেরিমেন্ট। এক্সপেরিমেন্টটি সফল হলে ভবিষ্যতে আরো বই এভাবে দেওয়ার ইচ্ছা আছে।

এ বইটির ব্যাপারে কোনো পরামর্শ কিংবা মতামত জানাতে ইমেইল করা যাবে book@subeen.com ঠিকানায়।

তামিম শাহ্‌রিয়ার সুবিন

সূচিপত্র

শুরুর আগে
প্রথম প্রোগ্রাম
ডাটা টাইপ, ইনপুট ও আউটপুট
কন্ডিশনাল লজিক
লুপ (Loop)
একটুখানি গণিত
অ্যারে
ফাংশন
বাইনারি সার্চ
স্ট্রিং (String)
মৌলিক সংখ্যা
আবারও অ্যারে
বাইনারি সংখ্যা পদ্ধতি
কিছু প্রোগ্রামিং সমস্যা
শেষের শুরু
পরিশিষ্ট ১: প্রোগ্রামিং প্রতিযোগিতা
পরিশিষ্ট ২: প্রোগ্রামিং ক্যারিয়ার
পরিশিষ্ট ৩: বই ও ওয়েবসাইটের তালিকা
প্রোগ্রামিং সমস্যা

অধ্যায় শূন্যঃ শুরুর আগে

এ বইটি কম্পিউটার প্রোগ্রামিংয়ের। তবে শুরুতে আমরা কিছু বিষয় আলাপ করে নেব। তাতে আমাদের অনেক সুবিধা হবে।

কম্পিউটার তো আসলে গণনা করার যন্ত্র, তাই না? যদিও আমরা এটি দিয়ে গান শুনি, ভিডিও দেখি, গেমস খেলি, আরও নানা কাজ করি। আসলে শেষ পর্যন্ত কম্পিউটার বোঝে শূন্য (0) আর একের (1) হিসাব। তাই ব্যবহারকারী (user) যা-ই করুক না কেন, কম্পিউটার কিন্তু সব কাজ গণনার মাধ্যমেই করে। কম্পিউটারের ব্যবহার এত ব্যাপক হওয়ার পেছনে অন্যতম কারণ হচ্ছে নানা রকম সফটওয়্যার দিয়ে নানা ধরনের কাজ করা যায় কম্পিউটারে। এসব সফটওয়্যার তৈরি করতে হয় প্রোগ্রাম লিখে অর্থাৎ কী হলে কী করবে এটি প্রোগ্রামের সাহায্যে কম্পিউটারকে বোঝাতে হয়।

একসময় কিন্তু কেবল 0 আর 1 ব্যবহার করেই কম্পিউটারের প্রোগ্রাম লিখতে হতো। কারণ কম্পিউটার তো 0, 1 ছাড়া আর কিছু বোঝে না, আর কম্পিউটারকে দিয়ে কোনো কাজ করাতে চাইলে তো তার ভাষাতেই কাজের নির্দেশ দিতে হবে। 0, 1 ব্যবহার করে যে প্রোগ্রামিং করা হতো, তার জন্য যে ভাষা ব্যবহৃত হতো, তাকে বলা হয় মেশিন ল্যাঙ্গুয়েজ। তারপর এল অ্যাসেম্বলি ল্যাঙ্গুয়েজ। এতে প্রোগ্রামাররা কিছু ইনস্ট্রাকশন যেমন ADD (যোগ), MUL (গুণ) ইত্যাদি ব্যবহারের সুযোগ পেল। আর এই ভাষাকে 0, 1-এর ভাষায় নিয়ে কাজ করাবার দায়িত্ব পড়ল অ্যাসেম্বলারের ওপর, প্রোগ্রামারদের সে বিষয়ে ভাবতে হতো না। কিন্তু মানুষের চাহিদার তো শেষ নেই। নতুন নতুন চাহিদার ফলে নতুন নতুন জিনিসের উদ্ভব হয়। একসময় দেখা গেল যে অ্যাসেম্বলি ল্যাঙ্গুয়েজ দিয়েও কাজ করা ঝামেলা হয়ে যাচ্ছে। তাই বড় বড় প্রোগ্রাম লেখার জন্য আরও সহজ ও উন্নত নানা রকম প্রোগ্রামিং ভাষা তৈরি হলো। যেমন - ফরট্রান (Fortran), বেসিক (Basic), প্যাসকেল (Pascal), সি (C)। তবে এখানেই শেষ নয়, এরপর এল আরও অনেক ল্যাঙ্গুয়েজ, যার মধ্যে অন্যতম হচ্ছে, সি প্লাস প্লাস (C++), ভিজ্যুয়াল বেসিক (Visual Basic), জাভা (Java), সি শার্প (C#), পার্ল (Perl), পিএইচপি (PHP), পাইথন (Python), রুবি (Ruby)। এখনো কম্পিউটার বিজ্ঞানীরা নিত্যনতুন প্রোগ্রামিং ভাষা তৈরি করে যাচ্ছেন। প্রোগ্রামাররা এসব ভাষা ব্যবহার করে প্রোগ্রাম লেখেন আর প্রতিটি ভাষার রয়েছে আলাদা কম্পাইলার, যার কাজ হচ্ছে ওই প্রোগ্রামকে কম্পিউটারের বোধগম্য ভাষায় রূপান্তর করা, তাই এটি নিয়ে প্রোগ্রামারদের ভাবতে হয় না।

প্রোগ্রাম লেখার সময় প্রোগ্রামারকে তিনটি প্রধান কাজ করতে হয়। প্রথমে তার বুঝতে হয় যে সে আসলে কী করতে যাচ্ছে, মানে তার প্রোগ্রামটি আসলে কী কাজ করবে। তারপর চিন্তাভাবনা করে এবং যুক্তি (logic) ব্যবহার করে অ্যালগরিদম দাঁড় করাতে হয়। মানে,

লজিকগুলো ধাপে ধাপে সাজাতে হয়। এর পরের কাজটি হচ্ছে অ্যালগরিদমটাকে কোনো একটি প্রোগ্রামিং ভাষায় রূপান্তর করা, যাকে আমরা বলি কোডিং করা। একেক ধরনের কাজের জন্য একেক ল্যাঙ্গুয়েজ বেশি উপযোগী।

এই বইতে আমরা প্রোগ্রামিংয়ের মৌলিক কিছু জিনিস শেখার চেষ্টা করব এবং প্রোগ্রামগুলো আমরা লিখব সি ল্যাঙ্গুয়েজ ব্যবহার করে। আমি ধরে নিচ্ছি, তোমরা কম্পিউটার ব্যবহার করে অভ্যস্ত এবং প্রোগ্রামিং জিনিসটার সঙ্গে সম্পূর্ণ অপরিচিত। আর সি ল্যাঙ্গুয়েজ ব্যবহার করার পেছনে কারণ হচ্ছে, এটি বেশ পুরোনো হলেও অত্যন্ত শক্তিশালী ও জনপ্রিয় ল্যাঙ্গুয়েজ। প্রোগ্রামিংয়ের মৌলিক জিনিসগুলো বোঝার জন্য সি ভাষা অত্যন্ত সহায়ক। আর জনপ্রিয় সব প্রোগ্রামিং প্রতিযোগিতায় যে অল্প কয়েকটি ল্যাঙ্গুয়েজ ব্যবহার করা যায়, তার মধ্যে সি অন্যতম। আমরা অবশ্য সি ল্যাঙ্গুয়েজের পুরোটা এখানে শিখব না, কেবল মৌলিক বিষয়গুলো নিয়ে কাজ করতে যা দরকার সেটি দেখব। এই বইটি পড়ার পরে তোমরা কেবল সি-এর জন্য কোন বই পড়তে পারো অথবা অন্য কোনো ভাষা (যেমন– সি প্লাস প্লাস, জাভা কিংবা পাইথন) শেখা শুরু করে দিতে পারো। বইয়ের পরিশিষ্ট অংশে আমি কিছু বইয়ের নাম দিয়েছি, যা তোমাদের কাজে লাগবে।

বইটি পড়তে তোমাদের তিনটি জিনিস লাগবে, কম্পিউটার (ইন্টারনেট সংযোগ থাকলে ভালো হয়), সি ল্যাঙ্গুয়েজের কম্পাইলার এবং যথেষ্ট সময়। তাড়াহুড়ো না করে দুই থেকে তিন মাস সময় নিয়ে বইটি পড়লে ভালো হয়। প্রোগ্রামিং শেখার জন্য কেবল পড়াই যথেষ্ট নয়, পাশাপাশি কোডিং করতে হবে। বইয়ের প্রতিটি উদাহরণ নিজে নিজে কোড করে কম্পিউটারে চালিয়ে দেখতে হবে। যখনই আমি কোনো প্রশ্ন করব, সেটি নিয়ে চিন্তা করতে হবে। তার জন্য যদি দু-তিন ঘণ্টা বা দু-তিন দিন সময় লাগে লাগুক, কোনো ক্ষতি নেই, বরং দীর্ঘ সময় কোনো সমস্যার সমাধান নিয়ে চিন্তা করার অভ্যাসটি খুব জরুরি। কোনো অধ্যায় পুরোপুরি বোঝার আগে পরের অধ্যায় পড়া শুরু করা যাবে না। আবার কোনো অংশ যদি তোমার কাছে খুব সহজ মনে হয়, সেই অংশ ঠিকভাবে না পড়ে এবং প্রোগ্রামগুলো না করে পরের অংশে চলে যেয়ো না কিন্তু। সাধারণ পড়ালেখার সঙ্গে প্রোগ্রামিং শেখার অনেক পার্থক্য। এখানে পড়ার সঙ্গে সঙ্গে কাজ করাও জরুরি। আর এই বই পড়েই কিন্তু তুমি প্রোগ্রামার হয়ে যাবে না, বইটি পড়ে তুমি প্রোগ্রামার হওয়া শুরু করবে।

এবার আসা যাক, কম্পাইলার পাবে কোথায়? সি-এর জন্য বেশ কিছু কম্পাইলার আছে। তুমি যদি লিনাক্স কিংবা ম্যাক ব্যবহারকারী হও, তবে সবচেয়ে ভালো হচ্ছে gcc। অধিকাংশ লিনাক্সেই এটি আগে থেকে ইনস্টল করা থাকে। তোমার কম্পিউটারে না থাকলে এটি ইনস্টল করে নিতে হবে। আর উইন্ডোজ ব্যবহার করলে তুমি Codeblocks (http://www.codeblocks.org/) ব্যবহার করতে পারো। এটি একটি ফ্রি ও ওপেন সোর্স IDE (Integrated Development Environment) এবং ম্যাক আর লিনাক্সেও চলে। এমনিতে সাধারণ

কোনো টেক্সট এডিটর (যেমন: নোটপ্যাড, জিএডিট, কেরাইট) ব্যবহার করে কোড লিখে সেটি কম্পাইলার দিয়ে কম্পাইল করে রান করা যায়। তবে অধিকাংশ আইডিই (IDE) গুলোতেই নিজস্ব টেক্সট এডিটর ও কম্পাইলার থাকে। প্রোগ্রাম রান করার ব্যবস্থাও থাকে। এ ছাড়াও নানা ধরনের টুলস্ থাকে।

Codeblocks টা সরাসরি তুমি http://www.codeblocks.org সাইট থেকে ডাউনলোড ও ইনস্টল করতে পারো। Downloads পেইজে Binaries-এ গেলে উইন্ডোজের জন্য তুমি তিনটি অপশন দেখবে। তুমি দ্বিতীয়টি (codeblocks-13.12mingw-setup.exe) ডাউনলোড করবে। আর ইনস্টল করার কাজটি অন্য যেকোনো সফটওয়্যার বা গেমসের মতোই। যারা উবুন্টু ব্যবহার করো, তারা Ubuntu Software Center (Applications > Ubuntu Software Center) থেকে এটি ডাউনলোড করতে পারো।

প্রোগ্রামিং চর্চার বিষয়। ইন্টারনেটে বেশ কিছু ওয়েবসাইট আছে, যেখানে প্রচুর সমস্যা দেওয়া আছে যেগুলো প্রোগ্রামের সাহায্যে সমাধান করতে হয়। সব জায়গাতেই তুমি সি ল্যাঙ্গুয়েজে প্রোগ্রামিং করতে পারবে। এর মধ্যে কিছু কিছু ওয়েবসাইট আবার নিয়মিত প্রোগ্রামিং প্রতিযোগিতারও আয়োজন করে। এসব প্রতিযোগিতায় অংশগ্রহণ নিঃসন্দেহে তোমার প্রোগ্রামিং-দক্ষতা বৃদ্ধি করবে আর সেই সঙ্গে বিশ্বের নানা দেশের প্রোগ্রামারদের সঙ্গে মেশারও সুযোগ করে দেবে। অবশ্য প্রোগ্রামিং প্রতিযোগিতায় ভালো করতে হলে কেবল প্রোগ্রামিং জানলেই চলবে না, গাণিতিক দক্ষতাও যথেষ্ট গুরুত্বপূর্ণ। পরিশিষ্ট অংশে প্রোগ্রামিং প্রতিযোগিতা নিয়ে আলাপ করব। আর তোমাদের চর্চা করার জন্য এই পিডিএফ সংস্করণের শেষে অনেকগুলো সমস্যা দেওয়া হয়েছে, যেগুলো তুমি সমাধানের চেষ্টা করবে। আর তোমার সমাধান সঠিক হলো কী না, সেটি যাচাই করতে পারবে এই বইয়ের ওয়েবসাইট (http://cpbook.subeen.com) থেকে।

বইয়ের প্রতিটি প্রোগ্রামের নিচে আমি একটি নম্বর দিয়েছি। প্রোগ্রামের নম্বর যদি ২.৫ হয়, তার মানে হচ্ছে এটি দ্বিতীয় অধ্যায়ের পাঁচ নম্বর প্রোগ্রাম।

এটি কিন্তু কোনো গল্পের বই নয়। তাই বিছানায় শুয়ে-বসে পড়া যাবে না। বইটি পড়ার সময় কম্পিউটার চালু রাখতে হবে এবং প্রতিটি উদাহরণ সঙ্গে সঙ্গে প্রোগ্রাম লিখে দেখতে হবে, কোনো সমস্যা সমাধান করতে দিলে তখনই সেটি সমাধানের চেষ্টা করতে হবে। মনে রাখবে, যত বেশি প্রোগ্রামিং তত বেশি আনন্দ।

আশা করছি, তুমি ধৈর্য নিয়ে বাকি অধ্যায়গুলো পড়বে এবং সবগুলো প্রোগ্রাম কম্পিউটারে চালিয়ে দেখবে। তোমার জন্য শুভ কামনা।

বই পড়ার পাশাপাশি তোমরা বিষয়গুলো ভিডিও লেকচারের মাধ্যমে বোঝার চেষ্টা করতে পারো। 'প্রোগ্রামিংয়ে হাতে খড়ি' কোর্সের ডিভিডি সংগ্রহ করতে পারো http://dimikcomputing.com থেকে। আর ডিভিডি কিনতে না পারলেও কোনো সমস্যা নেই। তোমার ভালো ইন্টারনেট সংযোগ থাকলে অনলাইনে এই কোর্সটি করতে পারো, বিনামূল্যেঃ
http://programming-course.appspot.com

অধ্যায় একঃ প্রথম প্রোগ্রাম

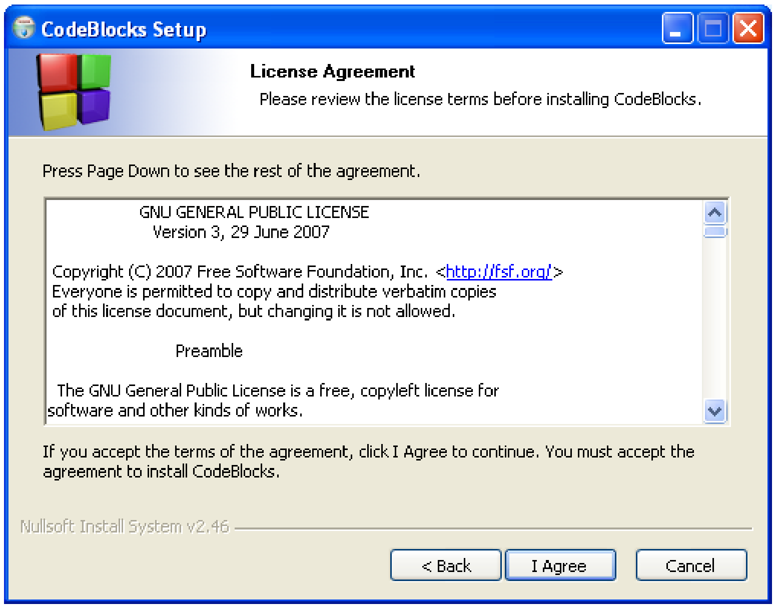
প্রোগ্রামিংয়ের জগতে স্বাগতম।

আমরা এখন একটি প্রোগ্রাম লিখে ফেলব, যেটি তোমার কম্পিউটারের স্ক্রিনে Hello World দেখাবে বা প্রিন্ট করবে। এটি হচ্ছে প্রোগ্রামিংয়ের একটি ঐতিহ্য। পৃথিবীর অধিকাংশ প্রোগ্রামারই জীবনের প্রথম প্রোগ্রাম হিসেবে এটি লেখে।

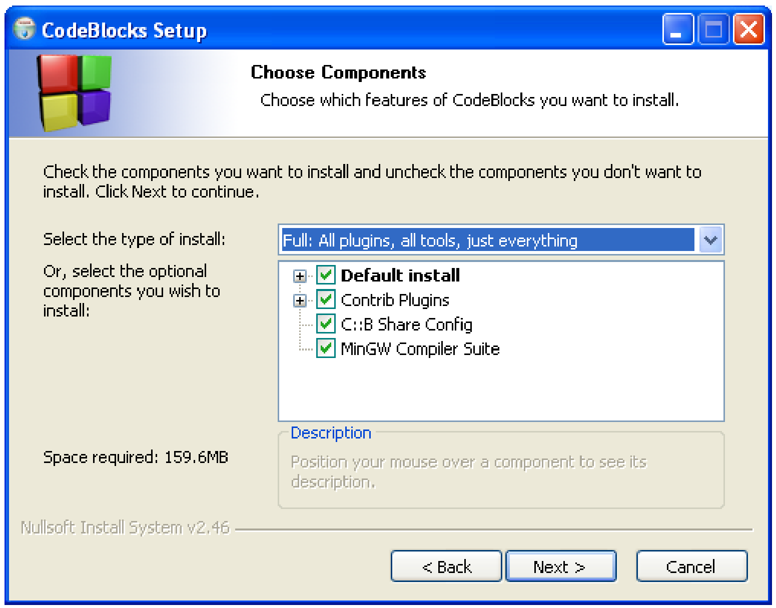
আমি এই বইয়ের প্রোগ্রামগুলো চালানোর জন্য Codeblocks ব্যবহার করব। তবে তোমরা অন্য কিছু ব্যবহার করলেও কোনো সমস্যা নেই, সবগুলোতে কাজের ধারা মোটামুটি একই রকম।

এখন চলো কোডব্লকস ইনস্টল করা যাক। বইয়ের সঙ্গে যেই সিডি আছে সেখানে তুমি ইনস্টলার পাবে। তারপর ইনস্টলারটি ডবল ক্লিক করো এবং নিচের ছবিগুলো অনুসরণ করো।

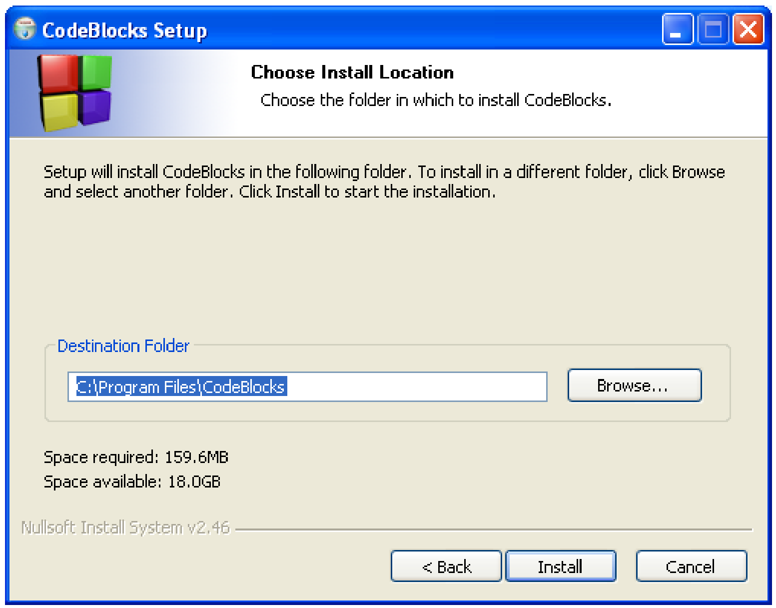
ছবি ১.১


CodeBlocks Setup
License Agreement
Please review the license terms before installing CodeBlocks.
Press Page Down to see the rest of the agreement.
GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007
Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
Preamble
The GNU General Public License is a free, copyleft license for
software and other kinds of works.
If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install CodeBlocks.
Nullsoft Install System v2.46
< Back
I Agree
Cancel

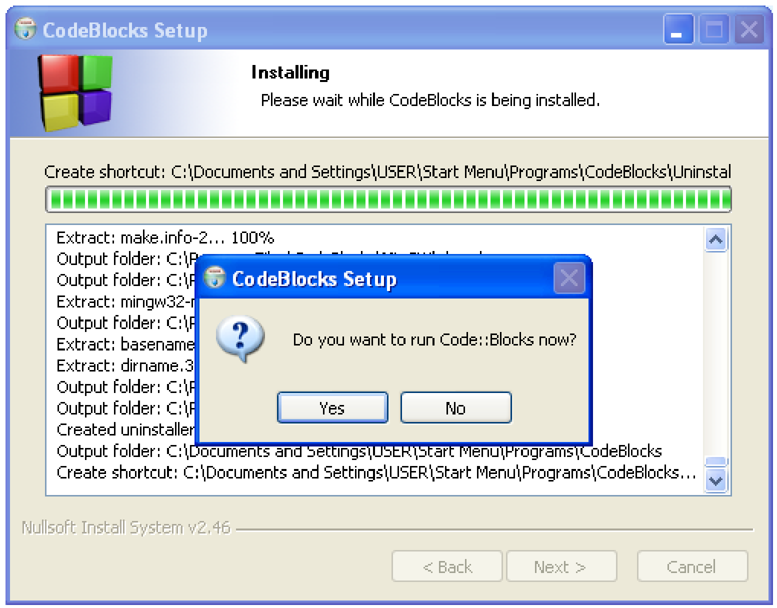
ছবি ১.২


CodeBlocks Setup
Choose Components
Choose which features of CodeBlocks you want to install.
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select the type of install:
Full: All plugins, all tools, just everything
Or, select the optional components you wish to install:
Default install
Contrib Plugins
C::B Share Config
MinGW Compiler Suite
Description
Position your mouse over a component to see its description.
Space required: 159.6MB
Nullsoft Install System v2.46
< Back
Next >
Cancel

ছবি ১.৩

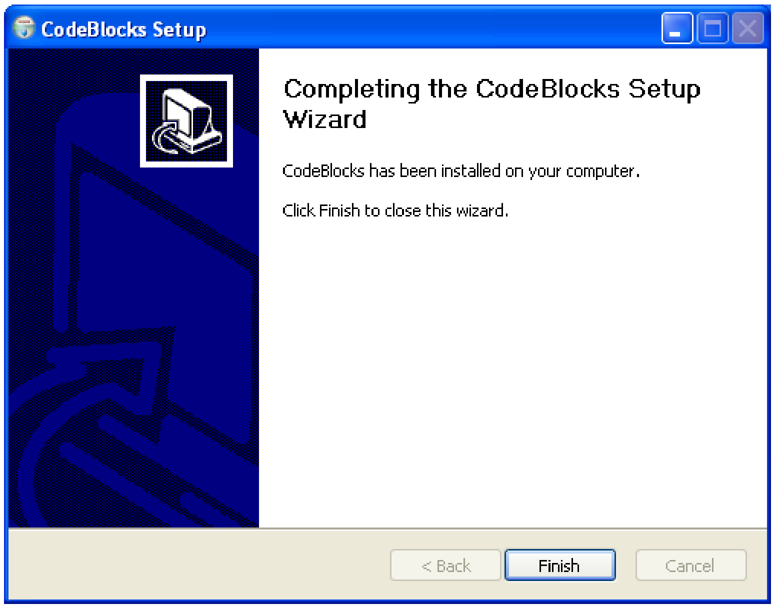

CodeBlocks Setup
Choose Install Location
Choose the folder in which to install CodeBlocks.
Setup will install CodeBlocks in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.
Destination Folder
C:\Program Files\CodeBlocks
Browse...
Space required: 159.6MB
Space available: 18.0GB
Nullsoft Install System v2.46
< Back
Install
Cancel

ছবি ১.৪



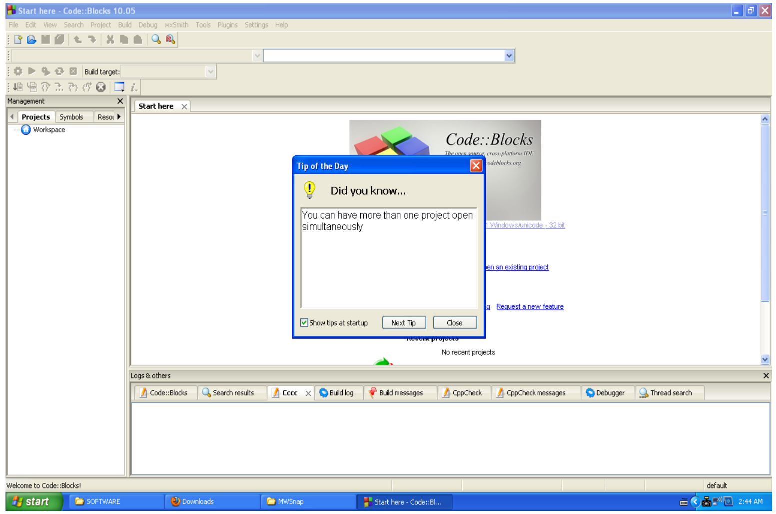
ছবি ১.৫

এবার No বাটনে ক্লিক করবে (ছবি ১.৫)।



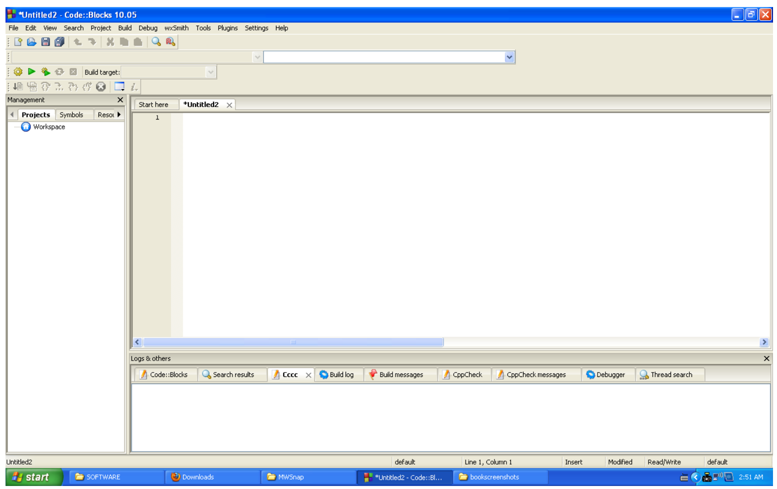
ছবি ১.৬

ইনস্টল হয়ে গেল। এখন উইন্ডোজের Start মেনুতে Programs-এ গিয়ে Codeblocks চালু করো। উবুন্টুতে এটি থাকবে Applications > Programming-এর ভেতর।



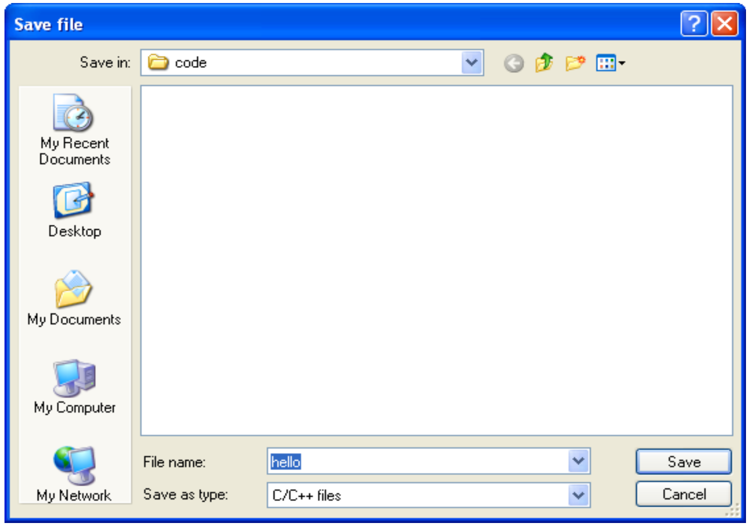
ছবি ১.৭

এখানে (ছবি ১.৭) তোমরা Show tips at startup চেকবক্সের টিক (tick) চিহ্নটি উঠিয়ে দিতে পারো। এখন File মেনু থেকে New File সাবমেনুতে গিয়ে Empty File-এ ক্লিক করো। নিচের ছবির (১.৮) মতো উইন্ডো আসবে।



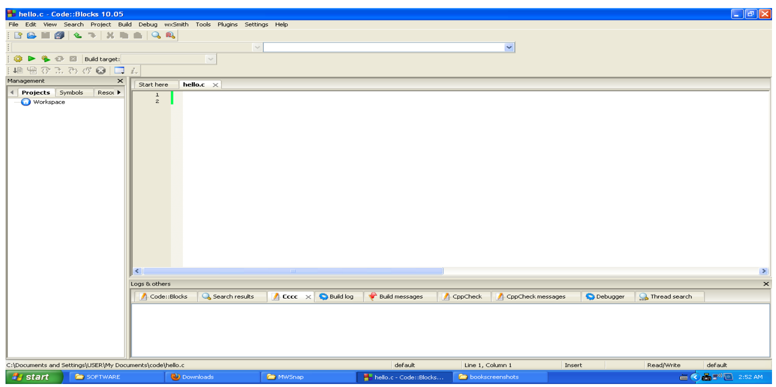
ছবি ১.৮

এখন তোমরা প্রোগ্রামগুলো রাখার জন্য হার্ডডিস্কের ভেতর একটি ফোল্ডার তৈরি করে নাও। ওই ফোল্ডারে ফাইলগুলো সেভ (Save) করবে। ফাইলের যেকোনো একটি নাম দাও। আর Save as type হবে C/C++ files (ছবি ১.৯)।



ছবি ১.৯

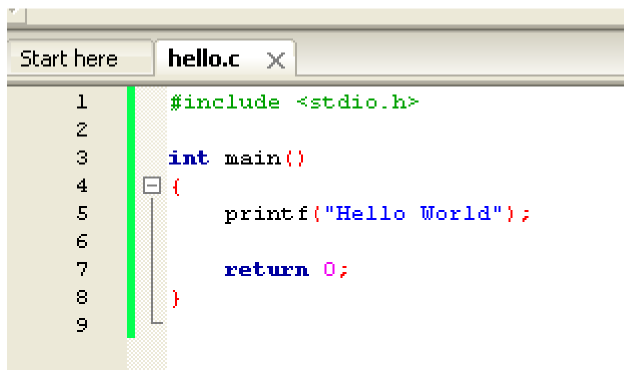
নিচের ছবিতে দেখো ফাইলের নাম হচ্ছে hello.c। সি প্রোগ্রামের সব ফাইলের এক্সটেনশন হবে .c।



ছবি ১.১০

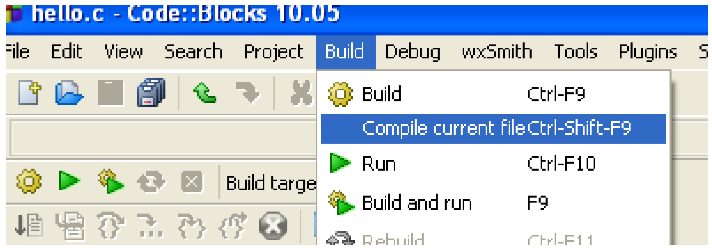
এখানে আমরা আমাদের কোড বা প্রোগ্রাম লিখব। নিচের কোডটি টাইপ করে ফেলো এবং ফাইলটি সেভ করো।

```
#include <stdio.h>

int main ()
{
    printf("Hello World");

    return 0;
}
```
প্রোগ্রাম: ১.১



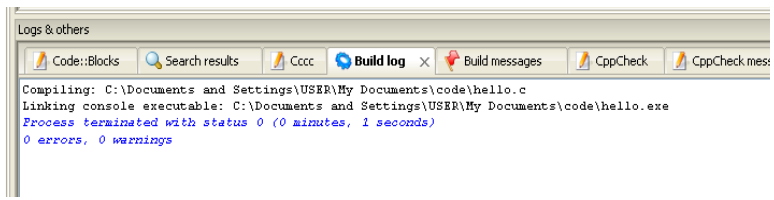
ছবি ১.১১

তোমরা হয়তো চিন্তা করছ, আমি এই হিজিবিজি কী লিখলাম? আস্তে ধীরে সব ব্যাখ্যা করব, চিন্তা নেই! আপাতত আমার কথামতো কাজ করে যাও। এবার Build মেনুতে গিয়ে Compile Current File-এ ক্লিক করো।



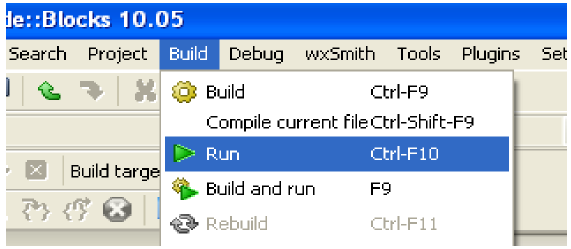
ছবি ১.১২

তুমি যদি প্রোগ্রামটি ঠিকভাবে টাইপ করে থাকো তবে কম্পাইলার তোমাকে বলবে যে 0 errors, 0 warnings, মানে - প্রোগ্রামে syntax ঠিক আছে। ১.১৩ নম্বর ছবিটি দেখো।

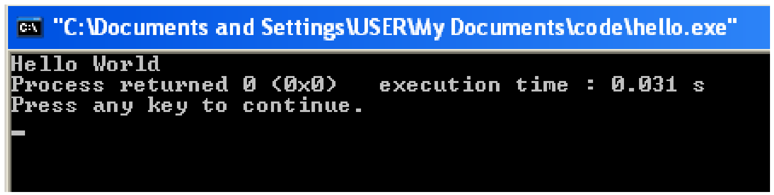


ছবি ১.১৩

এখন আবার Build মেনুতে গিয়ে Run-এ ক্লিক করো (ছবি ১.১৪)। তাহলে তোমার প্রোগ্রাম চালু হয়ে যাবে এবং তুমি ১.১৫ নম্বর ছবির মতো স্ক্রিন দেখতে পাবে।



ছবি ১.১৪



ছবি ১.১৫

এখানে দেখো, তোমার প্রোগ্রামটি স্ক্রিনে Hello World প্রিন্ট করেছে। পরের লাইনে বলা আছে Process returned 0 (0x0) (এটির অর্থ নিয়ে আমাদের এখন মাথা না ঘামালেও চলবে) আর execution time : 0.031 s মানে প্রোগ্রামটি চলতে 0.031 সেকেন্ড সময় লেগেছে। তারপরের লাইন হচ্ছে, Press any key to continue. কি-বোর্ডে Any key খুঁজে না পেলে অন্য যেকোনো কি চাপলেই চলবে।

তুমি যদি প্রোগ্রামটি ঠিকঠাকভাবে রান করাতে পারো এবং Hello World লেখাটা দেখে থাকো তাহলে তোমাকে অভিনন্দন। তুমি বেশ গুরুত্বপূর্ণ একটি কাজ করে ফেলেছ।

আর ঠিকঠাকভাবে রান করাতে না পারলে আবার শুরু থেকে চেষ্টা করো। প্রয়োজনে অভিজ্ঞ কারও সাহায্য নাও। কারণ এই প্রোগ্রাম না

চালাতে পারলে বইয়ের পরের অংশ পড়ে তেমন একটি লাভ হবে না।

এবারে দেখা যাক আমি কী লিখেছি কোডে।
প্রথম লাইন ছিল: #include <stdio.h>, এটি কেন লিখেছি একটু পরে বলছি।
দ্বিতীয় লাইন ফাঁকা। দেখতে সুন্দর লাগে তাই।
তৃতীয় লাইন: int main()। এটিকে বলে মেইন ফাংশন। সি প্রোগ্রামগুলো মেইন ফাংশন থেকে কাজ করা শুরু করে, তাই সব প্রোগ্রামে একটি (এবং কেবল একটি) মেইন ফাংশন থাকতে হয়। মেইন ফাংশনের শুরুতে দ্বিতীয় বন্ধনী দিয়ে শুরু করতে হয় আর শেষও করতে হয় একটি দ্বিতীয় বন্ধনী দিয়ে। শেষ করার আগে আমি return 0; লিখেছি, সেটি কেন এখন ব্যাখ্যা না করলেই ভালো হয়। তাই আপাতত তোমরা যেকোনো প্রোগ্রামে নিচের অংশটুকু লিখে ফেলবে:

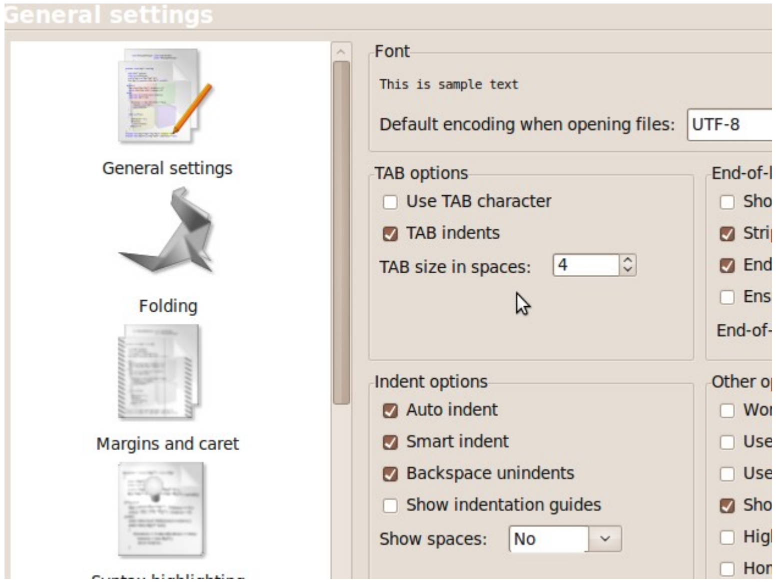
```
int main()
{
   এখানে কোড থাকবে।

   return 0;
}
```

প্রোগ্রামের পরের লাইন খেয়াল করো: printf(“Hello World”); এটি একটি স্টেটমেন্ট। এখানে printf() হচ্ছে একটি ফাংশন যার কাজ হচ্ছে স্ক্রিনে কিছু প্রিন্ট করা। ডবল কোটেশন চিহ্নের ভেতরে যা লিখবে তা-ই স্ক্রিনে সে প্রিন্ট করবে। এই ফাংশনটি স্ক্রিনে প্রিন্ট করে কীভাবে সেটি আসলে বলা আছে stdio.h নামে একটি ফাইলে। এই ফাইলগুলোকে বলে হেডার (header) ফাইল (.h হচ্ছে হেডার ফাইলের এক্সটেনশন)। stdio.h ফাইলে স্ট্যান্ডার্ড ইনপুট আর আউটপুট-সংক্রান্ত যাবতীয় ফাংশন লেখা আছে, আমরা কেবল সেগুলো ব্যবহার করব, ফাংশনগুলো কীভাবে কাজ করে সেটি এখন আমাদের জানার দরকার নেই। আর যেহেতু printf() ফাংশন ব্যবহার করেছি, তাই প্রোগ্রামের শুরুতে #include <stdio.h> লিখতে হয়েছে। এই রকম আরও অনেক প্রয়োজনীয় হেডার ফাইল আছে, যার কিছু আমরা পরবর্তী সময়ে কাজের প্রয়োজনে দেখব।

এখন একটি ব্যাপার খেয়াল করো। printf(“Hello World”);-এর শেষে একটি সেমিকোলন রয়েছে। সি ল্যাঙ্গুয়েজে প্রতিটি

স্টেটমেন্টের পরেই একটি সেমিকোলন থাকে। একটি স্টেটমেন্টের কাজ শেষ হলে পরের স্টেটমেন্টের কাজ শুরু হয়। return 0;ও একটি স্টেটমেন্ট, তাই এটিও সেমিকোলন দিয়ে শেষ করতে হয়েছে। শুরুর দিকে অনেকেই সেমিকোলন দিতে ভুলে যায়, তখন কম্পাইল এরর (compile error) হয়। তোমরা একটি সেমিকোলন মুছে দিয়ে কম্পাইল করার চেষ্টা করে দেখতে পারো।

এবারে একটি খুব গুরুত্বপূর্ণ কথা বলে রাখি। তোমরা কোডটি খেয়াল করলে দেখবে যে আমি #include <stdio.h>, int main(), { ও } যেই লাইনে আছে সেটি এডিটরের একেবারে বাঁ দিক থেকে শুরু করেছি। আর printf এবং return 0-এর আগে চারটি স্পেস (ফাঁকা জায়গা) দিয়ে নিয়েছি। এটিকে বলে ইনডেন্টেশন (Indentation)। এরকম না করলেও প্রোগ্রামটি চলত এবং তাই অনেকেই ইন্ডেন্টেশনের ব্যাপারটি গুরুত্ব দেয় না এবং ঠিকমতো ইনডেন্টেশন করে না। যেকোনো ভালো অভ্যাসের মতো ইন্ডেন্টেশনের অভ্যাস তৈরি করাটা একটু কঠিন, তবে বিষয়টা কিন্তু দাঁত মাজার মতোই গুরুত্বপূর্ণ। ইনডেন্টেশন করার অভ্যাস ঠিকমতো তৈরি না হলে প্রোগ্রামারদের সহকর্মী বা বসের বকা শুনতে হয়, অনেক জায়গায় তো ইন্টারভিউতেই বাদ পড়ে যেতে হয়। আশা করছি তোমরা ব্যাপারটি বেশ গুরুত্ব সহকারে নেবে। আমি বইয়ের সমস্ত উদাহরণেই যথাযথভাবে ইনডেন্টেশন করার চেষ্টা করব তবে ছাপার সময় একটু এদিক-ওদিক হতে পারে, সেটি তোমরা বুঝে নেবে। ইন্ডেন্টশনের জন্য সাধারণত চারটি স্পেস দেওয়াটাই এখন স্ট্যান্ডার্ড। তোমরা এডিটরে অপশন সেট করতে পারো যাতে ট্যাব (Tab) চাপলে সেটি চারটি স্পেসের সমান হয়। Codeblocks-এ Settings মেনুতে Editor-এ ক্লিক করে TAB Options-এ TAB indents চেক করো এবং TAB size in spaces 4 দাও (১.১৬ ছবিটি দেখো)।



ছবি ১.১৬

এবারে তোমাদের জন্য একটি কাজ। একটি প্রোগ্রাম লেখো যেটি স্ক্রিনে প্রিন্ট করবে: I love my country, Bangladesh।

প্রোগ্রামটি টাইপ করার পরে অবশ্যই কম্পাইল ও রান করবে। কম্পাইল করার আগে সেভ করতে ভুলবে না।

অধ্যায় দুইঃ ডাটা টাইপ, ইনপুট ও আউটপুট

এ অধ্যায়ে আমরা কিছু ছোট ছোট প্রোগ্রাম লিখব। সবগুলো প্রোগ্রাম অবশ্যই কম্পিউটারে চালিয়ে দেখবে এবং একটু পরিবর্তন করে কম্পাইল ও রান করার চেষ্টা করবে।

আমাদের প্রথম প্রোগ্রামটি হবে দুটি সংখ্যা যোগ করার প্রোগ্রাম। এখন কথা হচ্ছে, সংখ্যাগুলো তো কম্পিউটারের মেমোরিতে রাখতে হবে, সেই জটিল কাজটি কীভাবে করব? চিন্তা নেই! সব প্রোগ্রামিং ল্যাঙ্গুয়েজে ভেরিয়েবল বলে একটি জিনিস আছে যেটি কোন নির্দিষ্ট মান ধারণ করার জন্য ব্যবহার করা হয়। ভেরিয়েবলের একটি নাম দিতে হয়, তারপর ***ভেরিয়েবল = কোনো মান*** লিখে দিলে ভেরিয়েবলের ভেতর সেটি ঢুকে যায়। এটির সঙ্গে গাণিতিক সমীকরণের কিন্তু কোনো সম্পর্ক নেই।

চলো, প্রোগ্রামটি লিখে রান করাই, তারপর ব্যাখ্যা করা যাবে।

```
#include <stdio.h>

int main()
{
    int a;
    int b;
    int sum;

    a = 50;
    b = 60;

    sum = a + b;

    printf("Sum is %d", sum);
```

```
    return 0;
}
```
প্রোগ্রাম: ২.১

প্রোগ্রামটি রান করাও, তুমি স্ক্রিনে দেখবে: Sum is 110।

এখানে a, b, sum তিনটি ভেরিয়েবল (variable) আলাদা সংখ্যা ধারণ করে। প্রথমে আমাদের বলে দিতে হবে যে a, b, sum নামে তিনটি ভেরিয়েবল আছে। এবং এগুলোতে কী ধরনের ডাটা থাকবে সেটিও বলে দিতে হবে। int a; দিয়ে আমরা কম্পাইলারকে বলছি a নামে একটি ভেরিয়েবল এই প্রোগ্রামে আছে যেটি একটি পূর্ণসংখ্যা (integer)-এর মান ধারণ করার জন্য ব্যবহার করা হবে। এই কাজটিকে বলে ভেরিয়েবল ডিক্লারেশন। আর int হচ্ছে ডাটা টাইপ, যেটি দেখে সি-এর কম্পাইলার বুঝবে যে এতে ইন্টিজার টাইপ ডাটা থাকবে। আরও বেশ কিছু ডাটা টাইপ আছে, সেগুলো আমরা আস্তে আস্তে দেখব। আমরা চাইলে একই টাইপের ভেরিয়েবলগুলো ডিক্লেয়ার করার সময় আলাদা লাইনে না লিখে একসঙ্গে কমা দিয়েও লিখতে পারতাম, যেমন: int a, b, sum;। আর লক্ষ করো যে ভেরিয়েবল ডিক্লারেশনের শেষে সেমিকোলন ব্যবহার করতে হয়।

এরপর আমি দুটি স্টেটমেন্ট লিখেছি:
a = 50;
b = 60;
এখানে a-এর মান 50 আর b-এর মান 60 বলে দিলাম (assign করলাম), যতক্ষণ না এটি আমরা পরিবর্তন করছি, কম্পাইলার a-এর মান 50 আর b-এর মান 60 ধরবে।

পরের স্টেটমেন্ট হচ্ছে: sum = a + b;। এতে বোঝায়, sum-এর মান হবে a + b-এর সমান, অর্থাৎ a ও b-এর যোগফল যে সংখ্যাটি হবে সেটি আমরা sum নামের ভেরিয়েবলে রেখে দিলাম (বা assign করলাম)।

এবারে যোগফলটি মনিটরে দেখাতে হবে, তাই আমরা printf ফাংশন ব্যবহার করব।

printf(“Sum is %d”, sum);

এখানে দেখো printf ফাংশনের ভেতরে দুটি অংশ। প্রথম অংশে "Sum is %d" দিয়ে বোঝানো হয়েছে স্ক্রিনে প্রিন্ট করতে হবে Sum is এবং তার পরে একটি ইন্টিজার ভেরিয়েবলের মান যেটি %d-এর জায়গায় বসবে। তারপর কমা দিয়ে আমরা sum লিখে বুঝিয়ে দিয়েছি যে %d-তে sum-এর মান প্রিন্ট করতে হবে। ইন্টিজারের জন্য যেমন %d ব্যবহার করলাম, অন্য ধরনের ভেরিয়েবলের জন্য অন্য কিছু লিখতে হবে, যেটি আমরা ব্যবহার করতে করতে শিখব। খুব ভালো হতো যদি আমি এখন একটি চার্ট লিখে দিতাম যে সি ল্যাঙ্গুয়েজে কী কী ডাটা টাইপ আছে, সেগুলো কী দিয়ে লেখে এবং প্রিন্ট করার জন্য কী করতে হবে আর তোমরা সেই চার্ট মুখস্থ করে ফেলতে। কিন্তু শুধু শুধু মুখস্থ করার কোনো দরকার নেই, মুখস্থ করার প্রবণতা চিন্তাশক্তি কমায় আর প্রোগ্রামারদের জন্য চিন্তা করার ক্ষমতা খুবই গুরুত্বপূর্ণ।

আমরা ওপরের প্রোগ্রামটি চাইলে এভাবেও লিখতে পারতাম:

```
#include <stdio.h>

int main()
{
    int a, b, sum;

    a = 50;
    b = 60;

    sum = a + b;

    printf("Sum is %d", sum);

    return 0;
}
```
প্রোগ্রাম: ২.২

আবার ভেরিয়েবল ডিক্লেয়ার করার সময় একই সঙ্গে অ্যাসাইনও করা যায়:

```
#include <stdio.h>

int main()
{
    int a = 50, b = 60, sum;

    sum = a + b;

    printf("Sum is %d", sum);

    return 0;
}
```

প্রোগ্রাম: ২.৩

এখন তোমাদের জন্য একটি প্রশ্ন। নিচের প্রোগ্রামটির আউটপুট কী হবে?

```
#include <stdio.h>

int main()
{
    int x, y;
```

```
    x = 1;
    y = x;
    x = 2;

    printf("%d", y);

    return 0;
}
```
প্রোগ্রাম: ২.৪

কী মনে হয়? আউটপুট 1 নাকি 2? আউটপুট হবে 1, কারণ প্রথমে কম্পাইলার দেখবে, x-এর মান 1 অ্যাসাইন করা হয়েছে (x = 1;)। তারপর x-এর মানটি আবার y-এ অ্যাসাইন করা হয়েছে (y = x;)। এখন y-এর মান 1। তারপর আবার x-এর মান 2 অ্যাসাইন করা হয়েছে। কিন্তু তাতে y-এর মানের কোনো পরিবর্তন হবে না। প্রোগ্রামিংয়ে y = x; আসলে কোনো সমীকরণ না যে এটি সবসময় সত্য হবে। '=' চিহ্ন দিয়ে একটি ভেরিয়েবলে নির্দিষ্ট কোনো মান রাখা হয়।

এবারে নিচের প্রোগ্রামটি দেখো:

```
#include <stdio.h>

int main()
{
    int a = 50, b = 60, sum;

    sum = a + b;
```

```
    printf("%d + %d = %d", a, b, sum);

    return 0;
}
```
প্রোগ্রাম: ২.৫

প্রোগ্রামটি মনিটরে কী প্রিন্ট করে? চালিয়ে দেখো। printf("%d + %d = %d", a, b, sum); না লিখে printf("%d + %d = %d", b, a, sum); লিখে প্রোগ্রামটি আবার চালাও। এখন জিনিসটি চিন্তা করে বুঝে নাও।

লেখাপড়া করার সময় আমাদের মনে নানা বিষয়ে নানা প্রশ্ন আসে, যার উত্তর আমরা বইতে খুঁজি, শিক্ষককে জিজ্ঞাসা করি, ইন্টারনেটে খুঁজি বা চিন্তা করে যুক্তি দাঁড় করিয়ে উত্তরটি বের করি। আমাদের দুর্ভাগ্য যে বেশিরভাগ ছেলেমেয়েই শেষ কাজটি করে না, কারণ নিজে নিজে চিন্তা করতে একটু সময় লাগে ও পরিশ্রম হয়, সেই সময় আর শ্রম তারা দিতে চায় না। আর আমাদের অভিভাবক, শিক্ষক ও শিক্ষাব্যবস্থা চিন্তা করার জন্য কোনো পুরস্কার দেয় না, বরং মুখস্থ করার জন্যই পুরস্কৃত করে।

যা-হোক, প্রোগ্রামিংয়ের ব্যাপারে যখনই মনে কোনো প্রশ্ন আসবে, সঙ্গে সঙ্গে একটি প্রোগ্রাম লিখে ফেলবে। দেখো তোমার কম্পাইলার কী বলে। ধরা যাক, আমরা যদি int টাইপের ভেরিয়েবলে দশমিক যুক্ত সংখ্যা (বাস্তব সংখ্যা বা real number) ব্যবহার করতাম, তাহলে কী হতো?

```
#include <stdio.h>

int main()
{
    int a = 50.45, b = 60, sum;

    sum = a + b;

    printf("%d + %d = %d", a, b, sum);
```

```
    return 0;
}
```
প্রোগ্রাম: ২.৬

এখানে a-এর মান 50.45 ব্যবহার করলাম। এবারে প্রোগ্রাম চালাও, দেখো কী হয়। আবার মনে যদি প্রশ্ন আসে যে, main ফাংশনের শেষ লাইনে return 0; না লিখলে কী হয়? তাহলে return 0; ছাড়া প্রোগ্রাম চালিয়ে দেখো কী হয়।

আউটপুট হবে: 50 + 60 = 110।

সি কম্পাইলার a-এর মান 50 ধরেছে, যদিও আমরা 50.45 অ্যাসাইন করেছি। এই ব্যাপারটিকে বলে টাইপ কাস্ট (type cast)। বাস্তব সংখ্যা রাখার জন্য সি ভাষায় double নামের ডাটা টাইপ রয়েছে। টাইপ কাস্ট করে double সংখ্যাটিকে int-এ নেওয়া হয়েছে, এটি অটোমেটিক হয়। আবার কম্পাইলারকে বলেও দেওয়া যায়: int a = (int) 50.45।
int a = 50.99; এখানে a-এর মান হবে 50। int a = -50.9; লিখলে a-এর মান হয় -50। এক কথায় বললে double থেকে int-এ টাইপ কাস্ট করলে দশমিকের পরের অংশটি বাদ পড়ে যাবে।

আরেকটি কথা। যেই ভেরিয়েবলকে টাইপ কাস্ট করা হচ্ছে, তার মান কিন্তু পরিবর্তন হয় না। টাইপ কাস্ট করা মানটি একটি ভেরিয়েবলে রাখা যায়। নিচের প্রোগ্রামটি কম্পিউটারে চালালেই বুঝতে পারবে।

```
#include <stdio.h>

int main()
{
    int n;
    double x;

    x = 10.5;
```

```
    n = (int)x;

    printf("Value of n is %d\n", n);
    printf("Value of x is %lf\n", x);

    return 0;
}
```

প্রোগ্রাম: ২.৭

প্রোগ্রামের আউটপুট দেখো। x-এর মান কিন্তু পরিবর্তন হয়নি। আর বুঝতেই পারছ যে বাস্তব সংখ্যা রাখার জন্য সি-তে যে double টাইপের ভেরিয়েবল ব্যবহার করা হয়, তা প্রিন্ট করার সময় %lf (l এখানে ইংরেজি ছোট হাতের L) ব্যবহার করতে হয়।
int ডাটা টাইপে তো কেবল পূর্ণ সংখ্যা রাখা যায়। কিন্তু সেটি কী যেকোনো পূর্ণসংখ্যা? উত্তরের জন্য একটি প্রোগ্রাম লিখি:

```
#include <stdio.h>

int main()
{
    int a;

    a = 1000;
    printf("Value of a is %d", a);

    a = -21000;
    printf("Value of a is %d", a);

    a = 10000000;
```

```
    printf("Value of a is %d", a);

    a = -10000000;
    printf("Value of a is %d", a);

    a = 100020004000503;
    printf("Value of a is %d", a);

    a = -4325987632;
    printf("Value of a is %d", a);

    return 0;
}
```
প্রোগ্রাম: ২.৮

এখানে আমরা a-তে বিভিন্ন সংখ্যা অ্যাসাইন করলাম। সব মান কি ঠিকঠাক আসছে? আসেনি। কেন আসেনি সেটি ব্যাখ্যা করার আগে একটি কথা বলে নিই। পরপর এতগুলো printf-এর কারণে তোমার কম্পিউটারের স্ক্রিনে নিশ্চয়ই দেখতে একটু অস্বস্তিকর লাগছে। প্রতিটি printf তোমরা এভাবে লিখতে পারো: printf("Value of a is %d\n", a);। এখন printf ফাংশনে ""-এর ভেতর \n লিখলে কী হয় সেটি আমি বলব না। প্রোগ্রামটি চালালেই বুঝতে পারবে।

a-এর সবগুলো মান কিন্তু ঠিকভাবে দেখা যায়নি, যেসব মান - 2147483648 থেকে 2147483647 পর্যন্ত কেবল সেগুলোই ঠিকঠাক প্রিন্ট হবে, কারণ এই রেঞ্জের বাইরের সংখ্যা int টাইপের ভেরিয়েবলে রাখা যায় না। এটি হলো int টাইপের সংখ্যার সীমা। সি-তে int টাইপের ডাটার জন্য মেমোরিতে চার বাইট (byte) জায়গা ব্যবহৃত হয়। চার বাইট মানে বত্রিশ বিট (1 byte = 8 bit)। প্রতি বিটে দুটি জিনিস রাখা যায়, 0 আর 1। দুই বিটে রাখা যায় চারটি সংখ্যা (00, 01, 10, 11)। তাহলে 32 বিটে রাখা যাবে: 2^{32} টা সংখ্যা অর্থাৎ 4294967296 টি সংখ্যা। এখন অর্ধেক ধনাত্মক আর অর্ধেক ঋণাত্মক যদি রাখি, তাহলে - 2147483648 থেকে -1 পর্যন্ত মোট 2147483648 সংখ্যা আবার 0 থেকে 2147483647 পর্যন্ত মোট 2147483648 সংখ্যা, সর্বমোট 4294967296 টি সংখ্যা। আশা করি,

হিসাবটা বুঝতে পেরেছ।

এখন আমরা যোগ করার প্রোগ্রামটি লিখব যেটি সব বাস্তব সংখ্যা (real number) যোগ করতে পারবে। তোমাদের মনে করিয়ে দিই, পূর্ণসংখ্যা হচ্ছে, ... -3, -2, -1, 0, 1, 2, 3 ... ইত্যাদি। আর বাস্তব সংখ্যা হচ্ছে -5, -3, -2.43, 0, 0.49, 2.92 ইত্যাদি (সংখ্যারেখার ওপর সব সংখ্যাই কিন্তু বাস্তব সংখ্যা)।

```
#include <stdio.h>

int main()
{
    double a, b, sum;

    a = 9.5;
    b = 8.743;

    sum = a + b;

    printf("Sum is: %lf\n", sum);

    printf("Sum is: %0.2lf\n", sum);

    return 0;
}
```

প্রোগ্রাম: ২.৯

প্রোগ্রামটি কম্পাইল এবং রান করো। আউটপুট হবে নিচের দুই লাইন:

Sum is: 18.243000
Sum is: 18.24

%lf ব্যবহার করায় প্রথম লাইনে দশমিকের পরে ছয় ঘর পর্যন্ত প্রিন্ট হয়েছে। আবার দ্বিতীয় লাইনে দশমিকের পরে দুই ঘর পর্যন্ত প্রিন্ট হয়েছে, কারণ %0.2lf লিখেছি (তিন ঘর পর্যন্ত প্রিন্ট করতে চাইলে %0.3lf লিখতাম, আবার দশমিক অংশ প্রিন্ট করতে না চাইলে %0.0lf)। double টাইপের ডাটার জন্য 64 বিট ব্যবহৃত হয় এবং 1.7E-308 (1.7×10^{-308}) থেকে 1.7E+308 (1.7×10^{308}) পর্যন্ত ডাটা রাখা যায়। বিস্তারিত হিসাব বুঝতে হলে কম্পিউটার বিজ্ঞানসংক্রান্ত আরও কিছু জ্ঞানবুদ্ধির দরকার, তাই আমি আর এখন সেদিকে যাচ্ছি না।

এখন আমরা আমাদের প্রোগ্রামে এমন ব্যবস্থা রাখতে চাই, যাতে কোন দুটি সংখ্যা যোগ করতে হবে সেটি আমরা কোডের ভেতর লিখব না, ব্যবহারকারীর কাছ থেকে ইনপুট আকারে জেনে নেব। ব্যবহারকারীর (মানে যে প্রোগ্রামটি ব্যবহার করছে) কাছ থেকে ইনপুট নেওয়ার জন্য আমরা scanf ফাংশন ব্যবহার করব (সি-তে আরও ফাংশন আছে এই কাজের জন্য)। তাহলে দেরি না করে প্রোগ্রাম লিখে ফেলি:

```
#include <stdio.h>

int main()
{
    int a, b, sum;

    scanf("%d", &a);
    scanf("%d", &b);

    sum = a + b;

    printf("Sum is: %d\n", sum);
```

```
    return 0;
}
```
প্রোগ্রাম: ২.১০

প্রোগ্রামটি রান করলে দেখবে ফাঁকা স্ক্রিন (blank screen) আসে। তখন তুমি একটি সংখ্যা লিখবে, তারপর স্পেস (space) বা এন্টার (enter) দিয়ে আরেকটি সংখ্যা লিখবে। তারপর আবার এন্টার চাপলে যোগফল দেখতে পাবে।

তোমরা নিশ্চয়ই scanf ফাংশনের ব্যবহার শিখে ফেলেছ। scanf(“%d”, &a); এখানে ডবল কোটেশনের ভেতরে %d দিয়ে scanf-কে বলে দেওয়া হচ্ছে যে একটি ইন্টিজার বা int টাইপের ভেরিয়েবলের মান পড়তে হবে (ব্যবহারকারী কিবোর্ড থেকে ইনপুট দেবে)। আর দেখো a-এর আগে এমপারসেন্ড (&) চিহ্ন ব্যবহার করা হয়েছে, &a দিয়ে বোঝানো হয়েছে যে সংখ্যাটি ইনপুট দেওয়া হবে সেটি a ভেরিয়েবলের মাঝে অ্যাসাইন হবে। তোমরা যখন সি আরেকটু ভালোভাবে শিখবে, তখন &a-এর প্রকৃত অর্থ বুঝতে পারবে, আপাতত আমরা ব্যবহারের দিকেই মনোযোগ দিই। a এবং b-এর মান একটি scanf ফাংশন দিয়েও নেওয়া যেত এভাবে: scanf(“%d %d”, &a, &b);। ভেরিয়েবলের আগে & চিহ্ন না দিলে কী সমস্যা? নিচের প্রোগ্রামটি রান করার চেষ্টা করো, কিছু একটি এরর পাবে। এই মুহূর্তে এররটা ব্যাখ্যা করছি না, কারণ ব্যাখ্যাটা একটু জটিল আর এখন বোঝাতে গেলে তোমরা ভুল বুঝতে পারো এবং পরে আমাকে গালমন্দ করবে।

```
#include <stdio.h>

int main()
{
    int a, b, sum;

    scanf("%d", &a);
    scanf("%d", b);

    sum = a + b;
```

```
    printf("Sum is: %d\n", sum);

    return 0;
}
```
প্রোগ্রাম: ২.১১

এখন আমরা যদি ইনপুট হিসেবে ইন্টিজার না নিয়ে ডবল টাইপের ডাটা নিতে চাইতাম তাহলে কী করতে হতো? scanf-এ %d-এর বদলে %lf ব্যবহার করলেই চলত। তোমরা প্রোগ্রামটি লিখে ফেলো এবং দেখো ঠিকঠাক রান হয় কি না। তারপরে বাকি অংশ পড়া শুরু করো।

আসলে ঠিকঠাক রান হবে না, কারণ ডাটা টাইপও পরিবর্তন করতে হবে। মানে int না লিখে double লিখতে হবে। প্রোগ্রামটি ঠিক করে আবার চালাও।

বইতে যখনই আমি কোনো প্রোগ্রাম লেখতে বলব সেটি যত সহজ কিংবা কঠিনই মনে হোক না কেন, সেটি কম্পিউটারে লিখে কম্পাইল ও রান করতে হবে। এ কাজ না করে সামনে আগানো যাবে না। মনে রাখবে, গাড়ি চালানো শেখার জন্য যেমন গাড়ি চালানোর কোনো বিকল্প নেই, সাঁতার শেখার জন্য যেমন সাঁতার কাটার বিকল্প নেই, তেমনই প্রোগ্রামিং শেখার জন্য প্রোগ্রামিং করার কোনো বিকল্প নেই, শুধু বই পড়ে প্রোগ্রামার হওয়া যায় না।

এবারে আমরা আরেক ধরনের ডাটা টাইপ দেখব, সেটি হচ্ছে char (character) টাইপ। তো এই character টাইপের চরিত্র হচ্ছে একে মেমোরিতে রাখার জন্য মাত্র এক বাইট জায়গার দরকার হয়। সাধারণত যেকোনো অক্ষর বা চিহ্ন রাখার জন্য এই টাইপের ডাটা ব্যবহার করা হয়। তবে সেই অক্ষরটা ইংরেজি বর্ণমালার অক্ষর হতে হবে, অন্য ভাষার অক্ষর char টাইপের ভেরিয়েবলে রাখা যাবে না। নিচের প্রোগ্রামটি কম্পিউটারে লিখে রান করাও:
```
#include <stdio.h>

int main()
```

```
{
    char ch;

    printf("Enter the first letter of your name: ");
    scanf("%c", &ch);
    printf("The first letter of your name is: %c\n", ch);

    return 0;
}
```
প্রোগ্রাম: ২.১২

কোড দেখে বুঝতেই পারছ, char টাইপের জন্য printf এবং scanf ফাংশনের ভেতরে %c ব্যবহার করতে হয়। আরেকটি ফাংশন আছে getchar, এটি দিয়েও char টাইপের ডাটা রিড করা যায়। নিচের প্রোগ্রামটি দেখো:

```
#include <stdio.h>

int main()
{
    char ch;

    printf("Enter the first letter of your name: ");
    ch = getchar();
    printf("The first letter of your name is: %c\n", ch);

    return 0;
}
```
প্রোগ্রাম: ২.১৩

এটা রান করাও। আগের প্রোগ্রামটির মতো একই কাজ করবে। getchar ফাংশন একটি অক্ষর পড়ে সেটি ch ভেরিয়েবলের ভেতরে অ্যাসাইন করে দিল। আর সরাসরি কোনো কিছু char টাইপ ভেরিয়েবলে রাখতে চাইলে যেই অক্ষর বা চিহ্ন রাখবে তার দুই পাশে সিঙ্গেল কোটেশন চিহ্ন দেবে। যেমন: char c = 'A';

এখন নিচের প্রোগ্রামটি দেখো:

```
#include <stdio.h>

int main()
{
    int num1, num2;

    printf("Please enter a number: ");
    scanf("%d", &num1);
    printf("Please enter another number: ");
    scanf("%d", &num2);

    printf("%d + %d = %d\n", num1, num2, num1+num2);
    printf("%d - %d = %d\n", num1, num2, num1-num2);
    printf("%d * %d = %d\n", num1, num2, num1*num2);
    printf("%d / %d = %d\n", num1, num2, num1/num2);

    return 0;
}
```
প্রোগ্রাম: ২.১৪

এটি কম্পাইল ও রান করাও। এটি দেখে নিশ্চয়ই বুঝতে পারছ বিয়োগ, গুণ ও ভাগের কাজ কীভাবে করতে হয়। এবারে তোমাদের কাজ হচ্ছে চারটি। এক, num1 ও num2-এর মধ্যেকার যোগ, বিয়োগ, গুণ, ভাগের কাজটি printf ফাংশনের ভেতরে না করে আগে করা এবং মানটি অন্য একটি ভেরিয়েবলে রেখে দেওয়া। এর জন্য একটি প্রোগ্রাম লিখে ফেলো। দ্বিতীয় কাজ হচ্ছে প্রোগ্রামটি ডবল টাইপের ভেরিয়েবল ব্যবহার করে করো। তৃতীয় কাজ হচ্ছে, num2-এর মান 0 দিয়ে দেখো কী হয়। চতুর্থ কাজটি হচ্ছে printf ফাংশনের ভেতরে ডবল কোটেশনের ভেতরে যেই +, -, *, / চিহ্ন আছে সেগুলো সরাসরি ব্যবহার না করে একটি char টাইপ ভেরিয়েবলে রেখে তারপর ব্যবহার করা। চারটি কাজ ঠিকমতো করার পরে নিচের প্রোগ্রামটি দেখো:

```
#include <stdio.h>

int main()
{
    int num1, num2, value;
    char sign;

    printf("Please enter a number: ");
    scanf("%d", &num1);
    printf("Please enter another number: ");
    scanf("%d", &num2);

    value = num1 + num2;
    sign = '+';
    printf("%d %c %d = %d\n", num1, sign, num2, value);

    value = num1 - num2;
    sign = '-';
    printf("%d %c %d = %d\n", num1, sign, num2, value);
```

```
    value = num1 * num2;
    sign = '*';
    printf("%d %c %d = %d\n", num1, sign, num2, value);

    value = num1 / num2;
    sign = '/';
    printf("%d %c %d = %d\n", num1, sign, num2, value);

    return 0;
}
```
প্রোগ্রাম: ২.১৫

প্রোগ্রামটি দেখলেই বুঝতে পারবে কী কাজ করে। তবে শুধু দেখে বুঝলেই হবে না। কোড করে কম্পাইল ও রান করো।

সি ল্যাঙ্গুয়েজে আরও বেশ কিছু ডাটা টাইপ রয়েছে। ইনপুট ও আউটপুটের জন্যও রয়েছে নানা পদ্ধতি, যা তোমরা আস্তে আস্তে শিখবে (সব হয়তো এ বইয়ে থাকবে না, সি-এর অন্য বই পড়লে জানতে পারবে)। আপাতত যা শিখেছ, তা দিয়েই তোমরা অনেক প্রোগ্রাম লিখে ফেলতে পারবে।
এখন একটি মজার এবং দরকারি জিনিস বলে রাখি। প্রোগ্রামের কোডের ভেতরে তুমি নিজের ভাষাও ব্যবহার করতে পারো। এটিকে বলে কমেন্ট (comment) করা। কম্পাইলার কমেন্টগুলোকে প্রোগ্রামের অংশ ধরবে না। এক লাইনের কমেন্ট হলে // চিহ্ন দিয়ে কমেন্ট শুরু করতে পারো। আর একাধিক লাইন থাকলে /* দিয়ে শুরু এবং */ দিয়ে শেষ করতে হবে। নিচের প্রোগ্রামটি কিন্তু ঠিকঠাক কম্পাইল ও রান হবে।

```
#include <stdio.h>

int main()
```

```
{
    // test program - comment 1
    printf("Hello ");
    /* We have printed Hello,
    now we shall print World.
    Note that this is a multi-line comment */
    printf("World"); // printed world
    return 0;
}
```

প্রোগ্রাম: ২.১৬

এবারে একটি প্রশ্ন, (যেটি সি-এর টেক্সট বইয়ে এই চ্যাপ্টারের শুরুতেই বলে দিত), ভেরিয়েবলগুলোর নামকরণের নিয়মকানুন কী? ভেরিয়েবলের নাম এক বা একাধিক অক্ষরের হতে পারে, অক্ষরগুলো হতে পারে a থেকে z, A থেকে Z, 0 থেকে 9 এবং _ (আন্ডারস্কোর বা আন্ডারবার)। তবে একাধিক অক্ষরের ক্ষেত্রে প্রথম অক্ষরটা অঙ্ক (ডিজিট) হতে পারবে না। তুমি একটি প্রোগ্রামে int 7d; লিখে দেখো কম্পাইলার কী বলে। আর ভেরিয়েবলের নামগুলো অর্থপূর্ণ হলে ভালো হয়। যেমন, যোগফল রাখার জন্য ভেরিয়েবলের নাম sum হলেই ভালো, যদিও y নাম দিলেও প্রোগ্রাম চলে। অর্থপূর্ণ নাম দিলে বুঝতে সুবিধা হয়, ভেরিয়েবলটা কী উদ্দেশ্যে ব্যবহার করা হয়েছে।

প্রোগ্রামিং শেখার একমাত্র উপায় হচ্ছে বেশি বেশি প্রোগ্রামিং করা। গাড়ী না চালালে যেমন গাড়ী চালানো শেখা যায় না, তেমনি প্রোগ্রামিং না করলে প্রোগ্রামিং শেখা যায় না।

অধ্যায় তিনঃ কন্ডিশনাল লজিক

তোমরা অনেকেই হয়তো জানো যে 'চাচা চৌধুরীর বুদ্ধি কম্পিউটারের চেয়েও প্রখর'! এটি শুনে প্রথম প্রথম চাচা চৌধুরীর ওপর ভক্তি-শ্রদ্ধা অনেক বেড়ে গেলেও একটু চিন্তা করলেই তোমরা বুঝতে পারবে যে আসলে তোমাদের সবার বুদ্ধি কম্পিউটারের চেয়ে প্রখর। আসলে কম্পিউটারের তো কোনো বুদ্ধিই নেই। প্রোগ্রামাররা যেভাবে প্রোগ্রাম লিখে দেয় কম্পিউটার সেভাবে কাজ করে। এখন আমরা প্রোগ্রামিংয়ের সহজ অথচ খুব গুরুত্বপূর্ণ একটি বিষয় শিখব। সেটি হচ্ছে কন্ডিশনাল লজিক। কোন শর্তে কী করতে হবে সেটি প্রোগ্রাম লিখে কম্পিউটারকে বোঝাতে হবে। কথা না বাড়িয়ে আমরা প্রোগ্রাম লেখা শুরু করে দিই। তোমরা কিন্তু অবশ্যই প্রতিটি প্রোগ্রাম কম্পিউটারে চালিয়ে দেখবে।

```
#include <stdio.h>

int main()
{
    int n;

    n = 10;

    if(n >= 0) {
        printf("The number is positive\n");
    }
    else {
        printf("The number is negative\n");
    }

    return 0;
}
```

প্রোগ্রাম: ৩.১

ওপরের প্রোগ্রামটির কাজ কী? n-এর বিভিন্ন মান (যেমন: 0, -10, -2, 5, 988 ইত্যাদি) বসিয়ে তোমরা প্রোগ্রামটি চালাও। দেখা যাচ্ছে যে এর কাজ হচ্ছে n ধনাত্মক (positive) না ঋণাত্মক (negative) সেটি নির্ণয় করা। কোন সংখ্যা ধনাত্মক হতে গেলে একটি শর্ত পূরণ করতে হয়। সেটি হচ্ছে তাকে শূন্যের সমান বা তার চেয়ে বড় হতে হয়। তাহলে আমাদের লজিকটি দাঁড়াচ্ছে এ রকম যে, 'n যদি শূন্যের সমান বা বড় হয়, তবে n ধনাত্মক, না হলে n ঋণাত্মক'। এই ব্যাপারটি সি ল্যাঙ্গুয়েজে প্রকাশ করতে হয় if এবং তার সঙ্গে else ব্যবহার করে। if-এর ভেতর একটি শর্ত (কন্ডিশন) লিখে দিতে হয় যা সত্যি হলেই কেবল তার ভেতরের অংশের কাজ হয় (মানে if-এর পর যে দ্বিতীয় বন্ধনী { } ব্যবহার করা হয়েছে তার ভেতরের সব কাজ)। আর কন্ডিশনটা লিখতে হয় প্রথম বন্ধনীর ভেতরে। if-এর ভেতরে যেই কন্ডিশনটা আছে সেটি যদি মিথ্যা হয়, তবে else-এর ভেতরের (দ্বিতীয় বন্ধনীর ভেতরে) অংশের কাজ হয়। সব প্রোগ্রামিং ল্যাঙ্গুয়েজেই এটি আছে, তবে লেখার ধরন হয়তো আলাদা।

এখন আমাদের দেখতে হবে, কন্ডিশনগুলো কীভাবে লিখতে হবে? তোমরা এতক্ষণে জেনে গেছ যে 'বড় কিংবা সমান' এই তুলনা করার জন্য >= চিহ্ন ব্যবহার করা হয়। 'ছোট কিংবা সমান'-এর জন্য ব্যবহার করে <= চিহ্ন। দুটি সংখ্যা একটি আরেকটির সমান কি না সেটি পরীক্ষার জন্য ব্যবহার করে == চিহ্ন (লক্ষ করো এখানে দুটি সমান চিহ্ন আছে। শুরুর দিকে অনেকেই সমান কি না পরীক্ষার জন্য ভুল করে = (একটি সমান চিহ্ন যা দিয়ে আসলে কোনো ভেরিয়েবলে কোনোকিছু অ্যাসাইন করা হয়) ব্যবহার করে বিপদে পড়ে যায়)। দুটি সংখ্যা পরস্পর অসমান কি না, এটি পরীক্ষার জন্য != চিহ্ন ব্যবহার করে। আর ছোট কিংবা বড় পরীক্ষার জন্য < আর > চিহ্ন ব্যবহার করতে হয়। আরও ব্যাপার-স্যাপার আছে। একবারে সব না শিখে চলো আস্তে আস্তে প্রোগ্রাম লিখে শেখা যাক। এখানে ইন্ডেন্টেশনের ব্যাপারটিও কিন্তু খেয়াল কোরো। if কিংবা else-এর ভেতরের ব্লকের সব লাইন কিন্তু if বা else যেখানে শুরু, তার চার ঘর (চারটি স্পেস) ডান থেকে শুরু হয়েছে।
আমরা ওপরের প্রোগ্রামটি এভাবেও লিখতে পারতাম:

```
#include <stdio.h>

int main()
{
    int n;
    n = 10;
```

```
    if(n < 0) {
        printf("The number is negative\n");
    }
    else {
        printf("The number is positive\n");
    }

    return 0;
}
```

প্রোগ্রাম: ৩.২

এখানে আমরা প্রথমে পরীক্ষা করেছি যে n শূন্যের চেয়ে ছোট কি না। যদি ছোট হয়, তবে n নেগেটিভ; আর সেটি না হলে (সেটি না হওয়া মানে n অবশ্যই শূন্যের সমান বা বড়) n পজিটিভ।

তোমাদের মধ্যে যারা একটু খুঁতখুঁতে স্বভাবের, তারা নিশ্চয়ই ভাবছ যে শূন্য তো আসলে পজিটিভ বা নেগেটিভ কোনোটাই না। শূন্যের চেয়ে বড় সব সংখ্যা হচ্ছে পজিটিভ আর ছোট সব সংখ্যা হচ্ছে নেগেটিভ। কম্পিউটারকে সেটি বোঝাতে গেলে আমরা নিচের প্রোগ্রামটি লিখতে পারি:

```
#include <stdio.h>

int main()
{
    int n = 10;

    if(n < 0) {
        printf("The number is negative\n");
    }
```

```
    else if (n > 0) {
        printf("The number is positive\n");
    }
    else if (n == 0) {
        printf("The number is zero!\n");
    }

    return 0;
}
```
প্রোগ্রাম: ৩.৩

প্রোগ্রামটি একটু ব্যাখ্যা করা যাক:
if(n < 0): এখানে আমরা প্রথমে পরীক্ষা করেছি n শূন্যের চেয়ে ছোট কি না । ছোট হলে তো কথাই নেই। আমরা বলে দিচ্ছি যে সংখ্যাটি নেগেটিভ।
else if(n > 0): আর যদি ছোট না হয়, তবে n শূন্যের চেয়ে বড় কি না সেটি পরীক্ষা করেছি if(n > 0)। এই শর্ত সত্যি হলে সংখ্যাটি পজিটিভ।
else if(n == 0): আর n > 0 ও যদি সত্যি না হয় তবে কোন শর্তটি বাদ রইল? দুটি সমান কি না সেটি পরীক্ষা করা। তাহলে আমরা পরীক্ষা করছি যে n শূন্যের সমান কি না এবং সমান হলে বলে দিচ্ছি যে এটি শূন্য।

দুটি সংখ্যা তুলনা করার সময় প্রথমটা যদি দ্বিতীয়টির চেয়ে বড় না হয়, ছোটও না হয়, তবে তারা অবশ্যই সমান। তাই তৃতীয় কন্ডিশনটা আসলে আমাদের দরকার নেই। আমরা প্রথম দুটি কন্ডিশন মিথ্যা হলেই বলে দিতে পারি যে n-এর মান শূন্য।

```
#include <stdio.h>

int main()
{
```

```
    int n = 10;

    if(n < 0) {
        printf("The number is negative\n");
    }
    else if (n > 0) {
        printf("The number is positive\n");
    }
    else {
        printf("The number is zero!\n");
    }

    return 0;
}
```
প্রোগ্রাম: ৩.৪

আবার সব সময় যে if ব্যবহার করলেই সঙ্গে else কিংবা else if ব্যবহার করতে হবে, এমন কোনো কথা নেই। নিচের প্রোগ্রামটি দেখো:

```
#include <stdio.h>

int main()
{
    int number = 12;

    if(number > 10) {
        printf("The number is greater than ten\n");
```

```
    }
    return 0;
}
```
প্রোগ্রাম: ৩.৫

এখানে কেবল দেখা হচ্ছে যে সংখ্যাটির মান কি দশের চেয়ে বড় কি না।

নিচের প্রোগ্রামটি দেখে বলো তো আউটপুট কী হবে?

```
#include <stdio.h>

int main()
{
    int n = 10;

    if (n < 30) {
        printf("n is less than 30.\n");
    }
    else if(n < 50) {
        printf("n is less than 50.\n");
    }

    return 0;
}
```
প্রোগ্রাম: ৩.৬

আউটপুট হবে: n is less than 30. যদিও else if(n < 50) এটিও সত্য কিন্তু যেহেতু if (n < 30) সত্য হয়ে গেছে, তাই এর সঙ্গে

বাকি যত else if কিংবা else থাকবে, সেগুলো আর পরীক্ষা করা হবে না। এখন তোমরা নিশ্চয়ই নিচের প্রোগ্রামটির আউটপুট বলতে পারবে।

```
#include <stdio.h>

int main()
{
    int n = 10;

    if (n < 30) {
        printf("n is less than 30.\n");
    }
    if(n < 50) {
        printf("n is less than 50.\n");
    }

    return 0;
}
```
প্রোগ্রাম: ৩.৭

এখন আমরা আরেকটি প্রোগ্রাম লিখব। কোনো সংখ্যা জোড় না বেজোড় সেটি নির্ণয় করার প্রোগ্রাম। কোনো সংখ্যা জোড় কি না সেটি বোঝার উপায় হচ্ছে সংখ্যাটিকে 2 দিয়ে ভাগ করা। যদি ভাগশেষ শূন্য হয়, তবে সংখ্যাটি জোড়; আর ভাগশেষ শূন্য না হয়ে এক হলে সেটি বেজোড়। সি ল্যাঙ্গুয়েজে ভাগশেষ বের করার জন্য মডুলাস অপারেটর (modulus operator) আছে, যেটাকে '%' চিহ্ন দিয়ে প্রকাশ করা হয়। তাহলে আর চিন্তা নেই।

শুরুতে একটি সংখ্যা নেব: int number;

এবারে number-এর জন্য একটি মান ঠিক করি: number = 5;
এখন number কে 2 দিয়ে ভাগ করলে যে ভাগশেষ পাব সেটি বের করি: remainder = number % 2;
এখন if-এর সাহায্যে remainder-এর মান পরীক্ষা করে আমরা সিদ্ধান্তে পৌঁছে যেতে পারি। remainder-এর কেবল দুটি মানই সম্ভব– 0 আর 1। পুরো প্রোগ্রামটি লিখে ফেলি:

```
#include <stdio.h>

int main()
{
    int number, remainder;

    number = 5;

    remainder = number % 2;

    if(remainder == 0) {
        printf("The number is even\n");
    }
    else {
        printf("The number is odd\n");
    }

    return 0;
}
```
প্রোগ্রাম: ৩.৮

প্রোগ্রামটি remainder ভেরিয়েবল ব্যবহার না করেও লেখা যায়:

```
#include <stdio.h>

int main()
{
    int number = 9;

    if(number % 2 == 0) {
        printf("The number is even\n");
    }
    else {
        printf("The number is odd\n");
    }

    return 0;
}
```
প্রোগ্রাম: ৩.৯

আচ্ছা, আমাদের যদি কেবল জোড় সংখ্যা নির্ণয় করতে হতো, তাহলে আমরা কী করতাম? else ব্লকটা বাদ দিয়ে দিতাম।

তোমাদের জন্য এখন একটি ছোট্ট পরীক্ষা। মডুলাস অপারেটর ব্যবহার না করে ভাগশেষ বের করতে পারবে? একবার করে গুণ, ভাগ ও বিয়োগ (*, /, -) ব্যবহার করে কিন্তু কাজটি করা যায়। তোমরা সেটি করার চেষ্টা করতে পারো।

এবার আরেকটি প্রোগ্রাম দেখা যাক। কোনো একটি অক্ষর ছোট হাতের (small letter বা lower case letter) নাকি বড় হাতের (capital letter বা upper case letter), সেটি বের করতে হবে। এর জন্য সবচেয়ে সহজ সমাধানটা হতে পারে এই রকম যে আমরা একটি character টাইপের ভেরিয়েবলের ভেতরে অক্ষরটা রাখতে পারি। তারপর একে একে সেটিকে 26 টি lower case letter এবং 26 টি upper case letter-এর সঙ্গে তুলনা করে দেখতে পারি। যখনই মিলে যাবে, তখনই বলে দেওয়া যায়, অক্ষরটা

কোন ধরনের।

```
char ch = 'p';

if (ch == 'a')
{
    printf("%c is lower case\n", ch);
}
else if (ch == 'A')
{
    printf("%c is upper case\n", ch);
}
else if (ch == 'b')
{
    printf("%c is lower case\n", ch);
}
else if (ch == 'B')
{
    printf("%c is upper case\n", ch);
}
else if (ch == 'c')
{
    printf("%c is lower case\n", ch);
}
else if (ch == 'C')
{
    printf("%c is upper case\n", ch);
}
```

... এভাবে চলবে।

কিন্তু এই সমস্যার সমাধান করার জন্য এত কোড লেখার কোনো দরকার নেই। এটি সহজে করার জন্য আমাদের জানতে হবে এন্ড অপারেটরের (AND operator) ব্যবহার। সি ল্যাঙ্গুয়েজে একে '&&' চিহ্ন দিয়ে প্রকাশ করা হয়। নিচের কোডটি দেখলেই তোমরা এর কাজ বুঝে যাবে।

```
#include <stdio.h>

int main()
{
    char ch = 'W';

    if(ch >= 'a' && ch <= 'z') {
        printf("%c is lower case\n", ch);
    }
    if(ch >= 'A' && ch <= 'Z') {
        printf("%c is upper case\n", ch);
    }

    return 0;
}
```

প্রোগ্রাম: ৩.৯

'&&'-এর বাঁ পাশে একটি কন্ডিশন এবং ডান পাশে একটি কন্ডিশন থাকবে, এবং দুটি কন্ডিশন সত্য হলেই সম্পূর্ণ কন্ডিশনটা সত্য হবে। ch >= 'a' && ch <= 'z' এটি পুরোটা একটি কন্ডিশন। এখন &&-এর বাঁ দিকে একটি কন্ডিশন আছে ch >= 'a' আর ডানদিকে আরেকটি কন্ডিশন ch <= 'z'। দুটি কন্ডিশনই যদি সত্য হয়, তবে পুরো কন্ডিশনটা সত্য হবে। এখন কম্পিউটার প্রতিটি অক্ষর বোঝার জন্য যেই কোড ব্যবহার করে তাতে a-এর চেয়ে b-এর মান এক বেশি, b-এর চেয়ে c-এর মান এক বেশি, c-এর চেয়ে d-এর মান এক বেশি ... এরকম। তাই কোনো অক্ষর lower case হলে সেটি অবশ্যই 'a'-এর সমান কিংবা বড় হতে হবে। আবার

সেটি 'z'-এর সমান কিংবা ছোট হতে হবে। একইভাবে A-এর চেয়ে B-এর মান এক বেশি, B-এর চেয়ে C-এর মান এক বেশি ... এরকম। তাই কোনো ক্যারেক্টারের মান 'A' থেকে 'Z'-এর মধ্যে হলে আমরা বলতে পারি যে সেটি upper case। 'A'-এর সমান কিংবা বড় হতে হবে এবং 'Z'-এর সমান কিংবা ছোট হতে হবে।
আরেকটি ব্যাপার। দ্বিতীয় if-এর আগে else ব্যবহার করা উচিত। তাহলে কম্পাইলার প্রথম if-এর ভেতরের শর্ত সত্য হলে আর পরের if-এর কন্ডিশন পরীক্ষা করবে না। তাতে সময় বাঁচবে।

```
#include <stdio.h>

int main()
{
    char ch = 'k';

    if(ch >= 'a' && ch <= 'z') {
        printf("%c is lower case\n", ch);
    }
    else if(ch >= 'A' && ch <= 'Z') {
        printf("%c is upper case\n", ch);
    }

    return 0;
}
```
প্রোগ্রাম: ৩.১০

আশা করি, তোমরা '&&'-এর ব্যবহার বুঝে গেছ।

এখন আরেকটি অপারেটরের ব্যবহার দেখব। সেটি হচ্ছে অর (OR)। একে প্রকাশ করা হয় '||' চিহ্ন দিয়ে (পরপর দুটি |)। '&&'-এর ক্ষেত্রে যেমন দুই পাশের শর্ত সত্য হলেই সম্পূর্ণ শর্ত সত্য হয়, '||'-এর ক্ষেত্রে যেকোনো এক পাশের শর্ত সত্য হলেই সম্পূর্ণ শর্ত সত্য

হয়।
নিচের প্রোগ্রামটির আউটপুট কী হবে? কোড দেখে বলতে না পারলে প্রোগ্রামটি চালাও।

```
#include <stdio.h>

int main()
{
    int num = 5;

    if(num >= 1 || num <= 10) {
        printf("yes\n");
    }
    else {
        printf("no\n");
    }

    return 0;
}
```

প্রোগ্রাম: ৩.১১

এটির আউটপুট হবে yes। এখন num-এর মান 50 করে দাও। আউটপুট কী হবে?

এবারেও আউটপুট yes ই হবে। কারণ num-এর মান 50 হলে, প্রথম শর্তটি সত্য হবে (num >= 1) আর দ্বিতীয় শর্তটি (n <= 10) মিথ্যা হবে। তবে আমরা যেহেতু দুটি শর্তের মাঝে '||' ব্যবহার করেছি, তাই যেকোনো একটি শর্ত সত্য হলেই সম্পূর্ণ শর্তটি সত্য হবে।

এখন আরও একটি সমস্যা। কোনো অক্ষর vowel নাকি consonant, সেটি নির্ণয় করতে হবে।
আমরা জানি, vowel গুলো হচ্ছে a, e, i, o, u। এখন কোনো ক্যারেক্টার এই পাঁচটির মধ্যে পড়ে কি না সেটি নির্ণয় করার জন্য যদি

আমরা এমন শর্ত দিই: ch >= 'a' && ch <= 'u' তাহলে কিন্তু হবে না। কারণ তাহলে a থেকে u পর্যন্ত সব অক্ষরের জন্যই শর্তটি সত্যি হবে কিন্তু আমাদের দরকার নির্দিষ্ট কিছু অক্ষর। তাই শর্তটি আমরা এভাবে লিখতে পারি:

```
if(ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
    printf("%c is vowel\n", ch);
}
else {
    printf("%c is consonant\n", ch);
}
```

তাহলে এবার সম্পূর্ণ প্রোগ্রামটি তোমরা লিখে ফেলতে পারো।

dimik

দ্বিমিক কম্পিউটিং স্কুল নিয়ে আসছে প্রোগ্রামিং শেখার বিভিন্ন
অনলাইন কোর্স। ওয়েবসাইটঃ http://dimikcomputing.com

অধ্যায় চারঃ লুপ (Loop)

তোমরা এরই মধ্যে প্রোগ্রামের মধ্যে বিভিন্ন ধরনের শর্ত (condition) ব্যবহার করতে শিখে গেছ। এইসব শর্ত দিয়ে বিভিন্ন প্রোগ্রাম তৈরি করাও হয়তো শুরু করে দিয়েছ। খুব ভালো কথা। কিন্তু এখন আমরা আরেকটি সমস্যা ও তার সমাধানের পথ খুঁজব। একটি প্রোগ্রাম লিখতে হবে, যেটি 1 থেকে 10 পর্যন্ত সব পূর্ণসংখ্যা মনিটরে দেখাবে (প্রতি লাইনে একটি সংখ্যা থাকবে)। খুবই সহজ সমস্যা এবং সমাধানও অত্যন্ত সহজ। আমি জানি, তোমরা এক মিনিটের মধ্যেই নিচের প্রোগ্রামটি লিখে ফেলবে:

```
#include <stdio.h>

int main()
{
    printf("1\n");
    printf("2\n");
    printf("3\n");
    printf("4\n");
    printf("5\n");
    printf("6\n");
    printf("7\n");
    printf("8\n");
    printf("9\n");
    printf("10\n");

    return 0;
}
```
প্রোগ্রাম: ৪.১

এখানে আমরা 1 থেকে 10 পর্যন্ত সবগুলো সংখ্যা প্রিন্ট করে দিয়েছি। অবশ্য একটি printf() ব্যবহার করেও কাজটি করা যেত:

```
printf("1\n2\n3\n4\n5\n6\n7\n8\n9\n10\n");
```

আবার প্রোগ্রামটি এভাবেও লেখা যেত। n একটি ইন্টিজার ভেরিয়েবল, যার মান আমরা প্রথমে 1 বসাব। তারপর n-এর মান প্রিন্ট করব। তারপর n-এর মান এক বাড়াব (n = n + 1 অথবা সংক্ষেপে, n++ লিখে)।

```
int n = 1;
printf("%d\n", n);
n = n + 1;
printf("%d\n", n);
n = n + 1;
printf("%d\n", n);
n = n + 1;
/* এভাবে মোট দশ বার */
```

আবার n এর মান 1 বাড়ানোর কাজটি কিন্তু এক লাইনেই সেরে ফেলা যায়।

```
printf("%d\n", n);
n = n + 1;
```

এর পরিবর্তে আমরা লিখতে পারি:

```
printf("%d\n", n++);
```

যা-ই হোক, এ তো গেল 1 থেকে 10 পর্যন্ত প্রিন্ট করা। কিন্তু আমাদের যদি 1 থেকে 100, বা 1000, বা 10000 পর্যন্ত প্রিন্ট করতে বলা হতো তাহলে আমরা কী করতাম? ওপরে যে পদ্ধতি অবলম্বন করা হয়েছে সেটি তো অবশ্যই করা যেত। কিন্তু আমি জানি, তোমরা কেউই এত কষ্ট করতে রাজি না।

এ সমস্যা সমাধানের জন্য সব প্রোগ্রামিং ল্যাঙ্গুয়েজেই লুপ (loop) বলে একটি পদ্ধতি রয়েছে। এটি দিয়ে একই কাজ বারবার করা যায়। লুপের মধ্যে একটি শর্ত বসিয়ে দিতে হয়, যেটি পূরণ না হওয়া পর্যন্ত প্রোগ্রামটি লুপের ভেতরের কাজ বারবার করতে থাকবে। সি

ল্যাঙ্গুয়েজে দুটি জনপ্রিয় লুপ হচ্ছে while এবং for। আমরা এখন while ব্যবহার করে ওই প্রোগ্রামটি লিখব।

```
#include <stdio.h>

int main()
{
    int n = 1;
    while(n <= 10) {
        printf("%d\n", n);
        n++;
    }
    return 0;
}
```

প্রোগ্রাম: ৪.২

কী চমৎকার! এখন আমরা চাইলে 10-এর বদলে যত খুশি বসাতে পারি, যত বসাব 1 থেকে তত পর্যন্ত প্রিন্ট হবে। while লুপে প্রথম বন্ধনীর ভেতর শর্ত লিখে দিতে হয়। প্রোগ্রাম সেই শর্ত পরীক্ষা করে। যতক্ষণ পর্যন্ত শর্তটি সত্য হয় ততক্ষণ পর্যন্ত লুপের ভেতরের কাজগুলো চলতে থাকে। লুপের ভেতরের কাজগুলো থাকবে দ্বিতীয় বন্ধনীর ভেতর। যেমন এখানে লুপের ভেতরে আমরা দুটি কাজ করেছি। n-এর মান প্রিন্ট করেছি আর তারপর n-এর মান 1 বাড়িয়েছি। n-এর মান 1 করে বাড়তে থাকলে একসময় এটি 11 হবে আর তখন n <= 10 এই শর্তটি মিথ্যা হয়ে যাবে (কারণ 11 > 10)। আর প্রোগ্রামটিও লুপ থেকে বের হয়ে আসবে। অর্থাৎ, শর্তটি যখনই মিথ্যা হবে তখনই লুপ থেকে বের হয়ে যাবে।

ইন্ডেন্টেশনের ব্যাপারটিও খেয়াল করো। লুপের ভেতরের অংশের কোড চার ঘর ডানদিক থেকে শুরু হয়েছে।

এবারে তোমাদের জন্য একটি প্রশ্ন। বলো তো নিচের প্রোগ্রামটির আউটপুট কী হবে?

```
#include <stdio.h>

int main()
{
    int n = 1;
    while(n <= 10) {
        printf("%d\n", n);
    }
    n++;

    return 0;
}
```
প্রোগ্রাম: ৪.৩

এটাও কি 1 থেকে 10 পর্যন্ত সব সংখ্যা প্রিন্ট করবে? দেখা যাক। প্রোগ্রামটি রান করাও। আউটপুট কী?

মনিটরে প্রতি লাইনে 1 প্রিন্ট হচ্ছে এবং প্রোগ্রামটি বন্ধ হচ্ছে না। খুবই দুঃখের বিষয়। দেখা যাক দুঃখের পেছনে কারণটা কী।

int n = 1; প্রথমে প্রোগ্রামটি n-এর মান 1 বসাবে।
তারপর while লুপে গিয়ে শর্ত পরীক্ষা করবে। আমরা শর্ত দিয়েছি n <= 10 মানে n-এর মান 10-এর ছোট বা সমান। এই শর্ত তো সত্য কারণ n-এর মান 1। তারপর প্রোগ্রামটি n-এর মান প্রিন্ট করবে printf("%d\n", n);। তারপর কি n-এর মান 1 বাড়বে? বাড়বে না, কারণ আমরা দ্বিতীয় বন্ধনী শেষ করে দিয়েছি '}' চিহ্ন দিয়ে (মানে লুপ শেষ)। তার মানে প্রোগ্রামটি আবার শর্ত পরীক্ষা করবে, আবার n-এর মান প্রিন্ট করবে...এভাবে চলতেই থাকবে কারণ n-এর মান যেহেতু বাড়ছে না, n <= 10 শর্তটি সব সময় সত্যই রয়ে যাচ্ছে – কখনো মিথ্যা হচ্ছে না।

এখন তোমরা while লুপ নিয়ে বিভিন্ন ধরনের গবেষণা চালিয়ে যেতে পারো। সব সময় সত্য হয় এমন শর্ত ব্যবহার করে তোমার

কম্পিউটারকে ব্যস্ত রাখতে পারো। while(1){...} এখানে শর্ত হিসেবে 1 ব্যবহার করা হয়েছে। কম্পিউটার 1 বলতে বোঝে সত্য। সুতরাং লুপের ভেতরের কাজগুলো সব সময় চলতে থাকবে, বন্ধ হবে না। while(1 == 1){...} ও একই আচরণ করবে।

তবে এখন আমি তোমাদের একটি দরকারি জিনিস বলে রাখি, যেটি দিয়ে তোমরা জোর করে লুপ থেকে বের হয়ে যেতে পারবে। সেটি হচ্ছে break স্টেটমেন্ট। কথা না বাড়িয়ে একটি প্রোগ্রাম লিখলেই ব্যাপারটি পরিষ্কার হয়ে যাবে।

```
#include <stdio.h>

int main()
{
    int n = 1;
    while(n <= 100) {
        printf("%d\n", n);
        n++;
        if(n > 10) {
            break;
        }
    }
    return 0;
}
```

প্রোগ্রাম: 8.8

এই প্রোগ্রামটি কী করবে? 1 থেকে 10 পর্যন্ত প্রিন্ট করবে। যদিও while-এর ভেতর আমরা বলেছি যে শর্ত হচ্ছে n <= 100, কিন্তু লুপের ভেতরে আবার বলে দিয়েছি যে যদি n > 10 হয়, তবে break; মানে বের হয়ে যাও, বা লুপটি ভেঙে দাও। break সব সময় যেই লুপের ভেতর থাকে সেটির বাইরে প্রোগ্রামটিকে নিয়ে আসে। সুতরাং n-এর মান 10 প্রিন্ট হওয়ার পরে এর মান এক বাড়বে (n++;) অর্থাৎ n-এর মান হবে 11। আর তখন n > 10 সত্য হবে, ফলে প্রোগ্রামটি if কন্ডিশনের ভেতরে ঢুকে যাবে। সেখানে গিয়ে সে

দেখবে তাকে break করতে বলা হয়েছে তাই সে লুপের বাইরে চলে যাবে। break-এর উল্টা কাজ করে, এমন একটি স্টেটমেন্ট হচ্ছে continue;। কোনো জায়গায় continue ব্যবহার করলে লুপের ভেতরে continue-এর পরের অংশের কাজ আর হয় না। নিচের প্রোগ্রামটি কোড করে কম্পাইল ও রান করো:

```
#include <stdio.h>

int main()
{
    int n = 0;

    while (n < 10) {
        n = n + 1;
        if (n % 2 == 0) {
            continue;
        }
        printf("%d\n", n);
    }

    return 0;
}
```

প্রোগ্রাম: ৪.৫

এই প্রোগ্রামটি 1 থেকে 10-এর মধ্যে কেবল বেজোড় সংখ্যাগুলো প্রিন্ট করবে। জোড় সংখ্যার বেলায় continue ব্যবহার করার কারণে প্রোগ্রামটি `printf("%d\n", n);` স্টেটমেন্ট এক্সিকিউট না করে লুপের পরবর্তী ধাপের কাজ শুরু করবে।

এবারে আমরা আরেকটি প্রোগ্রাম লিখব। ছোটবেলায় যে নামতাগুলো তোমরা শিখেছ সেগুলো এখন আমরা প্রোগ্রাম লিখে কম্পিউটারের মনিটরে দেখব। চলো 5-এর নামতা দিয়ে শুরু করা যাক। আমাদের প্রোগ্রামের আউটপুট হবে এরকম:

```
5 X 1 = 5
5 X 2 = 10
5 X 3 = 15
5 X 4 = 20
5 X 5 = 25
5 X 6 = 30
5 X 7 = 35
5 X 8 = 40
5 X 9 = 45
5 X 10 = 50
```

তোমরা নিশ্চয়ই এখন অনেকগুলো printf ফাংশন লেখা শুরু করবে না। লুপের সাহায্যে প্রোগ্রামটি লিখে ফেলবে:

```
#include <stdio.h>

int main()
{
    int n = 5;
    int i = 1;

    while (i <= 10) {
        printf("%d X %d = %d\n", n, i, n*i);
        i = i + 1;
    }
    return 0;
}
```

প্রোগ্রাম: ৪.৬

এতক্ষণ আমরা while লুপ ব্যবহার করলাম। এবার চলো for লুপ ব্যবহার করতে শিখি। 5-এর নামতার প্রোগ্রামটি যদি আমরা for

লুপ ব্যবহার করে লিখি তাহলে সেটির চেহারা দাঁড়াবে:

```
#include <stdio.h>

int main()
{
    int n = 5;
    int i;

    for(i = 1; i <= 10; i = i + 1) {
        printf("%d X %d = %d\n", n, i, n*i);
    }

    return 0;
}
```

প্রোগ্রাম: ৪.৭

for লুপের প্রথম বন্ধনীর ভেতর তিনটি অংশ লক্ষ করো। প্রতিটি অংশ সেমিকোলন (;) দিয়ে আলাদা করা হয়েছে। প্রোগ্রামটি যখন লুপের ভেতর ঢুকে তখন প্রথম সেমিকোলনের আগে আমরা যে কাজগুলো করতে বলব, সেগুলো একবার করবে। যেমন এখানে i-এর মান 1 বসাবে। তারপর দ্বিতীয় অংশের কাজ করবে। দ্বিতীয় অংশে সাধারণত শর্ত ব্যবহার করা হয় (while লুপে প্রথম বন্ধনীর ভেতর আমরা যে কাজটি করি আরকি)। ওপরের প্রোগ্রামে আমরা দ্বিতীয় অংশে i <= 10 শর্তটি ব্যবহার করেছি। এই শর্ত যদি মিথ্যা হয় তবে প্রোগ্রামটি লুপ থেকে বেরিয়ে আসবে। আর যদি সত্য হয় তবে লুপের ভেতরের কাজগুলো করবে এবং তার পর for লুপের সেই প্রথম বন্ধনীর ভেতর তৃতীয় অংশে যে কাজগুলো করতে বলা হয়েছে সেগুলো করবে। তারপর আবার দ্বিতীয় অংশে এসে শর্ত পরীক্ষা করবে। প্রথম অংশের কাজ কিন্তু আর হবে না। তো আমাদের প্রোগ্রামটি আবার লক্ষ করো। i <= 10 সত্য, কারণ i-এর মান 1। তারপর printf() ফাংশনের কাজ হবে। তারপর i = i + 1 স্টেটমেন্ট এক্সিকিউট হবে (i-এর মান এক বেড়ে যাবে)। তারপর আবার i <= 10 সত্য না মিথ্যা সেটি পরীক্ষা করা হবে (i-এর মান এখন 2)। তারপর আবার লুপের ভেতরের কাজ হবে (printf())। এভাবে যতক্ষণ না i <= 10 শর্তটি মিথ্যা হচ্ছে ততক্ষণ লুপের ভেতরের কাজ চলতে থাকবে। i-এর মান এক এক করে বেড়ে বেড়ে যখন 11 হবে তখন

শর্তটি মিথ্যা হবে আর প্রোগ্রামটি লুপ থেকে বের হয়ে আসবে।

for লুপের প্রথম বন্ধনীর ভেতরের তিনটি অংশই যে ব্যবহার করতে হবে এমন কোন কথা নেই। কোন অংশ ব্যবহার করতে না চাইলে আমরা সেটি ফাঁকা রেখে দিতে পারি, তবে সেমিকোলন কিন্তু অবশ্যই দিতে হবে। যেমন আমরা যদি i-এর মান আগেই নির্ধারণ করে দেই তবে সেটি লুপের ভেতর না করলেও চলে।

```
int i = 1;
for(; i<= 10; i = i + 1) {
    printf("%d X %d = %d\n", n, i, n*i);
}
```

যদি তিনটি অংশের কোনোটিই লিখতে না চাই, তবে পুরো প্রোগ্রামটি এভাবে লেখা যায়:

```
#include <stdio.h>

int main()
{
    int n = 5;
    int i = 1;

    for( ; ; ) {
        printf("%d X %d = %d\n", n, i, n*i);
        i = i + 1;
        if (i > 10) {
            break;
        }
    }
```

```
    return 0;
}
```
প্রোগ্রাম: ৪.৮

এখন আমরা আরেকটি কাজ করব। for লুপ ব্যবহার করে 5-এর নামতায় যে গুণ করেছি (n*i) সেটি না করে কেবল যোগ করে প্রোগ্রামটি লিখব। তোমরা কি অবাক হচ্ছ যে নামতার প্রোগ্রাম আবার গুণ ছাড়া কীভাবে হবে? আমরা কিন্তু 5 x 3-কে লিখতে পারি 5 + 5 + 5। আমি কী করতে যাচ্ছি তা বুঝতে পারছ নিশ্চয়ই। প্রোগ্রামটি লিখে ফেলি:

```
#include <stdio.h>

int main()
{
    int m, n = 5;
    int i;

    m = 0;
    for(i = 1; i <= 10; i = i + 1) {
        m = m + n;
        printf("%d X %d = %d\n", n, i, m);
    }

    return 0;
}
```
প্রোগ্রাম: ৪.৯

প্রোগ্রামটিতে আমরা গুণ না করে যোগ করলাম। কম্পাইল ও রান করে দেখো। কাজ করবে ঠিকঠাক। কোনো সংখ্যার গুণিতকগুলো যেমন গুণ করে বের করা যায়, তেমনই যোগ করেও করা যায়। আমরা যদি কোনো প্রোগ্রামে দেখি যে গুণ না করে যোগ করলেই কাজ

হচ্ছে, তাহলে যোগ করাই ভালো কারণ কম্পিউটারের প্রসেসর একটি যোগ করতে যে সময় নেয়, একটি গুণ করতে তার চেয়ে অনেক বেশি সময় নেয়। যদিও তুমি হয়তো প্রোগ্রাম রান করার সময় তা বুঝতে পারো না। কম্পিউটারের প্রসেসর সম্পর্কে বিস্তারিত লেখাপড়া করলে বিষয়টা জানতে পারবে। আপাতত এটি জানলেই চলবে যে একটি গুণ করার চেয়ে একটি যোগ করা ভালো, কারণ যোগ করতে কম্পিউটার অপেক্ষাকৃত কম সময় নেয়।

তো আমরা for লুপ শিখে ফেললাম। এখন আমরা চেষ্টা করব শুধু নির্দিষ্ট একটি সংখ্যার নামতা না লিখে 1 থেকে 20 পর্যন্ত সবগুলো সংখ্যার নামতা একবারে লিখে ফেলতে। অর্থাৎ n-এর মান 5 নির্দিষ্ট না করে 1 থেকে 20 পর্যন্ত হবে। এটি করার একটি বোকা পদ্ধতি (নাকি চোরা পদ্ধতি?) হচ্ছে নামতা লেখার অংশটি বারবার কপি-পেস্ট করা। কিন্তু আমরা এটি করব লুপের ভেতর লুপ ব্যবহার করে। একটি লুপের সাহায্যে n-এর মান 1 থেকে 20 পর্যন্ত এক করে বাড়াব। আর তার ভেতর n-এর একটি নির্দিষ্ট মানের জন্য নামতাটা লিখব।

```
#include <stdio.h>

int main()
{
    int n, i;

    for(n = 1; n <= 20; n = n + 1) {
        for(i = 1; i <= 10; i = i + 1) {
            printf("%d X %d = %d\n", n, i, n*i);
        }
    }

    return 0;
}
```
প্রোগ্রাম: ৪.১০

এখন তোমরা প্রোগ্রামটি চালাও। তারপর তোমাদের কাজ হবে গুণ না করে কেবল যোগ ব্যবহার করে প্রোগ্রামটি লেখা।

আমরা এখানে একটি for লুপের ভেতর আরেকটি for লুপ, যাকে নেস্টেড লুপও (nested loop) বলে, সেটি ব্যবহার করলাম। তো আমরা চাইলে for লুপের ভেতর for বা while অথবা while লুপের ভেতর for বা while লুপ একাধিকবার ব্যবহার করতে পারি। অবশ্য সেটি কখনোই চার বা পাঁচবারের বেশি দরকার হওয়ার কথা না। নেস্টেড লুপ দিয়ে আমরা এখন আরেকটি প্রোগ্রাম লিখব। 1, 2, 3 – এই তিনটি সংখ্যার সব বিন্যাস (permutation) বের করার প্রোগ্রাম। বিন্যাসগুলো ছোট থেকে বড় ক্রমে দেখাতে হবে অর্থাৎ প্রোগ্রামটির আউটপুট হবে এই রকম:

```
1, 2, 3
1, 3, 2
2, 1, 3
2, 3, 1
3, 1, 2
3, 2, 1
```

এই প্রোগ্রামটি অনেকভাবে লেখা যেতে পারে, কিন্তু আমরা এখন পর্যন্ত যতটুকু প্রোগ্রামিং শিখেছি, তাতে নেস্টেড লুপের ব্যবহারই সবচেয়ে ভালো সমাধান।

এখানে আমরা প্রথম সংখ্যাটির জন্য একটি লুপ, দ্বিতীয় সংখ্যাটির জন্য প্রথম লুপের ভেতরে একটি লুপ এবং তৃতীয় সংখ্যাটির জন্য দ্বিতীয় লুপের ভেতর আরেকটি লুপ ব্যবহার করব।

```
#include <stdio.h>

int main()
{
```

```
    int a, b, c;
    for (a = 1; a <= 3; a++) {
        for (b = 1; b <= 3; b++) {
            for (c = 1; c <= 3; c++) {
                printf ("%d, %d, %d\n", a, b, c);
            }
        }
    }
   return 0;
}
```
প্রোগ্রাম: ৪.১১

এখন প্রোগ্রামটি রান করলে আমরা এই রকম আউটপুট পাব:

```
1, 1, 1
1, 1, 2
1, 1, 3
1, 2, 1
1, 2, 2
1, 2, 3
1, 3, 1
1, 3, 2
1, 3, 3
2, 1, 1
2, 1, 2
2, 1, 3
2, 2, 1
2, 2, 2
2, 2, 3
```

```
2, 3, 1
2, 3, 2
2, 3, 3
3, 1, 1
3, 1, 2
3, 1, 3
3, 2, 1
3, 2, 2
3, 2, 3
3, 3, 1
3, 3, 2
3, 3, 3
```

কিন্তু আমরা তো আসলে এই রকম জিনিস চাচ্ছি না। a-এর মান যখন 1 তখন b ও c-এর মান 1 হবে না, আবার b এবং c-এর মানও সমান হবে না। মানে a, b ও c আলাদা হবে। তাহলে আমরা লুপের ভেতর শর্তগুলো একটু পরিবর্তন করব। দ্বিতীয় লুপের শর্ত b <= 3-এর সঙ্গে আরেকটি শর্ত জুড়ে দেব, b != a। b <= 3 && b != a মানে b-এর মান 3-এর চেয়ে ছোট বা সমান হবে এবং b-এর মান a-এর মানের সমান হবে না। তৃতীয় লুপে আমরা এখন শর্ত দেব, c <= 3 && c != a && c != b, মানে c-এর মান 3-এর ছোট বা সমান হতে হবে এবং c-এর মান a-এর মানের সমান হওয়া চলবে না এবং c-এর মান b-এর মানের সমান হলেও চলবে না। তাহলে আমাদের প্রোগ্রামটির চেহারা দাঁড়াবে এই রকম:

```
#include <stdio.h>

int main()
{
   int a, b, c;

   for (a = 1; a <= 3; a++) {
```

```
      for (b = 1; b <= 3 && b != a; b++) {
        for (c = 1; c <= 3 && c != a && c != b; c++) {
            printf ("%d, %d, %d\n", a, b, c);
        }
      }
    }

    return 0;
}
```
প্রোগ্রাম: ৪.১২

রান করলে আমরা আউটপুট কী দেখব?
3, 2, 1

মাত্র একটি লাইন! আমরা প্রোগ্রামটি ঠিক করতে গিয়ে ঝামেলা পাকিয়ে ফেলেছি মনে হচ্ছে। তোমরা কি একটু চিন্তা করে ঝামেলার কারণ বের করতে পারবে?

প্রথমে a-এর মান 1 তাই a <= 3 সত্য। প্রোগ্রামটি প্রথম লুপের ভেতর ঢুকে গেল। তারপর দ্বিতীয় লুপের শুরুতে b-এর মান 1। b <= 3 সত্য। কিন্তু b != a মিথ্যা। কারণ a ও b-এর মান তো সমান, দুটোর মানই 1। তাই প্রোগ্রামটি আর দ্বিতীয় লুপের ভেতর ঢুকবে না। এরপর a-এর মান 1 বাড়ল (a++)। a <= 3 সত্য (a-এর মান 2)। এখন দ্বিতীয় লুপ শুরু হবে। b-এর মান 1। এবারে b <= 3 এবং b != a দুটি শর্তই সত্য। প্রোগ্রামটি দ্বিতীয় লুপের ভেতর ঢুকে যাবে। তৃতীয় লুপের শুরুতে c-এর মান 1। c <=3 সত্য, c !=a সত্য কিন্তু c !=b মিথ্যা (দুটোর মানই 1)। তাই প্রোগ্রামটি তৃতীয় লুপ থেকে বের হয়ে যাবে– কেবল তিনটি শর্ত সত্য হলেই প্রোগ্রামটি তৃতীয় লুপের ভেতর ঢুকবে এবং a, b ও c-এর মান প্রিন্ট করবে। এভাবে কিছুক্ষণ গবেষণা করলে তোমরা দেখবে যে যখন a-এর মান 3, b-এর মান 2 এবং c-এর মান 1, তখনই কেবল সব শর্ত সত্য হয় আর আমরা আউটপুট পাই: 3, 2, 1।

আসলে দ্বিতীয় লুপে আমরা b-এর মান a-এর মানের সমান হলে লুপ থেকে বের হয়ে যাচ্ছি। সেই কাজটি করা ঠিক হচ্ছে না। আমাদের উচিত দুটো মান সমান হলে পরবর্তী মানের জন্য চেষ্টা করা। আর মান দুটো সমান না হলেই কেবল পরবর্তী কাজ করা। তাহলে আমরা লিখতে পারি:

```
for (b = 1; b <= 3; b++) {
    if (b != a) { /* b-এর মান a-এর মানের সমান না হলেই ভেতরের অংশে প্রোগ্রামটি ঢুকবে। */
        for (c = 1; c <= 3; c++) {
            if (c != a && c != b) { /*c-এর মান a-এর মানের সমান না হলে এবং c-এর মান b-এর মানের সমান না হলেই কেবল ভেতরের অংশে প্রোগ্রামটি ঢুকবে। */
                printf ("%d, %d, %d\n", a, b, c);
            }
        }
    }
}
```

তাহলে আমাদের পুরো প্রোগ্রামটি দাঁড়াচ্ছে এই রকম:

```
#include <stdio.h>

int main() {
   int a, b, c;
   for (a = 1; a <= 3; a++) {
      for (b = 1; b <= 3; b++) {
         if (b != a) {
            for (c = 1; c <= 3; c++) {
               if (c != b && c != a){
                  printf ("%d, %d, %d\n", a, b, c);
```

```
                    }
                }
            }
        }
    }

    return 0;
}
```
প্রোগ্রাম: ৪.১৩

প্রোগ্রামটি চালালে আমরা নিচের আউটপুট দেখব, যেটি আমরা চাচ্ছিলাম।

```
1, 2, 3
1, 3, 2
2, 1, 3
2, 3, 1
3, 1, 2
3, 2, 1
```

যাক, অবশেষে আমাদের সমস্যার সমাধান হলো। তবে আমরা কিন্তু আরও সহজেই সমাধান করতে পারতাম এভাবে–

```
#include <stdio.h>

int main()
{
    int a, b, c;
    for (a = 1; a <= 3; a++) {
        for (b = 1; b <= 3; b++) {
```

```
            for (c = 1; c <= 3; c++) {
                if(b != a && c != a && c != b) {
                    printf ("%d, %d, %d\n", a, b, c);
                }
            }
        }
    }
   return 0;
}
```

প্রোগ্রাম: ৪.১৪

এখানে আমাদের বেশি চিন্তা করতে হলো না। কেবল প্রিন্ট করার সময় a, b, c তিনটির মান পরস্পরের সমান নয়, সেটি নিশ্চিত করে নিলেই হলো! বুদ্ধিটা ভালোই, তবে এটির চেয়ে আমাদের আগের প্রোগ্রামটি কম্পিউটারকে দিয়ে কম কাজ করায়, তাই চলতেও কম সময় লাগে, বা কম্পিউটারের ভাষায় বললে রান টাইম (run time) কম। আসলে একটি প্রোগ্রাম চলতে কেমন সময় লাগবে সেটি নির্ভর করে মূলত প্রোগ্রামটি মোট কয়টি অ্যাসাইনমেন্ট অপারেশন আর কয়টি কম্পারিজন অপারেশন করল তার ওপর।

তুমি যদি এই পিডিএফ বইটি কারো কাছ থেকে কপি করে থাকো, কিংবা ইন্টারনেট থেকে ফ্রি ডাউনলোড করে থাকো, তুমি চাইলে লেখকের পরিশ্রমের মূল্য পরিশোধ করতে পারো। এই পিডিএফ বইটির দাম একেবারেই হাতের নাগালে। মাত্র **25** টাকা বিকাশ-এর মাধ্যমে পাঠিয়ে দাও এই নাম্বারেঃ **01622624182**, তোমরা যত বেশি বই কিনবে, সেটি হার্ডকপিই হোক, কিংবা ইবুক (যেমন এই পিডিএফ), লেখকরা তত বেশি উৎসাহ পাবেন এবং আরো বই লেখার ব্যাপারে আগ্রহী হবেন।

বিকাশের মাধ্যমে পেমেন্ট করার নিয়মঃ

01. Go to your bKash Mobile Menu by dialing *247#
02. Choose “Payment”
03. Enter the Merchant bKash Account Number you want to pay to (01622624182)
04. Enter the amount you want to pay (25)
05. Enter a reference against your payment (cpbook pdf)
06. Enter the Counter Number (enter 0)
07. Now enter your bKash Mobile Menu PIN to confirm

১) প্রথমে *247# নাম্বারে ডায়াল করে বিকাশ মেনু আনতে হবে।
২) তারপরে 'পেমেন্ট' নির্বাচন করতে হবে।
৩) এবারে মার্চেন্ট বিকাশ একাউন্ট নাম্বার হিসেবে 01622624182 দিতে হবে।
৪) তারপরে টাকার পরিমাণ দিতে হবে (এই পিডিএফ বইয়ের জন্য ২৫ টাকা)
৫) রেফারেন্স হিসেবে cpbook pdf লিখে দিতে হবে।
৬) কাউন্টার নাম্বার ০ দিবে।
৭) এবারে তোমার বিকাশ মোবাইল মেনু পিন (PIN) দিতে হবে।

তোমার বিকাশ একাউন্ট না থাকলে একটি খুলে নিতে পারো (এটি খুলতে কোনো টাকা লাগে না) কিংবা অন্য কারো বিকাশ একাউন্ট ব্যবহার করতে পারো।

অধ্যায় পাঁচঃ একটুখানি গণিত

এই অধ্যায়ে আমরা প্রোগ্রামিংয়ের নতুন কিছু শিখব না। এখন পর্যন্ত আমরা যতটুকু প্রোগ্রামিং শিখেছি, তা দিয়েই কিছু সহজ-সরল গাণিতিক সমস্যার সমাধান করব।

১) $x + y = 15$, $x - y = 5$ হলে x ও y-এর মান কত?

সমীকরণদুটি যোগ করলে পাই $2x = 20$, বা $x = 10$। আবার বিয়োগ করলে পাই, $2y = 10$, বা $y = 5$। এখন একটি প্রোগ্রাম লিখতে হবে যেখানে $x + y$ ও $x - y$-এর মান দেওয়া থাকবে, x ও y-এর মান বের করতে হবে। আমি প্রোগ্রামটি একটু পরে লিখে দেব। এর মধ্যে তুমি নিজে লেখার চেষ্টা করো। সহজ প্রোগ্রাম।

২) $4x + 5y = 14$, $5x + 6y = 17$ হলে x ও y-এর মান কত?

সমীকরণদুটিকে আমরা এভাবে লিখতে পারি:
$a_1x + b_1y = c_1$, $a_2x + b_2y = c_2$। তোমরা বিভিন্নভাবে এর সমাধান করতে পার। এর মধ্যে দুটি জনপ্রিয় উপায় হচ্ছে প্রতিস্থাপন (substitution) ও নির্ণায়কের (determinant) সাহায্যে সমাধান। পদ্ধতিগুলো জানা না থাকলে ক্লাস এইট বা নাইনের গণিত বই দেখো। সমাধান করলে দেখবে,
$x = (b_2c_1 - b_1c_2) / (a_1b_2 - a_2b_1)$ এবং $y = (a_1c_2 - a_2c_1) / (a_1b_2 - a_2b_1)$। এখন a1, a2, b1, b2, c1, c2-এর জায়গায় নির্দিষ্ট মান বসিয়ে দিলেই x ও y-এর মান পেয়ে যাবে।

এই ধরনের সমীকরণ সমাধানের জন্যও আমরা একটি প্রোগ্রাম লিখব, যার ইনপুট হবে a1, a2, b1, b2, c1, c2 এবং আউটপুট হবে x ও y-এর মান। এটিও সহজ প্রোগ্রাম। নিজে চেষ্টা করো।

আশা করি, তোমরা দুটি সমস্যারই সমাধান নিজে করে ফেলতে পারবে। এখন আমি প্রথম সমস্যার কোড দিচ্ছি:

```
#include <stdio.h>

int main()
{
    double x, y, x_plus_y, x_minus_y;

    printf("Enter the value of x + y: ");
    scanf("%lf", &x_plus_y);

    printf("Enter the value of x - y: ");
    scanf("%lf", &x_minus_y);

    x = (x_plus_y + x_minus_y) / 2;
    y = (x_plus_y - x_minus_y) / 2;

    printf("x = %0.2lf, y = %0.2lf\n", x, y);

    return 0;
}
```
প্রোগ্রাম: ৫.১

সমাধান খুবই সহজ। তবে লক্ষ করো যে আমি ভেরিয়েবলের ডাটা টাইপ int ব্যবহার না করে double ব্যবহার করেছি।

এবারে দ্বিতীয় সমস্যার কোড:

```c
#include <stdio.h>

int main()
{
    double a1, a2, b1, b2, c1, c2, x, y;

    printf("a1 = ");
    scanf("%lf", &a1);
    printf("a2 = ");
    scanf("%lf", &a2);
    printf("b1 = ");
    scanf("%lf", &b1);
    printf("b2 = ");
    scanf("%lf", &b2);
    printf("c1 = ");
    scanf("%lf", &c1);
    printf("c2 = ");
    scanf("%lf", &c2);

    x = (b2 * c1 - b1 * c2) / (a1 * b2 - a2 * b1);
    y = (a1 * c2 - a2 * c1) / (a1 * b2 - a2 * b1);

    printf("x = %0.2lf, y = %0.2lf\n", x, y);

    return 0;
}
```
প্রোগ্রাম: ৫.২

এটিও সহজ প্রোগ্রাম! তবে তোমরা দেখো (a1 * b2 - a2 * b1)-এর মান আমি দুবার বের করেছি (x-এর মান বের করার সময়, আবার y-এর মান বের করার সময়)। কাজটি একবারেই করা যেত এবং একবারে করলেই ভালো, তাহলে আমাদের প্রোগ্রাম দুটি গুণ ও একটি বিয়োগের কাজ কম করবে। আবার (a1 * b2 - a2 * b1)-এর মান যদি শূন্য হয়, তাহলে একটি ঝামেলা হয়ে যাচ্ছে, কারণ কোনো কিছুকে তো শূন্য দিয়ে ভাগ করা যায় না। তাই ওই মানটি শূন্য হলে আসলে সমীকরণের কোনো সমাধান নেই। এবার প্রোগ্রামটি আরও ভালোভাবে লিখে ফেলি।

```
#include <stdio.h>

int main()
{
    double a1, a2, b1, b2, c1, c2, d, x, y;

    printf("a1 = ");
    scanf("%lf", &a1);
    printf("a2 = ");
    scanf("%lf", &a2);
    printf("b1 = ");
    scanf("%lf", &b1);
    printf("b2 = ");
    scanf("%lf", &b2);
    printf("c1 = ");
    scanf("%lf", &c1);
    printf("c2 = ");
    scanf("%lf", &c2);

    d = a1 * b2 - a2 * b1;
```

```
    if ((int) d == 0) {
        printf("Value of x and y can not be determined.\n");
    }
    else {
        x = (b2 * c1 - b1 * c2) / d;
        y = (a1 * c2 - a2 * c1) / d;

        printf("x = %0.2lf, y = %0.2lf\n", x, y);
    }

    return 0;
}
```
প্রোগ্রাম: ৫.৩

এখানে একটি ব্যাপার খেয়াল করো। আমি if-এর ভেতর লিখেছি `(int) d == 0`। এখানে আমি প্রথমে d (যা একটি double টাইপের ভেরিয়েবল)-কে ইন্টিজারে টাইপ কাস্ট করে তারপর তার মানটি 0-এর সমান কি না তা পরীক্ষা করেছি। পরীক্ষাটা এভাবেও করা যেত: if (d == 0.0) তবে এতে মাঝে মাঝে ঝামেলা হয়, ফ্লোটিং পয়েন্ট-সংক্রান্ত হিসাব-নিকাশের জন্য। তোমরা কম্পিউটার আর্কিটেকচার নিয়ে লেখাপড়া করলে বিষয়টা বুঝতে পারবে।

তোমাদের মনে একটি প্রশ্ন আসতে পারে যে এই সহজ সমস্যাগুলো প্রোগ্রামিং করে সমাধান করে কী লাভ? আসলে একবার প্রোগ্রাম লিখে ফেলার পরে কিন্তু আর সমাধান করতে হয় না। তারপর শুধু ইনপুট দেবে, প্রোগ্রামটি নিজেই সমস্যার সমাধান করে তোমাকে আউটপুট দেবে।

৩) আমি যদি তোমাকে দশ হাজার টাকা ঋণ দিই 35% সুদে এবং টাকাটা পাঁচ বছর সময়ের মধ্যে তোমাকে সুদে-আসলে পরিশোধ করতে বলি, তাহলে পাঁচ বছরে মোট কত টাকা তোমার দিতে হবে এবং প্রতি মাসে কত টাকা দিতে হবে? ঋণটা যদি জটিল কিছু না হয়, তাহলে তোমার মোট পরিশোধ করতে হবে 10000 + 10000 * 35 / 100 টাকা। এই সহজ-সরল ঋণের জন্য একটি প্রোগ্রাম

লিখে ফেলা যাক:

```
#include <stdio.h>

int main()
{
    double loan_amount, interest_rate, number_of_years, total_amount,
monthly_amount;

    printf("Enter the loan amount: ");
    scanf("%lf", &loan_amount);
    printf("Enter the interest rate: ");
    scanf("%lf", &interest_rate);
    printf("Number of years: ");
    scanf("%lf", &number_of_years);

    total_amount = loan_amount + loan_amount * interest_rate / 100.00;
    monthly_amount = total_amount / (number_of_years * 12);

    printf("Total amount: %0.2lf\n", total_amount);
    printf("Monthly amount: %0.2lf\n", monthly_amount);

    return 0;
}
```
প্রোগ্রাম: ৫.৪

আমাদের ফর্মুলাতে একটু সমস্যা আছে। আসলে 35% সুদ দিতে হলে সেটা বাৎসরিক সুদ হবে। অর্থাৎ প্রতি বছর মোট ঋণের উপর

35% সুদ দেওয়া লাগবে। তাহলে দেখা যাচ্ছে পাঁচ বছরে তোমার মোট পরিশোধ করতে হবে 10000 + 10000 * 35 * 5 / 100 টাকা। এখন এই ফর্মুলা অনুযায়ী প্রোগ্রাম লিখে ফেলো।

তবে বাস্তবে ঋণের হিসাব-নিকাশ কিন্তু এত সরল নয়। তুমি ব্যাংক থেকে ঋণ নিতে গেলেই সেটি টের পাবে।

৪) পদার্থবিজ্ঞানের একটি সমস্যার সমাধান করা যাক।

কোনো বস্তু u আদিবেগে (initial velocity) এবং a ত্বরণে (acceleration) যাত্রা শুরু করল (ত্বরণের মান সব সময় a থাকবে, বাড়বে বা কমবে না)। t সময় পরে এর বেগ যদি v হয় তাহলে 2t সময়ে বস্তুটি কত দূরত্ব অতিক্রম করবে? (সমস্যাটি দিয়েছেন শাহরিয়ার মঞ্জুর, এটি ভ্যালাডলিড অনলাইন জাজের 10071 নম্বর সমস্যা)।

2t সময়ে অতিক্রান্ত দূরত্ব হবে v x 2t। এটি প্রমাণ করে ফেলো। তারপর আবার পড়া শুরু করো।

নবম-দশম শ্রেণীর পদার্থবিজ্ঞান বইতে তোমরা দুটি সূত্র পাবে:
$v = u + at$
$s = ut + 0.5\ at^2$ (এখানে s হচ্ছে t সময়ে অতিক্রান্ত দূরত্ব)।
তাহলে 2t সময় পরে অতিক্রান্ত দূরত্ব হবে
$u \times 2t + 0.5 \times a \times (2t)^2 = u \times 2t + 0.5 \times a \times 4t^2 = u \times 2t + a \times 2t^2 = 2t\ (u + at) = 2tv$
এখন, তোমাদেরকে একটি প্রোগ্রাম লিখতে হবে, যেখানে v ও t-এর মান ইনপুট হিসেবে দেওয়া হবে, 2t সময়ে অতিক্রান্ত দূরত্ব নির্ণয় করতে হবে। প্রোগ্রামটি নিজে নিজে লিখে ফেলো।

৫) 1 + 2 + 3 + … + 998 + 999 + 1000 এই ধারার সমষ্টি কত?

তোমরা যারা ধারার যোগফলের সূত্র জানো, তারা চট করে বলে দিতে পারবে, এই ধারাটির যোগফল হচ্ছে 1000 x 1001 / 2। তাহলে এর জন্য একটি প্রোগ্রাম লিখে ফেলা যাক, যেখানে শেষ পদের মান হবে ইনপুট আর আউটপুট হবে যোগফল।

```
#include <stdio.h>

int main()
{
    int n, sum;

    scanf("%d", &n);

    sum = (n * (n + 1)) / 2;

    printf("Summation is %d\n", sum);

    return 0;
}
```
প্রোগ্রাম: ৫.৫

ধারার যোগফল নির্ণয়ের সূত্র জানা না থাকলে আমরা লুপ ব্যবহার করে প্রোগ্রামটি লিখতে পারি।

```
#include <stdio.h>

int main()
{
    int i, n, sum;

    scanf("%d", &n);

    for(i = 1, sum = 0; i <= n; i++) {
        sum = sum + i;
```

```
    }

    printf("Summation is %d\n", sum);

    return 0;
}
```
প্রোগ্রাম: ৫.৬

সুতরাং ধারার সমস্যা নিয়ে আর চিন্তা নেই। তুমি যদি একটি পদের মান তার আগের পদের চেয়ে কত করে বাড়ছে, সেটি বের করতে পারো, তাহলেই লুপ ব্যবহার করে যোগফল বের করে ফেলতে পারবে। তবে সূত্র বের করতে পারলে লুপ ব্যবহার না করাই ভালো। কারণ প্রথম প্রোগ্রামটি দেখো (যেখানে সূত্র ব্যবহার করেছি)। সেখানে একটি যোগ, একটি গুণ আর একটি ভাগ করতে হয়েছে, n-এর মান যত বড়ই হোক না কেন। আর দ্বিতীয় প্রোগ্রামে (যেখানে লুপ ব্যবহার করেছি) n-এর মান যত, ততবার যোগ করতে হয়েছে, আবার সেই যোগফলটি sum ভেরিয়েবলে রাখতে হয়েছে (ভেরিয়েবলে কোনো মান রাখতেও কিন্তু একটু সময় লাগে)।
এখন তোমাদের একটি সহজ প্রোগ্রাম লিখতে হবে। প্রথম n সংখ্যক ধনাত্মক বেজোড় সংখ্যার যোগফল নির্ণয়ের প্রোগ্রাম। n-এর মান হবে ইনপুট, আর যোগফল হবে আউটপুট।

৬) আমাদের এবারকার প্রোগ্রামটি হবে তাপমাত্রাকে সেলসিয়াস (Celsius) থেকে ফারেনহাইটে (Farenheit) রূপান্তর করার প্রোগ্রাম।

সেলসিয়াসকে ফারেনহাইটে রূপান্তরের সূত্র হচ্ছে: $°F = (°C \times 1.8) + 32$।

```
#include <stdio.h>

int main()
{
    double celsius, farenheit;
```

```
    printf("Enter the temperature in celsius: ");
    scanf("%lf", &celsius);

    farenheit = 1.8 * celsius + 32;

    printf("Temperature in farenheit is: %lf\n", farenheit);

    return 0;
}
```

প্রোগ্রাম: ৫.৭

এখন তোমাদের কাজ হচ্ছে ফারেনহাইট থেকে সেলসিয়াসে রূপান্তরের প্রোগ্রাম লেখা।

৭) এখন আমরা দুটি সংখ্যার গসাগু (GCD → Greatest Common Divisor বা HCF → Highest Common Factor) ও লসাগু (LCM → Least Common Multiple) নির্ণয় করার জন্য প্রোগ্রাম লিখব।

দুটি সংখ্যার গসাগু হচ্ছে যেসব সংখ্যা দিয়ে ওই দুটি সংখ্যা নিঃশেষে বিভাজ্য হয়, তাদের মধ্যে সবচেয়ে বড় সংখ্যা। তাহলে আমরা যেটি করব, দুটি সংখ্যা a ও b নেব। তারপর এদের মধ্যে যেটি ছোট, সেই মানটি আবার x ভেরিয়েবলে রাখব। গসাগু এর মান x-এর চেয়ে বড় হওয়া সম্ভব নয় (5 ও 10-এর গসাগু-এর মান নিশ্চয়ই 5-এর চেয়ে বড় হবে না)। এখন a ও b, x দিয়ে নিঃশেষে বিভাজ্য হয় কি না (a % x == 0 এবং b % x == 0) সেটি পরীক্ষা করব। যদি হয় তবে আমরা গসাগু পেয়ে গেছি। যদি a ও b উভয়েই নিঃশেষে বিভাজ্য না হয়, তখন x-এর মান এক কমিয়ে পরীক্ষা করব। যতক্ষণ না আমরা গসাগু পাচ্ছি x-এর মান কমাতেই থাকব। একসময় আমরা গসাগু পাবই, কারণ x-এর মান যখন 1 হবে, তখন তো x দিয়ে a ও b দুটি সংখ্যাই নিঃশেষে বিভাজ্য। তোমরা কি প্রোগ্রামটি নিজে লেখার চেষ্টা করবে? না পারলে আমার কোড দেখো:

```
#include <stdio.h>
```

```
int main()
{
    int a, b, x, gcd;

    scanf("%d %d", &a, &b);

    if (a < b) {
        x = a;
    }
    else {
        x = b;
    }

    for(; x >= 1; x--) {
        if (a % x == 0 && b % x == 0) {
            gcd = x;
            break;
        }
    }

    printf("GCD is %d\n", gcd);

    return 0;
}
```
প্রোগ্রাম: ৫.৮

প্রোগ্রামে দেখো gcd পাওয়ার সঙ্গে সঙ্গে লুপ থেকে বের হয়ে যেতে হবে (আমি break ব্যবহার করেছি এই জন্য)। break ব্যবহার না

করলে কী হবে সেটি পরীক্ষা করে দেখো।

তবে গসাগু বের করার জন্য আমি যেই পদ্ধতি ব্যবহার করেছি সেটি খুব সহজ পদ্ধতি হলেও ইফিশিয়েন্ট (efficient) নয়। যেমন, সংখ্যা দুটি খুব বড় হলে এবং সহমৌলিক (co-prime) হলে লুপটি কিন্তু অনেকবার ঘুরবে। কারণ সহমৌলিক হলে গসাগু হবে 1। তোমরা নিশ্চয়ই জানো যে, দুটি সংখ্যার মধ্যে 1 ছাড়া আর কোনো সাধারণ উৎপাদক না থাকলে সংখ্যা দুটি সহমৌলিক।

গসাগু বের করার জন্য ইউক্লিডের একটি চমৎকার পদ্ধতি আছে। ইউক্লিড ভাগশেষ উপপাদ্যের (division algorithm) সাহায্যে গসাগু বের করার উপায় দেখিয়েছেন। এই পদ্ধতিতে খুব সহজে গসাগু বের করা যায় এবং প্রোগ্রামটিও বেশ ইফিসিয়েন্ট হয়। এর জন্য দুটি জিনিস জানা লাগবে:
a ও 0-এর গসাগু-এর মান a।
a ও b-এর গসাগু = b ও a % b-এর গসাগু।

তাহলে প্রোগ্রামে যেটি করতে হবে, একটি লুপের সাহায্যে a-এর মান b আর b-এর মান a%b বসিয়ে যেতে হবে, যতক্ষণ না b-এর মান শূন্য হয়। b-এর মান শূন্য হলেই বুঝে যাব যে গসাগু হচ্ছে a (এটা কিন্তু প্রোগ্রাম শুরুর সময় a-এর মান না, b-এর মান যখন শূন্য হবে সেই সময় a-এর মান)।

```
#include <stdio.h>

int main()
{
    int a, b, t, x, gcd;

    scanf("%d %d", &a, &b);

    if (a == 0) gcd = a;
```

```
    else if (b == 0) gcd = b;
    else {
        while (b != 0) {
            t = b;
            b = a % b;
            a = t;
        }
        gcd = a;
    }

    printf("GCD is %d\n", gcd);

    return 0;
}
```
প্রোগ্রাম: ৫.৯

এই প্রোগ্রামটি আরও ইফিশিয়েন্ট করার চেষ্টা করো।

এবার লসাগু বের করার প্রোগ্রাম। তোমরা নিশ্চয়ই স্কুলে শিখেছ, কীভাবে লসাগু বের করতে হয়। সেই পদ্ধতি অবলম্বন করে প্রোগ্রাম লিখে ফেলো। আর যারা সেই পদ্ধতি জানো না, তাদের জন্য একটি সূত্র বলে দিচ্ছি।

দুটি সংখ্যার লসাগু x দুটি সংখ্যার গসাগু = সংখ্যা দুটির গুণফল।

আশা করি, লসাগু বের করার প্রোগ্রাম লিখতে আর সমস্যা হবে না।

অধ্যায় ছয়ঃ অ্যারে

এতক্ষণে তোমাদের প্রোগ্রামিং জ্ঞান-বুদ্ধি একটু বেড়েছে। চলো, এবার তাহলে কিছু জনসেবামূলক কর্মকাণ্ড করা যাক। আমরা স্কুলের প্রিয় গণিত শিক্ষকের জন্য পরীক্ষার ফলাফল বের করার প্রোগ্রাম লিখে দেব। ওই স্কুলে প্রথম সাময়িক, দ্বিতীয় সাময়িক ও বার্ষিক এই তিনটি পরীক্ষাই 100 নম্বরের হয়। তারপর বার্ষিক পরীক্ষার 50%, দ্বিতীয় সাময়িক পরীক্ষার 25% ও প্রথম সাময়িক পরীক্ষার 25% নিয়ে চূড়ান্ত ফলাফল প্রকাশ করা হয়। তাহলে আমাদের প্রোগ্রামের ইনপুট হচ্ছে ওই তিনটি পরীক্ষার নম্বর। আমাদেরকে চূড়ান্ত ফলাফল দেখাতে হবে। এটি কোনো ব্যাপারই নয়:

```
#include <stdio.h>

int main()
{
    int ft_marks, st_marks, final_marks;
    double total_marks;

    ft_marks = 80;
    st_marks = 74;
    final_marks = 97;

    total_marks = ft_marks / 4.0 + st_marks / 4.0 + final_marks / 2.0;
    printf("%0.0lf\n", total_marks);

    return 0;
}
```

প্রোগ্রাম: ৬.১

প্রোগ্রামটির আউটপুট 87। (কিন্তু আমি যদি total_marks = ft_marks / 4.0 + st_marks / 4.0 + final_marks / 2.0; না

লিখে এভাবে লিখতাম total_marks = ft_marks / 4 + st_marks / 4 + final_marks / 2; তাহলে আউটপুট আসে 86। কারণ কী? কম্পিউটারের মাথা খারাপ নাকি আমার?)

আমরা কিন্তু আমাদের প্রিয় শিক্ষকের তেমন কোনো উপকার করতে পারলাম না। কারণ তাঁর ক্লাসে মোট ছাত্রছাত্রীর সংখ্যা চল্লিশ। তাহলে স্যারকে চল্লিশবার প্রোগ্রামটি চালাতে হবে! কিন্তু এটি তো কোনো কাজের কথা হলো না। আমাদের উচিত, সবার চূড়ান্ত ফলাফল একটি প্রোগ্রামের মাধ্যমে নির্ণয় করা। তেমন কোনো কঠিন কাজ নয় এটি। আমরা এমন একটি প্রোগ্রাম লেখা শুরু করে দিতে পারি:

```
#include<stdio.h>

int main()
{
       int  ft_marks_1,  st_marks_1,  final_marks_1,  ft_marks_2,  st_marks_2,
final_marks_2, ft_marks_3, st_marks_3, final_marks_3,
```

তোমরা নিশ্চয়ই বুঝতে পারছ, আমি কী করতে যাচ্ছি? বলো তো এভাবে প্রোগ্রামটি লিখতে গেলে মোট কয়টি ভেরিয়েবলের দরকার? 160 টি। স্যারের কষ্ট কমাতে গিয়ে আমাদের কষ্ট এত বাড়ানোর কোনো মানে হয় না। কিন্তু এধরনের প্রোগ্রাম তো আমাদের প্রায়ই লিখতে হবে। চিন্তা নেই! প্রায় সব প্রোগ্রামিং ল্যাংগুয়েজেই অ্যারে (Array) নামে একটি চমৎকার জিনিস আছে। এতে একই ধরনের অনেকগুলো ভেরিয়েবল একসঙ্গে রাখা যায়। ভেরিয়েবলের যেমন নাম রাখি, অ্যারের বেলাতেও তেমন একটি নাম দিতে হয়। C তেও অ্যারে আছে।

ভেরিয়েবলের যেমন একটি ডাটা টাইপ থাকে, অ্যারেরও থাকে। অ্যারেটি যে ডাটা টাইপের হবে তাতে কেবল সেই রকম ডাটাই রাখা যাবে। যেমন char টাইপের অ্যারেতে কেবল char টাইপের জিনিস থাকবে।

অ্যারেতে কয়টি উপাদান থাকবে সেটি শুরুতেই বলে দিতে হয়।
int ara[10]; এভাবে আমরা একটি অ্যারে ডিক্লেয়ার করতে পারি, যার নাম হচ্ছে ara, যেটিতে কেবল ইন্টিজার টাইপের ডাটা থাকবে

আর এই অ্যারেতে মোট দশটি সংখ্যা রাখা যাবে। প্রথমটি হচ্ছে ara[0] (হ্যাঁ, ara[1] না কিন্তু), দ্বিতীয়টি ara[1], তৃতীয়টি ara[2], এভাবে দশম সংখ্যাটি হচ্ছে ara[9]। অর্থাৎ, ara[i] হচ্ছে i+1 তম উপাদান।

এবারে চলো অ্যারে নিয়ে একটু খেলাধুলা করা যাক। প্রতিটি প্রোগ্রাম কিন্তু অবশ্যই কম্পিউটারে চালিয়ে দেখবে।

```
#include <stdio.h>

int main()
{
    int ara[5] = {10, 20, 30, 40, 50};

    printf("First element: %d\n", ara[0]);
    printf("Third element: %d\n", ara[2]);

    return 0;
}
```

প্রোগ্রাম: ৬.২

আউটপুট ঠিকঠাক দেখতে পাচ্ছ?

আরেকটি প্রোগ্রাম:

```
#include <stdio.h>

int main()
{
    int ara[5] = {6, 7, 4, 6, 9};

    printf("%d\n", ara[-1]);
    printf("%d\n", ara[5]);
    printf("%d\n", ara[100]);

    return 0;
}
```
প্রোগ্রাম: ৬.৩

এটির জন্য কী আউটপুট আসা উচিত? আমি জানি না এবং এটি জানা সম্ভব নয়। যেকোনো ধরনের সংখ্যা আসতে পারে। এগুলোকে গারবেজ (garbage) বলে। কারণ আসলে তো ওই অ্যারেতে -1, 5, 100 এই ইনডেক্স বলতে কিছু নেই। অ্যারেটির দৈর্ঘ্যই হচ্ছে 5 সুতরাং ইনডেক্স হবে 0 থেকে 4।

এখন কোনো অ্যারের সব উপাদান যদি একসঙ্গে দেখাতে চাই, তাহলে উপায় কী? উপায় হচ্ছে প্রথম উপাদান (ara[0]), দ্বিতীয় উপাদান (ara[1]), তৃতীয় উপাদান (ara[2]) … এভাবে একে একে সবগুলো প্রিন্ট করা। আর তার জন্য অবশ্যই আমরা লুপের সাহায্য নেব।

```
#include <stdio.h>

int main()
{
    int ara[10] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
```

```
    int i;

    for(i = 0; i < 10; i++) {
        printf("%d th element is: %d\n", i+1, ara[i]);
    }

    return 0;
}
```
প্রোগ্রাম: ৬.৪

আর যদি শেষ উপাদান থেকে প্রথম উপাদান পর্যন্ত দেখাতে হতো? কোনো সমস্যা নেই, শুধু লুপে এ index টি 9 থেকে 0 পর্যন্ত আনলেই চলবে। এখন তোমরা প্রোগ্রামটি লিখে ফেলো।

এবারে একটি ছোট সমস্যা। কোনো একটি অ্যারেতে দশটি উপাদান আছে, সেগুলো বিপরীত ক্রমে রাখতে হবে। অর্থাৎ দশম উপাদানটি হবে প্রথম উপাদান, প্রথমটি হবে দশম, দ্বিতীয়টি হবে নবম, নবমটি হবে দ্বিতীয়.. এই রকম। তার জন্য আমরা যেটি করতে পারি, আরেকটি অ্যারের সাহায্য নিতে পারি। দ্বিতীয় অ্যারেটিতে প্রথম অ্যারের উপাদানগুলো বিপরীত ক্রমে রাখবো। তারপর দ্বিতীয় অ্যারেটি প্রথম অ্যারেতে কপি করে ফেলব।

```
#include <stdio.h>

int main()
{
    int ara[] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
    int ara2[10];
    int i, j;
```

```
    for(i = 0, j = 9; i < 10; i++, j--) {
        ara2[j] = ara[i];
    }

    for(i = 0; i < 10; i++) {
        ara[i] = ara2[i];
    }

    for(i = 0; i < 10; i++) {
        printf("%d\n", ara[i]);
    }

    return 0;
}
```
প্রোগ্রাম: ৬.৫

এখানে লক্ষ করো যে প্রথম অ্যারেটির ক্ষেত্রে আমি তৃতীয় বন্ধনীর ভেতর অ্যারের উপাদান সংখ্যা বলে দিইনি, কারণ সি-এর কম্পাইলার দ্বিতীয় বন্ধনীর ভেতর সংখ্যাগুলো দেখেই বুঝে নিতে পারে যে ara তে দশটি উপাদান আছে। দ্বিতীয় অ্যারে অর্থাৎ ara2 তে এখন কোনো কিছু নেই। তাই শুরুতেই বলে দিতে হবে যে তাতে কয়টি উপাদান থাকবে। তাহলে কম্পাইলার সেই অনুসারে কম্পিউটারের মেমোরির মধ্যে অ্যারের জন্য জায়গা করে নেবে।

প্রোগ্রামটি ভালোভাবেই কাজ করছে। কিন্তু তোমরা একটু চিন্তাভাবনা করলেই বুঝতে পারবে যে দ্বিতীয় অ্যারেটি ব্যবহার করার কোনো দরকার ছিল না। আমরা একটি বহুল প্রচলিত পদ্ধতিতেই কাজটি করতে পারতাম।

```
int temp;
temp = ara[9];
ara[9] = ara[0];
```

ara[0] = temp;
প্রথম ও দশম উপাদান অদলবদল হয়ে গেল। তারপর
temp = ara[8];
ara[8] = ara[1];
ara[1] = temp;
দ্বিতীয় ও নবম উপাদান অদলবদল হয়ে গেল। তাহলে চলো প্রোগ্রামটি লিখে ফেলি:

```
#include <stdio.h>

int main()
{
    int ara[] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
    int i, j, temp;

    for(i = 0, j = 9; i < 10; i++, j--) {
        temp = ara[j];
        ara[j] = ara[i];
        ara[i] = temp;
    }

    for(i = 0; i < 10; i++) {
        printf("%d\n", ara[i]);
    }

    return 0;
}
```
প্রোগ্রাম: ৬.৬

প্রোগ্রামটি চালাও। কী দেখলে? আউটপুট কি এরকম?

```
10
20
30
40
50
60
70
80
90
100
```

তারমানে কাজ হয়নি! আসলে আমি একটি ছোট্ট ভুল করেছি, সেটি তোমরা খুঁজে বের করো। এ ধরনের ভুলকে বলে বাগ (bug), তখন প্রোগ্রাম ঠিকমতো রান করে কিন্তু সঠিক আউটপুট দেয় না। আমার কোডে বাগ আছে, তোমরা ডিবাগ (debug) করো (মানে বাগটি বের করে ঠিক করো)।

এখন চলো আমাদের আগের সমস্যায় ফিরে যাই। আমরা এখন প্রথম সাময়িক পরীক্ষায় সবার গণিতের নম্বর একটি অ্যারেতে রাখব, দ্বিতীয় সাময়িক পরীক্ষার নম্বর আরেকটি অ্যারেতে, বার্ষিক পরীক্ষার নম্বরের জন্য আরও একটি এবং রেজাল্টের জন্যও একটি অ্যারে ব্যবহার করব।

```
int ft_marks[40], st_marks[40], final_marks[40];
double total_marks[40];
```

যার রোল নম্বর 1 তার নম্বরগুলো থাকবে অ্যারের প্রথম ঘরে (মানে index 0 হবে)। এখন বলো তো total_marks[34]-এ কার সর্বমোট নম্বর আছে? যার রোল নম্বর 35। তাহলে কারও রোল নম্বর n হলে তার সর্বমোট নম্বর হচ্ছে total_marks[n-1]।

এখন প্রোগ্রামটি লিখে ফেলা যাক:

```
#include <stdio.h>

int main()
{
    int ft_marks[40] = {83, 86, 97, 95, 93, 95, 86, 52, 49, 41, 42, 47, 90,
59, 63, 86, 40, 46, 92, 56, 51, 48, 67, 49, 42, 90, 42, 83, 47, 95, 69, 82,
82, 58, 69, 67, 53, 56, 71, 62},
    st_marks[40] = {86, 97, 95, 93, 95, 86, 52, 49, 41, 42, 47, 90, 59, 63,
86, 40, 46, 92, 56, 51, 48, 67, 49, 42, 90, 42, 83, 47, 95, 69, 82, 82, 58,
69, 67, 53, 56, 71, 62, 49},
    final_marks[40] = {87, 64, 91, 43, 89, 66, 58, 73, 99, 81, 100, 64, 55,
69, 85, 81, 80, 67, 88, 71, 62, 78, 58, 66, 98, 75, 86, 90, 80, 85, 100, 64,
55, 69, 85, 81, 80, 67, 88, 71};

    int i;
    double total_marks[40];

    for(i = 0; i < 40; i++) {
        total_marks[i] = ft_marks[i] / 4.0 + st_marks[i] / 4.0 +
final_marks[i] / 2.0;
    }

    for(i = 1; i <= 40; i++) {
        printf("Roll NO: %d\tTotal Marks: %0.0lf\n", i, total_marks[i-1]);
    }
```

```
    return 0;
}
```
প্রোগ্রাম: ৬.৭

রান করে দেখো, কী সুন্দর আউটপুট! printf ফাংশনের ভেতরে দেখো এক জায়গায় আমি \t লিখেছি, এতে ট্যাব (Tab) প্রিন্ট হবে (কিবোর্ডের বাঁ দিকে দেখো)। রোল নং প্রিন্ট করার পরে একটি ট্যাব দিয়ে টোটাল মার্কস প্রিন্ট করলে দেখতে একটু ভালো লাগে এই জন্য \t ব্যবহার করেছি, এমনিতে কোনো দরকার নেই।

কিন্তু এত সুন্দর প্রোগ্রাম দেখে তোমার শিক্ষক কোথায় তোমাকে একটু চটপটি খাওয়াবেন না উল্টা আরেকটি আবদার করে বসলেন। কোন নম্বর কতজন পেয়েছে সেটি উনি দেখতে চান। মানে 50 কতজন পেল, 51 কতজন পেল ... এই রকম আর কি। বাকি অংশ পড়ার আগে প্রোগ্রামটি তোমরা নিজে নিজে লেখার চেষ্টা করো। এখন ইচ্ছা না করলে বইটি পড়া বন্ধ করে দাও এবং পরে কোনো একসময় চেষ্টা করবে।

আশা করি, তোমাদের মধ্যে কেউ কেউ প্রোগ্রামটি লিখে ফেলেছ। যদি কমপক্ষে এক ঘণ্টা চেষ্টার পরেও লিখতে না পারো তাহলে এখন আমরা সমাধানের চেষ্টা করতে পারি। শুরুতেই একটি ব্যাপার খেয়াল করো যে কেউ কিন্তু 50-এর নিচে নম্বর পায়নি। তাই 50 থেকে 100 পর্যন্ত কোন নম্বর কতজন পেল সেটি বের করলেই চলবে। আমার মাথায় প্রথমেই যে সমাধান আসছে সেটি হলো total_marks অ্যারেতে প্রথমে দেখব, কয়টি 50 আছে, তারপর আবার দেখব কয়টি 51 আছে ... এভাবে 100 পর্যন্ত দেখব। মানে 50 থেকে 100 পর্যন্ত সব সংখ্যার জন্য total_marks অ্যারেতে সংখ্যাগুলো চেক করব।

for(marks = 50; marks <= 100; marks++) { লুপের সাহায্যে প্রথমে marks-এর মান 50, তারপরে 51, এভাবে এক এক করে বাড়াব 100 পর্যন্ত।

count = 0; ধরে নিচ্ছি শূন্য জন 'marks' নম্বর পেয়েছে। marks-এর সব কটি মানের জন্যই প্রথমে আমরা এই কাজটি করব।

এবারে total_marks অ্যারেতে দেখব যে কোনো নম্বর যদি marks-এর সমান হয়, তবে count-এর মান এক বাড়িয়ে দেব। তাহলে কোনো একটি নম্বর (marks) যতবার অ্যারেতে আছে, count-এর মান তত হবে।

```
for(i = 0; i < 40; i++) {
    if(total_marks[i] == marks) {
```

count++;
}
}
printf("Marks: %d Count: %d\n", marks, count); এখানে আমরা প্রতিটি marks এবং সেটি কতবার আছে (count) তা প্রিন্ট করে দিচ্ছি।
}

তাহলে পুরো প্রোগ্রাম লিখে ফেলি:

```
#include <stdio.h>

int main()
{
    int marks, i, count;
    int total_marks[] = {86, 78, 94, 68, 92, 78, 64, 62, 72, 61, 72, 66, 65, 65, 80, 72, 62, 68, 81, 62, 56, 68, 58, 56, 82, 70, 74, 78, 76, 84, 88, 73, 62, 66, 76, 70, 67, 65, 77, 63};

    for(marks = 50; marks <= 100; marks++) {
        count = 0;
        for(i = 0; i < 40; i++) {
            if(total_marks[i] == marks) {
                count++;
            }
        }
        printf("Marks: %d  Count: %d\n", marks, count);
    }
```

```
    return 0;
}
```
প্রোগ্রাম: ৬.৮

তেমন কঠিন কিছু নয়। নেস্টেড ফর লুপ ব্যবহার করে সহজ-সরল সমাধান করে ফেললাম। আচ্ছা বলো তো if-এর ভেতর যে শর্তটি আমরা পরীক্ষা করছি (total_marks[i] == marks) এই কাজটি প্রোগ্রামে কতবার হয়? বাইরের লুপটি ঘুরবে 51 বার এবং প্রতিবারের জন্য ভেতরের লুপটি ঘুরবে 40 বার। তাহলে মোট 51 x 40 = 2040 বার।

ওপরের প্রোগ্রামটি আমরা এখন একটু অন্যভাবে লেখার চেষ্টা করব। নিচের প্রোগ্রামটি চটপট টাইপ করে ফেলো এবং রান করো:

```
#include <stdio.h>

int main()
{
    int i;
    int total_marks[] = {86, 78, 94, 68, 92, 78, 64, 62, 72, 61, 72, 66, 65,
65, 80, 72, 62, 68, 81, 62, 56, 68, 58, 56, 82, 70, 74, 78, 76, 84, 88, 73,
62, 66, 76, 70, 67, 65, 77, 63};
    int marks_count[101];

    for(i = 0; i < 101; i++) {
        marks_count[i] = 0;
    }

    for(i = 0; i < 40; i++) {
        marks_count[total_marks[i]]++;
    }
```

```
    for(i = 50; i <= 100; i++) {
        printf("Marks: %d  Count: %d\n", i, marks_count[i]);
    }

    return 0;
}
```
প্রোগ্রাম: ৬.৯

এখানে আমি যেটি করেছি, একটি অতিরিক্ত অ্যারে ব্যবহার করেছি। marks_count একটি ইন্টিজার টাইপের অ্যারে এবং marks_count[n] দিয়ে আমরা বুঝব n সংখ্যাটি কতবার total_marks-এর মধ্যে আছে। নম্বর যেহেতু 0 থেকে 100-এর মধ্যে হতে পারে তাই আমরা ওই অ্যারেতে মোট 101 টি সংখ্যা রাখার ব্যবস্থা করলাম।

```
int marks_count[101];
```
শুরুতে যেহেতু কিছুই জানি না, তাই ধরে নিই, সব সংখ্যা শূন্য বার আছে। তাই marks_count অ্যারের সব ঘরে 0 বসিয়ে দিই:

```
for(i = 0; i < 101; i++) {
   marks_count[i] = 0;
}
```

এখন total_marks অ্যারের প্রতিটি সংখ্যার জন্য marks_count অ্যারের ওই ঘরের মান এক বাড়িয়ে দিই।

```
for(i = 0; i < 40; i++) {
   marks_count[total_marks[i]]++;
}
```
বুঝতে সমস্যা হচ্ছে নাকি? একটু চিন্তা করো।

যখন i-এর মান 0, তখন total_marks[i] হচ্ছে total_marks[0], অর্থাৎ 86। এখন আমাদের দরকার হচ্ছে marks_count অ্যারের ওই ঘরটার (মানে marks_count[86]) মান এক বাড়িয়ে দেওয়া। শুরুতে ছিল শূন্য, এখন হবে এক। আমরা কিন্তু সে কাজটিই করেছি marks_count[total_marks[i]]-এর মান এক বাড়িয়ে দিয়েছি marks_count[total_marks[i]]++;

আসলে ব্যাপারটি এইভাবেও লেখা যেত:
t_m = total_marks[i];
marks_count[t_m]++;

এখনো যারা মাথা চুলকাচ্ছ তারা নিচের প্রোগ্রামটি কম্পিউটারে রান করাও। এখানে প্রতিবার marks_count[total_marks[i]]++; করার পরে marks_count অ্যারেটি আমরা এক লাইনে প্রিন্ট করেছি।

```
#include <stdio.h>

int main()
{
    int i, j;
    int total_marks[] = {6, 7, 4, 6, 9, 7, 6, 2, 4, 3, 4, 1};
    int marks_count[11];

    for(i = 0; i < 11; i++) {
        marks_count[i] = 0;
    }

    for(i = 0; i < 12; i++) {
        marks_count[total_marks[i]]++;

        for(j = 0; j <= 10; j++) {
            printf("%d  ", marks_count[j]);
        }
        printf("\n");
    }
```

```
    return 0;
}
```
প্রোগ্রাম: ৬.১০

আশা করি, এখন আর বুঝতে সমস্যা হওয়ার কথা নয়। আর তুমি তোমার প্রোগ্রামের জন্য তোমার গণিত শিক্ষকের কাছ থেকে একদিন চটপটি দাবি করতেই পারো। তবে চটপটি পেতে হলে প্রোগ্রামটি এমনভাবে লিখতে হবে যেন তোমার শিক্ষক সবার পরীক্ষার নম্বর নিজে ইনপুট হিসেবে দিতে পারেন। তুমি কি সেটি করতে পারবে?

তুমি যদি এই পিডিএফ বইটি কারো কাছ থেকে কপি করে থাকো, কিংবা ইন্টারনেট থেকে ফ্রি ডাউনলোড করে থাকো, তুমি চাইলে লেখকের পরিশ্রমের মূল্য পরিশোধ করতে পারো। এই পিডিএফ বইটির দাম একেবারেই হাতের নাগালে। মাত্র **25** টাকা বিকাশ-এর মাধ্যমে পাঠিয়ে দাও এই নাম্বারেঃ **01622624182**, তোমরা যত বেশি বই কিনবে, সেটি হার্ডকপিই হোক, কিংবা ইবুক (যেমন এই পিডিএফ), লেখকরা তত বেশি উৎসাহ পাবেন এবং আরো বই লেখার ব্যাপারে আগ্রহী হবেন।

বিকাশের মাধ্যমে পেমেন্ট করার নিয়মঃ

01. Go to your bKash Mobile Menu by dialing *247#
02. Choose “Payment”
03. Enter the Merchant bKash Account Number you want to pay to (01622624182)
04. Enter the amount you want to pay (25)
05. Enter a reference against your payment (cpbook pdf)
06. Enter the Counter Number (enter 0)
07. Now enter your bKash Mobile Menu PIN to confirm

১) প্রথমে *247# নাম্বারে ডায়াল করে বিকাশ মেনু আনতে হবে।
২) তারপরে 'পেমেন্ট' নির্বাচন করতে হবে।
৩) এবারে মার্চেন্ট বিকাশ একাউন্ট নাম্বার হিসেবে 01622624182 দিতে হবে।
৪) তারপরে টাকার পরিমাণ দিতে হবে (এই পিডিএফ বইয়ের জন্য ২৫ টাকা)
৫) রেফারেন্স হিসেবে cpbook pdf লিখে দিতে হবে।
৬) কাউন্টার নাম্বার ০ দিবে।
৭) এবারে তোমার বিকাশ মোবাইল মেনু পিন (PIN) দিতে হবে।

তোমার বিকাশ একাউন্ট না থাকলে একটি খুলে নিতে পারো (এটি খুলতে কোনো টাকা লাগে না) কিংবা অন্য কারো বিকাশ একাউন্ট ব্যবহার করতে পারো।

অধ্যায় সাতঃ ফাংশন

তোমরা কি একটি মজার ব্যাপার জানো? একজন লেখক সারা জীবনে যতটা সময় লেখেন তার চেয়ে বেশি সময় তিনি অন্যের লেখা পড়েন? ব্যাপারটি প্রোগ্রামারদের বেলাতেও সত্য। একজন প্রোগ্রামার তার প্রোগ্রামিং জীবনে যতটা সময় নিজে কোড লেখে তার চেয়ে বেশি সময় অন্যের লেখা কোড পড়ে! তাই কোড লেখার সময় খেয়াল রাখতে হবে, যেন সেটি পড়াও সুবিধাজনক হয়।

যারা বইটি শুরু থেকে পড়ে এসেছ তারা ইতিমধ্যে অনেকবার ফাংশন শব্দটি দেখেছ। যারা আরও বেশি মনোযোগ দিয়ে পড়েছ তারা এটিও খেয়াল করেছ যে printf, scanf ইত্যাদি, যেগুলো তোমরা ব্যবহার করছ সেগুলো একেকটি ফাংশন। আবার main ও একটি ফাংশন। আমরা এবার দেখব ফাংশন ব্যাপারটি আসলে কী, এর দরকারটাই বা কী। আর তারপর আমরা নিজেদের ফাংশন তৈরি করা শিখব।

ফাংশন ব্যবহার করা হয় কোনো একটি নির্দিষ্ট কাজ করার জন্য। যেমন printf ফাংশনটি দিয়ে আমরা মনিটরে আউটপুট দিই। আবার scanf, getchar এসব ফাংশন দিয়ে আমরা কিবোর্ড থেকে ইনপুট নিই। এখন printf ফাংশনটি যে আমরা লিখলাম, কম্পিউটারের তো আর এটি বোঝার কথা নয়। printf ফাংশনটি কী কাজ করবে, কীভাবে করবে সেটি আসলে বলে দেওয়া আছে stdio.h নামের একটি হেডার (header) ফাইলের মধ্যে। এজন্যই আমরা আমাদের প্রোগ্রামগুলোতে (যেখানে printf, scanf ইত্যাদি ব্যবহার করেছি) ওই হেডার ফাইলটির কথা বলে দিই (#include <stdio.h>)। আবার স্ট্রিং-সংক্রান্ত ফাংশনগুলো ব্যবহার করলে string.h – এই হেডার ফাইলটির কথাও বলে দিই। এখন চিন্তা করো, printf ফাংশনের এই কোডটি যদি আমাদের নিজেদের লিখতে হতো, তাহলে ব্যাপারটি কী বিরক্তিকরই না হতো! এরকম অনেক ফাংশন আছে যেগুলোর ব্যবহার তোমরা আস্তে আস্তে জেনে যাবে।

আচ্ছা, main কে ও তো আমি একটি ফাংশন বলেছি, কিন্তু এটি দিয়ে আমরা আবার কী করি? সি ল্যাঙ্গুয়েজে এই ফাংশনটি দিয়েই আসলে আমরা একটি প্রোগ্রাম চালাই। কম্পাইলার জানে যে main ফাংশন যেখানে আছে, সেখান থেকেই কাজ শুরু করতে হবে। তাই একটি প্রোগ্রামে কেবল একটিই main ফাংশন থাকে।
এবারে দেখি, আমরা নিজেরা কীভাবে ফাংশন তৈরি করতে পারি। একটি ফাংশন যখন আমরা তৈরি করব সেটির গঠন হবে মোটামুটি এই রকম:

```
return_type function_name (parameters) {
    function_body

    return value
}
```

return_type: এখানে বলে দিতে হবে ফাংশনটি কাজ শেষ করে বের হবার সময় কী ধরনের ডাটা রিটার্ন করবে। সেটি, int, double এসব হতে পারে। আবার কিছু রিটার্ন করতে না চাইলে সেটি void হতে পারে। অর্থাৎ সে কিছুই রিটার্ন করবে না। এর মানে দাঁড়াচ্ছে, তুমি আসলে ফাংশনকে দিয়ে কোনো একটি কাজ করাবে, সেজন্য কাজ শেষে সে তোমাকে কী ধরনের ডাটা ফেরত দেবে সেটি বলে দিতে হবে।

function_name: এখানে আমাদের ফাংশনের নাম লিখতে হবে। ফাংশনের নাম হতে হবে অর্থপূর্ণ যাতে নাম দেখেই ধারনা করা যায় যে ফাংশনটি কী কাজ করবে। যেমন কোন সংখ্যার বর্গমূল নির্ণয়ের জন্য যদি আমরা একটি ফাংশন লিখি তবে সেটির নাম আমরা দিতে পারি square_root বা sqrt। আমরা নিশ্চয়ই সেটির নাম beautiful দিব না, যদিও কম্পাইলার তাতে কোন আপত্তি করবে না।

parameters: এখানে ফাংশনটি কাজ করার জন্য প্রয়োজনীয় ডাটা আমরা দেব। যেমন স্ট্রিং-এর দৈর্ঘ্য নির্ণয়ের জন্য আমরা যখন strlen ফাংশনটি ব্যবহার করি সেখানে কোন স্ট্রিং-এর দৈর্ঘ্য নির্ণয় করতে হবে সেটি বলে দিতে হয় (নইলে সেটি কার দৈর্ঘ্য নির্ণয় করবে?)। আবার বর্গমূল নির্ণয়ের জন্য ফাংশন লিখলে কোন সংখ্যার বর্গমূল বের করতে হবে সেটি বলে দিতে হবে। প্যারামিটারের মাধ্যমে আমরা সেসব ডাটা ওই ফাংশনের কাছ পাঠাতে পারি। আবার কোনো কিছু পাঠাতে না চাইলে সেটি খালিও রাখতে পারি। যেমন, getchar() বা main() ফাংশন। একাধিক প্যারামিটার পাঠানোর সময় প্রতিটি প্যারামিটার কমা (,) দিয়ে আলাদা করতে হবে।
function_body: ফাংশনটি কীভাবে কী কাজ করবে সেটি বডিতে বলে দিতে হবে। মানে কোড লিখতে হবে আর কি।
return value: ফাংশনটি কাজ শেষ করে, তাকে যে জায়গা থেকে কল করা হয়েছে সে জায়গায় ফিরে যায়। ফেরার সময় আমরা কোনো মান পাঠাতে পারি। যেমন sqrt() ফাংশনে আমরা চাই সে বর্গমূল বের করবে। তো বর্গমূলটি বের করে তো সেটি ফেরত পাঠাবার ব্যবস্থা রাখতে হবে? বর্গমূলটির মান যদি x হয়, তবে আমরা return x; স্টেটমেন্ট দিয়ে সেটির মান ফেরত পাঠাব।

int root = sqrt(25);
এখানে sqrt ফাংশন 25-এর বর্গমূল নির্ণয় করার পর বর্গমূলটি ফেরত পাঠাবে এবং সেটি root নামের একটি ইন্টিজার ভেরিয়েবলে জমা হবে।

একটি উদাহরণ দিই। তোমরা যারা ত্রিকোণমিতি পড়েছ তারা নিশ্চয়ই sin, cos, tan ইত্যাদির সঙ্গে পরিচিত। sin 30^0-এর মান হচ্ছে 0.5। এখানে sin কিন্তু আসলে একটি ফাংশন, যার প্যারামিটার হিসেবে আমরা কোণের মান দিচ্ছি। আর ফাংশনটি ওই কোণের sine (সংক্ষেপে sin)-এর মান রিটার্ন করছে।

এবারে চলো, আর বকবক না করে প্রোগ্রামিং শুরু করে দিই। তারপর দেখি কী করলে কী হয়।

```
#include <stdio.h>

int main()
{
    double a, b, c;
    a = 2.5;
    b = 2.5;
    c = a + b;
    printf("%lf\n" c);

    return 0;
}
```
প্রোগ্রাম: ৭.১

প্রোগ্রামটি চালাও। আউটপুট কী? 5.000000।

এবার আমরা দুটি সংখ্যা যোগ করার জন্য একটি ফাংশন লিখে ফেলি। যোগের কাজটি আর main ফাংশনের ভেতরে করব না।

```
#include <stdio.h>

int add(int num1, int num2)
{
    double sum = num1 + num2;
    return sum;
}

int main()
{
    double a, b, c;
    a = b = 2.5;
    c = add(a, b);
    printf("%lf\n", c);

    return 0;
}
```

প্রোগ্রাম: ৭.২

প্রোগ্রামটি চালাও। আউটপুট কী? 4.000000! ওহ্ আমরা তো গাধার মতো একটি ভুল করেছি। num1 ও num2 তো আসলে int টাইপের হবে না, double টাইপের হবে। ওই দুটি ভেরিয়েবল ইন্টিজার হিসেবে ডিক্লেয়ার করার কারণে 2.5 হয়ে গিয়েছে 2 (টাইপ কাস্টিংয়ের কথা মনে আছে তো?)। আমরা ভুল ঠিক করে ফেলি:

```
int add(double num1, double num2) {
    double sum = num1 + num2;
```

```
    return sum;
}
```

এবারে প্রোগ্রামটি রান করলে আউটপুট কী? 5.000000। যাক, সমস্যার সমাধান হয়ে গেল! আচ্ছা, এবারে আমরা a, b-এর মান একটু বদলাই। a = 2.8; b = 2.7; করে দিই। আউটপুট কত হবে? 5.500000? এটিই হওয়া উচিত (2.8 + 2.7 = 5.5) কিন্তু প্রোগ্রামটি রান করে দেখো তো কত হয়? তুমি আউটপুট পাবে 5.000000। কারণ কী?

কারণ, আমাদের ফাংশনের রিটার্ন টাইপ int, যা কিনা একটি ইন্টিজার রিটার্ন করতে সক্ষম। num1 ও num2 যোগ করার পর sum-এর মধ্যে 5.5 ঠিকই থাকবে কিন্তু রিটার্ন করার সময় সেটি ইন্টিজারে বদলে যাবে। সুতরাং রিটার্ন টাইপ আমরা double করে দেব। এবার আমাদের প্রোগ্রাম ঠিকঠাক কাজ করবে:

```
#include <stdio.h>

double add(double n1, double n2)
{
    double sum = n1 + n2;
    return sum;
}
int main()
{
    double a, b, c;
    a = 2.8;
    b = 2.7;
    c = add(a, b);
    printf("%lf\n", c);

    return 0;
}
```

প্রোগ্রাম: ৭.৩

এখন আমরা একটি পরীক্ষা করব। add ফাংশনটি main ফাংশনের পরে লিখব:

```
#include <stdio.h>

int main()
{
    double a = 2.8, b = 2.7, c;

    c = add(a, b);
    printf("%lf\n", c);

    return 0;
}

double add(double n1, double n2)
{
    double sum = n1 + n2;
    return sum;
}
```
প্রোগ্রাম: ৭.৪

এবারে প্রোগ্রামটি রান করতে গেলে দেখবে, কম্পাইলার এরর দিচ্ছে: “error: ‘add’ was not declared in this scope”, অর্থাৎ সে আর add ফাংশনটিকে চিনতে পারছে না। তবে চিন্তা নেই, এটিকে চিনিয়ে দেওয়ার ব্যবস্থাও আছে। সেটি হচ্ছে main ফাংশনের আগে add ফাংশনের প্রোটোটাইপ (prototype) বলে দেওয়া:

double add(double n1, double n2);

প্রোটোটাইপে পুরা ফাংশনটি লিখতে হয় না। এর অংশগুলো হচ্ছে:
return_type function_name (parameters) ;

সেমিকোলন দিতে ভুল করবে না কিন্তু। আর প্রোটোটাইপের প্যারামিটারে যে ভেরিয়েবল ব্যবহার করবে তার সঙ্গে মূল ফাংশনের ভেরিয়েবলের নাম একরকম না হলে কোনো অসুবিধা নেই, তবে ডাটা টাইপ একই হতে হবে। এখন নিচের প্রোগ্রামটি ঠিকঠাক কাজ করবে:

```
#include <stdio.h>

double add(double x, double y);

int main()
{
    double a = 2.8, b = 2.7, c;

    c = add(a, b);
    printf("%lf\n", c);

    return 0;
}

double add(double n1, double n2)
{
    double sum = n1 + n2;
    return sum;
}
```
প্রোগ্রাম: ৭.৫

এবার আমরা আরও কিছু পরীক্ষা-নিরীক্ষা করব।

```
#include <stdio.h>

int test_function(int x)
{
    int y = x;
    x = 2 * y;
    return (x * y);
}

int main()
{
    int x = 10, y = 20, z = 30;
    z = test_function(x);
    printf(“%d %d %d\n”, x, y, z);

    return 0;
}
```

প্রোগ্রাম: ৭.৬

প্রোগ্রামটি না চালিয়ে শুধু কোড দেখে বলো তো আউটপুট কী হবে? আমাদের কোনো তাড়া নেই, তাই ধীরেসুস্থে চিন্তা করে বলো। এবার কে কে আমার সঙ্গে একমত যে আউটপুট হবে:
20 10 200 (অর্থাৎ x = 20, y = 10, z = 200)?
কারণ x, y-এর মান তো test_function-এর ভেতরে আমরা বদলে দিয়েছি। প্রথমে x-এর মান 10 যাচ্ছে প্যারামিটার হিসেবে, তারপরে সেই মানটি আমরা y-তে বসাচ্ছি। মানে y-এর মান এখন 10। তারপর x-এর মান বসাচ্ছি 2 * y মানে 20। তারপর রিটার্ন করছি x * y (যার মান, 20 * 10 বা 200)। সুতরাং z-এর মান হবে 200।

এবারে প্রোগ্রামটি চালাও, আউটপুট দেখবে: 10 20 200 (অর্থাৎ x = 10, y = 20, z = 200)। এমন হওয়ার কারণ কী? z-এর মান নিয়ে কোনো আপত্তি নেই, ফাংশনটি 200 রিটার্ন করে আর সেটি আমরা z-এ বসিয়ে দিয়েছি। কথা হচ্ছে, x আর y-এর মান নিয়ে। আসলে test_function-এর ভেতরে আমরা x, y-এর মান পরিবর্তন করায় main ফাংশনের x, y-এর কিছু আসে-যায় না। প্রত্যেক ফাংশনের ভেরিয়েবলগুলো আলাদা। একে বলে লোকাল ভেরিয়েবল (local variable)। আমরা main ফাংশনের x, y-এর মান প্রিন্ট করেছি test_function ফাংশনের x, y-এর মান প্রিন্ট করিনি। এক ফাংশনের লোকাল ভেরিয়েবলের অস্তিত্ব অন্য ফাংশনে থাকে না। তুমি এখন কিছু প্রোগ্রাম লিখে আরও পরীক্ষা-নিরীক্ষা করে দেখতে পারো। কী প্রোগ্রাম লিখবে সেটি তোমার ওপর ছেড়ে দিলাম।

আমরা যদি চাই, কোনো ভেরিয়েবলের অস্তিত্ব আমাদের প্রোগ্রামের সব ফাংশনের ভেতরে থাকতে হবে, তবে আমরা সেটি করতে পারি গ্লোবাল (global) ভেরিয়েবল ডিক্লেয়ার করার মাধ্যমে। আমরা প্রোগ্রামের শুরুতে কোনো ফাংশন বা ফাংশনের প্রোটোটাইপ লেখার আগে সেগুলো ডিক্লেয়ার করে দেব। যেমন:

```
#include <stdio.h>

double pi = 3.14;

void my_fnc() {
    pi = 3.1416; /* এখানে আমরা pi-এর মান একটু পরিবর্তন করে দিলাম */
    return; /* ফাংশনের রিটার্ন টাইপ void হলে এই return; না দিলেও কিন্তু চলে */
}

int main() {
    printf("%lf\n", pi); /* এখানে pi-এর মান হবে 3.14 */
    my_fnc();
    printf("%lf\n", pi); /* এখানে pi-এর মান হবে 3.1416 কারণ আমরা সেটি my_fnc ফাংশনে গিয়ে বদলে দিয়েছি। */
```

```
    return 0;
}
```

আবার আমরা যদি my_fnc ফাংশনের ভেতরে গিয়ে pi নামে একটি ভেরিয়েবল ডিক্লেয়ার করতাম (double pi;), তবে সেটি একটি লোকাল ভেরিয়েবল হতো এবং গ্লোবাল pi-এর মানের কোন পরিবর্তন হতো না।

এতক্ষণ আমরা ফাংশনের প্যারামিটার হিসেবে কেবল ভেরিয়েবল ব্যবহার করেছি। এবারে আসো আমরা ফাংশনের প্যারামিটার হিসেবে অ্যারে পাঠাই। আমরা একটি প্রোগ্রাম লিখব যেটি কোনো একটি ইন্টিজার অ্যারে থেকে সবচেয়ে বড় সংখ্যাটি খুঁজে বের করবে। অ্যারে থেকে সর্বোচ্চ সংখ্যা খুঁজে বের করার কাজটি করার জন্য একটি ফাংশন লিখে ফেলি, কী বলো?

```
int find_max(int ara[], int n) { /* এখানে আমরা দুটি প্যারামিটার দিচ্ছি। প্রথমটা হচ্ছে একটি অ্যারে, আর তারপর একটি সংখ্যা যেটি নির্দেশ করবে অ্যারেতে কয়টি সংখ্যা আছে। লক্ষ করো, প্যারামিটারে যখন অ্যারের কথাটি বলে দিচ্ছি তখন সেখানে কয়টি উপাদান আছে সেটি না দিলেও চলে, যেমন আমরা int ara[11] ও লিখতে পারতাম। */

int max = ara[0]; /*  এখানে একটি ভেরিয়েবলে ধরে নিচ্ছি যে সবচেয়ে বড় সংখ্যাটি হচ্ছে অ্যারের প্রথম সংখ্যা। তারপরে আমরা অ্যারের বাকি উপাদানগুলোর সঙ্গে max কে তুলনা করব আর যদি অ্যারের কোনো উপাদানের মান max-এর চেয়ে বড় হয় তখন সেই মানটি max-এ রেখে দেব। অর্থাৎ তখন আবার max হয়ে যাবে ওই অ্যারের সর্বোচ্চ সংখ্যা। */

    int i;

    for(i = 1; i < n; i++) {
        if (ara[i] > max) {
            max = ara[i]; /* ara[i] যদি max-এর চেয়ে বড় হয় তবে max-এ ara[i]-এর মানটি অ্যাসাইন করে দিচ্ছি। */
        }
    }
```

```
    return max; /* ফাংশন থেকে সর্বোচ্চ মানটি ফেরত পাঠাচ্ছি */
}
```

এখন কথা হচ্ছে এই ফাংশনকে আমরা কল করব কীভাবে? ভেরিয়েবলের জায়গায় তো এর নাম দিয়ে কল করতে হয়, কিন্তু অ্যারের বেলায় কী দেব? অ্যারের বেলাতেও শুধু নাম দিলেই চলবে। পুরো প্রোগ্রামটি এবারে রান করে দেখো:

```
#include <stdio.h>

int find_max(int ara[], int n);

int main()
{
    int ara[] = {-100, 0, 53, 22, 83, 23, 89, -132, 201, 3, 85};
    int n = 11;
    int max = find_max(ara, n);
    printf("%d\n", max);

    return 0;
}

int find_max(int ara[], int n)
{
    int max = ara[0];
    int i;

    for(i = 1; i < n; i++) {
        if (ara[i] > max) {
            max = ara[i];
        }
```

```
    }

    return max;
}
```
প্রোগ্রাম: ৭.৭

এখন তোমরা find_min নামে আরেকটি ফাংশন লেখো যার কাজ হবে সবচেয়ে ছোট সংখ্যাটি খুঁজে বের করা। find_sum, find_average এসব ফাংশনও লিখে ফেলতে পারো। আর তোমাদের নিশ্চয়ই বলে দিতে হবে না এইসব ফাংশন কী কাজ করবে।

ফাংশনে ভেরিয়েবল পাস করা (pass, পাঠানো অর্থে) আর অ্যারে পাস করার মধ্যে একটি গুরুত্বপূর্ণ পার্থক্য রয়েছে। আমরা ইতিমধ্যে দেখেছি যে ফাংশনের ভেতর ভেরিয়েবল পাস করলে ওই ফাংশনের ভেতরে সেটির আরেকটি কপি তৈরি হয়, সুতরাং সেখানে ওই ভেরিয়েবলের মান পরিবর্তন করলে মূল ফাংশন (যেখান থেকে ফাংশন কল করা হয়েছে) ভেরিয়েবলের মানের কোনো পরিবর্তন হয় না। তবে অ্যারের বেলায় ব্যাপারটি আলাদা। আগে আমরা একটি প্রোগ্রাম লিখে দেখি:

```
#include <stdio.h>

void test_function(int ara[])
{
    ara[0] = 100;
    return;
}

int main()
{
    int ara [] = {1, 2, 3, 4, 5};
    printf("%d\n", ara[0]);
    test_function(ara);
    printf("%d\n", ara[0]);
```

```
    return 0;
}
```
প্রোগ্রাম: ৭.৮

এই প্রোগ্রামের আউটপুট কী হবে? প্রথম printf ফাংশনটি 1 প্রিন্ট করবে সেটি নিয়ে তো কোনো সন্দেহ নেই, কিন্তু দ্বিতীয় printf কী প্রিন্ট করবে? test_function-এর ভেতর আমরা অ্যারের প্রথম উপাদানের মান 100 অ্যাসাইন করেছি। এখন যদি সেটি মূল অ্যারেকে পরিবর্তন করে, তবে ara[0]-এর মান হবে 100, আর পরিবর্তন না হলে মান হবে আগে যা ছিল তা-ই, মানে 1।

আমরা আউটপুট দেখব 100, কারণ অ্যারেটির প্রথম উপাদানের মান পরিবর্তিত হয়েছে। অর্থাৎ আমরা বুঝতে পারলাম ফাংশনের ভেতরে অ্যারে পাস করলে ওই অ্যারের আলাদা কোনো কপি তৈরি হয় না। কারণ হচ্ছে আমরা ফাংশনের ভেতর অ্যারের নামটি কেবল পাঠাই, যেটি কিনা ওই অ্যারেটি মেমোরির কোন জায়গায় আছে তার অ্যাড্রেস।

এখন তোমরা বৃত্তের ক্ষেত্রফল নির্ণয়ের জন্য একটি ফাংশন লিখে ফেলো। ক্ষেত্রফল বের করার সূত্রটি মনে আছে তো? মনে না থাকলে জ্যামিতি বই থেকে দেখে নাও।

প্রোগ্রামিং শিখতে তিনটি জিনিস প্রয়োজনঃ
ধৈর্য্য, সময় ও কম্পিউটার।

অধ্যায় আটঃ বাইনারি সার্চ

একটি সহজ খেলা দিয়ে শুরু করা যাক। এটি খেলতে দুজন দরকার। একজন মনে মনে একটি সংখ্যা ধরবে। আর দ্বিতীয়জন কিছু প্রশ্ন করে সেই সংখ্যাটি বের করবে। তবে 'তোমার সংখ্যাটি কত?' - এমন প্রশ্ন কিন্তু সরাসরি করা যাবে না। প্রশ্নটি হচ্ছে:

- সংখ্যাটি কি N (একটি সংখ্যা)-এর চেয়ে বড়, ছোট নাকি সমান?

আর সংখ্যাটি কিন্তু একটি নির্দিষ্ট সীমার মধ্যে হতে হবে (যেমন 1 থেকে 100, 10 থেকে 1000, -1000 থেকে 100000)।
এখন ধরা যাক, প্রথমজন যে সংখ্যাটি ধরেছে সেটি 1 থেকে 1000-এর ভেতর একটি সংখ্যা। তাহলে কিন্তু সর্বোচ্চ এক হাজার বার 'সংখ্যাটি কি N-এর সমান?' প্রশ্নটি করে সেটি বের করে ফেলা যায়। (সংখ্যাটি কি 1? সংখ্যাটি কি 2? ... সংখ্যাটি কি 999, সংখ্যাটি কি 1000?)। এভাবে প্রশ্ন করতে থাকলে সংখ্যাটি অবশ্যই বের হবে। তবে ভাগ্য খারাপ হলে এক হাজার বার ওই প্রশ্নটি করতে হবে।

কিন্তু আমাদের তো এত সময় নেই। ধরা যাক, 1 থেকে 1000-এর ভেতর ওই সংখ্যাটি হচ্ছে 50। তাহলে আমাদের প্রথম প্রশ্ন হবে:
১) সংখ্যাটি কি 500-এর চেয়ে বড়, ছোট নাকি সমান? ছোট।
২) সংখ্যাটি কি 250-এর চেয়ে বড়, ছোট নাকি সমান? ছোট।
৩) সংখ্যাটি কি 125-এর চেয়ে বড়, ছোট নাকি সমান? ছোট।
৪) সংখ্যাটি কি 62-এর চেয়ে বড়, ছোট নাকি সমান? ছোট।
৫) সংখ্যাটি কি 31-এর চেয়ে বড়, ছোট নাকি সমান? বড়।
৬) সংখ্যাটি কি 46-এর চেয়ে বড়, ছোট নাকি সমান? বড়।
৭) সংখ্যাটি কি 54-এর চেয়ে বড়, ছোট নাকি সমান? ছোট।
৮) সংখ্যাটি কি 50-এর চেয়ে বড়, ছোট নাকি সমান? সমান।
আমরা মাত্র আটটি প্রশ্ন করেই সংখ্যাটি পেয়ে গেছি!

তোমরা নিশ্চয়ই পদ্ধতিটি বুঝে ফেলেছ? প্রতিবার প্রশ্ন করে সংখ্যাটি যে সীমার মধ্যে আছে তাকে অর্ধেক করে ফেলা হয়েছে। খেলা শুরুর সময় সীমাটি ছিল 1 থেকে 1000। তারপর সেটি হয়েছে 1 থেকে 500। তারপর 1 থেকে 250, 1 থেকে 125, 1 থেকে 62, 31 থেকে 62, 46 থেকে 62, 46 থেকে 54।

সংখ্যা বের করার এই পদ্ধতিকে বলে বাইনারি সার্চ। চলো আমরা তাহলে অ্যালগরিদমটি লেখার চেষ্টা করি:
বাইনারি সার্চ (low, high, N): (শুরুতে আমাদের তিনটি সংখ্যা জানতে হবে, সংখ্যাটির নিম্নসীমা (low), উচ্চসীমা (high) এবং সেই সংখ্যা (N))
ধাপ 1: mid = (low + high) / 2
ধাপ 2: যদি mid এবং N-এর মান সমান হয় তবে ধাপ 5-এ যাও।
ধাপ 3: যদি N, mid-এর চেয়ে বড় হয়, তাহলে low = mid + 1. ধাপ 1-এ যাও।
ধাপ 4: যদি N, mid-এর চেয়ে ছোট হয়, তাহলে high = mid - 1. ধাপ 1-এ যাও।
ধাপ 5: সংখ্যাটি পেয়ে গেছি (mid)।

এখন আমরা দেখব একটি অ্যারে থেকে কীভাবে বাইনারি সার্চ করে কোনো সংখ্যা খুঁজে বের করতে হয়। অ্যারেতে কিন্তু সংখ্যাগুলো ছোট থেকে বড় কিংবা বড় থেকে ছোট ক্রমানুসারে থাকতে হবে। নইলে বাইনারি সার্চ ব্যবহার করা যাবে না। কারণটি কি কেউ বলতে পারো?
প্রথমে আমরা একটি ইন্টিজার অ্যারে নিই যেখানে সংখ্যাগুলো ছোট থেকে বড় ক্রমানুসারে সাজানো আছে।
int ara[] = {1, 4, 6, 8, 9, 11, 14, 15, 20, 25, 33 83, 87, 97, 99, 100};

এখন বলো তো low আর high-এর মান কত হবে? low = 1 এবং high = 100 ? ঠিকই ধরেছ কিন্তু এখানে একটু সমস্যা আছে। আমরা এখানে সব সংখ্যার মধ্যে খুঁজব না, বরং অ্যারের ইনডেক্সের মধ্যে খুঁজব। আর অ্যারের ইনডেক্সগুলো ক্রমানুসারে থাকে বলেই অ্যারেতে বাইনারি সার্চ করা যায়। এখানে ara-এর সর্বনিম্ন ইনডেক্স হচ্ছে 0 এবং সর্বোচ্চ ইনডেক্স হচ্ছে 15। তাহলে আমরা দুটি ভেরিয়েবলের মান নির্দিষ্ট করে দিই -
low_indx = 0;
high_indx = 15;
যে সংখ্যাটি খুঁজব ধরা যাক সেটি হচ্ছে 97।
num = 97;

তোমাদের অনেকেই হয়তো ভাবছ, num সংখ্যাটি যদি ara-তে না থাকে তখন কী হবে? সেটিও আমরা দেখব। সংখ্যাটি যদি খুঁজে পাওয়া না যায় তবে সেটি জানিয়ে দেওয়ার ব্যবস্থা রাখতে হবে আমাদের প্রোগ্রামে।

আমাদের যেহেতু খোঁজার কাজটি বারবার করতে হবে, আমাদেরকে একটি লুপ ব্যবহার করতে হবে। লুপের ভেতর আমরা খোঁজাখুঁজি করব আর সংখ্যাটি পেয়ে গেলে (কিংবা সংখ্যাটি নেই সেটি নিশ্চিত হলে) আমরা লুপ থেকে বের হয়ে যাব।

```
while(1) {
        mid_indx = (low_indx + high_indx) / 2;

        if(num == ara[mid_indx]) {
                /* num যদি ara[mid_indx]-এর সমান হয়, তবে সেটি আমরা পেয়ে গেছি */
                break;
        }

        if(num < ara[mid_indx]) {
                /* num যদি ara[mid_indx]-এর ছোট হয়, তবে আমরা low_indx থেকে mid_indx – 1 সীমার মধ্যে খুঁজব। */
                high_indx = mid_indx – 1;
        }
        else {
                /* num যদি ara[mid_indx]-এর বড় হয়, তবে আমরা mid_indx + 1 থেকে high_indx সীমার মধ্যে খুঁজব। */
                low_indx = mid_indx + 1;
        }
}
```

বাইনারি সার্চের প্রোগ্রাম আমরা লিখে ফেললাম। খুবই সহজ-সরল প্রোগ্রাম। সংখ্যাটি খুঁজে না পাওয়া পর্যন্ত লুপটি চলতেই থাকবে, কারণ আমরা লিখেছি while(1) আর 1 সব সময় সত্যি। কিন্তু সংখ্যাটি যদি ara-তে না থাকে তবে লুপটি চলতেই থাকবে এবং আমাদের প্রোগ্রাম কখনো বন্ধ হবে না। সুতরাং একটা ব্যবস্থা করা দরকার। আচ্ছা, আমরা কীভাবে বুঝব যে সংখ্যাটি ara-তে নেই? তোমরা ইতিমধ্যে লক্ষ করেছ যে আমরা প্রতিবার সার্চের সীমাটা অর্ধেক করে ফেলি। এভাবে চলতে থাকলে একসময় ওই সীমার ভেতর একটি সংখ্যাই থাকবে। তখন low এবং high-এর মান সমান হবে। আর প্রতিবার যেহেতু হয় low-এর মান বাড়ছে নাহয় high-এর মান কমছে, সুতরাং যেবার low আর high সমান হবে, তার পরের বার low-এর মান high-এর মানের চেয়ে বেশি হবে। তখন আমরা বুঝব

যে সংখ্যাটি খুঁজে পাওয়া যায়নি। সুতরাং যতক্ষণ low <= high ততক্ষণ লুপটি চলবে। লুপ থেকে বের হয়ে যদি দেখি low > high, তখন বুঝব যে সংখ্যাটি খুঁজে পাওয়া যায়নি, আর না হলে বুঝব সংখ্যাটি খুঁজে পাওয়া গেছে এবং-এর মান ara[mid_indx]।
তাহলে পুরো প্রোগ্রামটি এবারে লিখে ফেলা যাক:

```
#include <stdio.h>

int main()
{
    int ara[] = {1, 4, 6, 8, 9, 11, 14, 15, 20, 25, 33 83, 87, 97, 99, 100};
    int low_indx = 0;
    int high_indx = 15;
    int mid_indx;
    int num = 97;

    while (low_indx <= high_indx) {
        mid_indx = (low_indx + high_indx) / 2;
        if (num == ara[mid_indx]) {
            break;
        }
        if (num < ara[mid_indx]) {
            high_indx = mid_indx – 1;
        }
        else {
            low_indx = mid_indx + 1;
        }
    }
```

```
    if (low_indx > high_indx) {
        printf("%d is not in the array\n", num);
    }
    else {
        printf("%d is found in the array. It is the %d th element of the
array.\n", ara[mid_indx], mid_indx);
    }

    return 0;
}
```
প্রোগ্রাম: ৮.১

এবার তোমাদের কাজ হবে বাইনারি সার্চের জন্য একটি আলাদা ফাংশন লেখা।

তামিম শাহরিয়ার সুবিন

মীর ওয়াসি আহমেদ

তাহমিদ রাফি

প্রোগ্রামিংয়ে হাতে খড়ি

DIMIK COMPUTING SCHOOL

অধ্যায় নয়ঃ স্ট্রিং (string)

তোমরা যারা string শব্দটির বাংলা অর্থ জানো, তাদের আতঙ্কিত হওয়ার কোনো কারণ নেই, প্রোগ্রামিংয়ে স্ট্রিং মোটেও দড়ি টানাটানির মতো কষ্টকর ব্যাপার নয়। আবার তোমাদের মধ্যে যারা একটু জ্ঞানী টাইপের তাদের মাথায় হয়তো স্ট্রিং থিওরী শব্দটি চলে এসেছে। যা-ই হোক, উদ্বেগের কোনো কারণ নেই।

এক বা একাধিক character মিলে string তৈরি হয়। সোজা কথায় স্ট্রিং হচ্ছে ক্যারেক্টার টাইপের অ্যারে। তবে প্রোগ্রামিংয়ে এটির ব্যবহার এতই বেশি যে কোনো কোনো ল্যাঙ্গুয়েজে স্ট্রিংকে আলাদা একটি ডাটা টাইপ হিসেবে ধরা হয়। তবে সি-তে আমরা char টাইপের অ্যারে দিয়েই স্ট্রিংয়ের কাজ করব।

নিচের উদাহরণগুলো লক্ষ করো:

```
char country[11] = {'B', 'a', 'n', 'g', 'l', 'a', 'd', 'e', 's', 'h', '\0'};

char country[] = {'B', 'a', 'n', 'g', 'l', 'a', 'd', 'e', 's', 'h', '\0'};

char country[] = “Bangladesh”;

char *country = “Bangladesh”;
```

এভাবে আমরা স্ট্রিং ডিক্লেয়ার করতে পারি। চারটি ডিক্লারেশন আসলে একই জিনিস। সবার শেষে একটি Null character ('\0') দিলে কম্পাইলার বুঝতে পারে এখানেই স্ট্রিংয়ের শেষ। আবার তৃতীয় উদাহরণে অ্যারের উপাদানগুলো আলাদা করে লেখা হয়নি, একসঙ্গে লেখা হয়েছে। এ ক্ষেত্রে কম্পাইলার নিজেই Null character বসিয়ে নেবে। চতুর্থ উদাহরণটি একটু অদ্ভুত। এখানে যে জিনিসটা ব্যবহার করা হয়েছে তার নাম পয়েন্টার (pointer)। এ বইতে এরকম জিনিস আমরা মাঝে মাঝে ব্যবহার করলেও বিস্তারিত আলোচনায় যাব না।

এবারে প্রোগ্রাম লেখার পালা।

```
#include <stdio.h>

int main()
{
     char country[] = {'B', 'a', 'n', 'g', 'l', 'a', 'd', 'e', 's', 'h', '\0'};

    printf("%s\n", country);

    return 0;
}
```

প্রোগ্রাম: ৯.১

এখানে লক্ষ করো যে printf-এর ভেতরে %s ব্যবহার করা হয়েছে স্ট্রিং প্রিন্ট করার জন্য। আর অ্যারেতে শেষের '\0'টা ব্যবহার না করলেও চলে আসলে। বর্তমানের কম্পাইলারগুলো এটি বুঝে নিতে পারে।

```
#include <stdio.h>

int main()
{
    char country[] = {'B', 'a', 'n', 'g', 'l', 'a', 'd', 'e', 's', 'h', ' ', 'i', 's', ' ', 'm', 'y', ' ', 'c', 'o', 'u', 'n', 't', 'r', 'y'};

    printf("%s\n", country);
```

```
    return 0;
}
```
প্রোগ্রাম: ৯.২

প্রোগ্রামটি চালাও। তারপর নিচের প্রোগ্রামটি চালাও। আউটপুটে কি পার্থক্য দেখতে পাচ্ছ? পার্থক্যের কারণটা কী?

```
#include <stdio.h>

int main()
{
     char country[] = {'B', 'a', 'n', 'g', 'l', 'a', 'd', 'e', 's', 'h', '\0', 'i', 's', ' ', 'm', 'y', ' ', 'c', 'o', 'u', 'n', 't', 'r', 'y'};

    printf("%s\n", country);

    return 0;
}
```
প্রোগ্রাম: ৯.৩

পার্থক্যটা কী সেটি তোমরা প্রোগ্রাম দুটি কম্পিউটারে চালালেই বুঝবে। পার্থক্যের কারণ হচ্ছে দ্বিতীয় প্রোগ্রামে স্ট্রিংয়ের ভেতরে এক জায়গায় '\0' থাকায় কম্পাইলার ধরে নিচ্ছে ওখানে স্ট্রিংটা শেষ হয়ে গেছে।
এবারে আমরা একটি প্রোগ্রাম লিখব। একটি স্ট্রিংয়ের ভেতরের সব অক্ষরকে বড় হাতের অক্ষরে (অর্থাৎ capital letter বা uppercase character) রূপান্তর করা। তবে এর জন্য আমাদের একটি জিনিস জানতে হবে। প্রতিটি অক্ষরের বিপরীতে কম্পিউটার একটি সংখ্যার কোড ব্যবহার করে। সেই কোড অনুযায়ী, 'A'-এর মান হচ্ছে 65, 'B'-এর মান হচ্ছে 66, 'C'-এর মান হচ্ছে 67... এভাবে 'Z'-এর মান হচ্ছে 90। তেমনই 'a' হচ্ছে 97, 'b' হচ্ছে 98 ... এভাবে 'z' হচ্ছে 122। সুতরাং কোনো ক্যারেক্টার বড় হাতের কি না সেটি আমরা নির্ণয় করতে পারি এভাবে:

if(ch >= 'A' && ch <= 'Z') অথবা if(ch >= 65 && ch <= 90)।

তেমনই ছোট হাতের অক্ষরের জন্য: if(ch >= 'a' && ch <= 'z') অথবা if(ch >= 97 && ch <= 122)।

এখন কোনো ক্যারেক্টার যদি ছোট হাতের হয়, তবে তাকে বড় হাতের অক্ষরে রূপান্তর করার উপায় কী? উপায় খুব সহজ। একটি উদাহরণ দেখো:

char ch = 'c';
ch = 'A' + (ch – 'a');

এখানে যেটি হচ্ছে, প্রথমে ch থেকে 'a' বিয়োগ করা হচ্ছে মানে 'c' থেকে 'a' বিয়োগ (আসলে 99 থেকে 97 বিয়োগ হচ্ছে)। বিয়োগফল 2। এবারে 'A'-এর সঙ্গে যদি ওই 2 যোগ করে দিই তবে সেটি 'C' হয়ে যাবে!

এখন প্রোগ্রামটি লিখে ফেলা যাক:

```
#include <stdio.h>

int main()
{
    char country[] = {'B', 'a', 'n', 'g', 'l', 'a', 'd', 'e', 's', 'h'};

    int i, length;

    printf("%s\n", country);

    length = 10;
```

```
    for(i = 0; i < length; i++) {
        if(country[i] >= 97 && country[i] <= 122) {
            country[i] = 'A' + (country[i] - 'a');
        }
    }

    printf("%s\n", country);

    return 0;
}
```

প্রোগ্রাম: ৯.৪

এখন তোমরা uppercase থেকে lowercase-এ রূপান্তরের প্রোগ্রামটি লিখে ফেলো। তারপরে আবার বইটি পড়া শুরু করো।

এখানে লক্ষ করো যে স্ট্রিংয়ে (বা ক্যারেক্টারের অ্যারেতে) মোট কয়টি উপাদান আছে সেটি আমি দেখেই লিখে ফেলেছি এবং সরাসরি বসিয়ে দিয়েছি length = 10।

এবার আমরা কোনো স্ট্রিংয়ের দৈর্ঘ্য মাপার জন্য একটি ফাংশন লিখব! এটি তেমন কঠিন কিছু নয়। একটি লুপের সাহায্যে স্ট্রিংয়ের প্রতিটি উপাদান পরীক্ষা করতে হবে এবং Null character ('\0') পেলে লুপ থেকে বের হয়ে যাবে অর্থাৎ, '\0' না পাওয়া পর্যন্ত লুপ চলতে থাকবে। আর লুপ যতবার চলবে স্ট্রিংয়ের দৈর্ঘ্যও তত হবে।

```
#include <stdio.h>

int string_length(char str[])
{
    int i, length = 0;
```

```
    for(i = 0; str[i] != '\0'; i++) {
        length++;
    }

    return length;
}

int main()
{
    char country[100];

    int length;

    while(1 == scanf("%s", country)) {
        length = string_length(country);
        printf("length: %d\n", length);
    }

    return 0;
}
```
প্রোগ্রাম: ৯.৫

ওপরের প্রোগ্রামটায় তোমরা দেখতে পাচ্ছ যে ইনপুট নেওয়ার জন্য scanf ফাংশন ব্যবহার করা হয়েছে এবং স্ট্রিং ইনপুট নেওয়ার জন্য %s ব্যবহৃত হয়েছে। scanf ফাংশনটি যতটি উপাদান ইনপুট হিসেবে নেয়, সেই সংখ্যাটি রিটার্ন করে। সাধারণত রিটার্ন ভ্যালুটি আমাদের দরকার হয় না, তাই scanf ব্যবহার করলেও আমরা ওই ভ্যালুটি রাখি না। যেমন দুটি ইন্টিজার ইনপুট নিতে গেলে আমরা লিখি: scanf(“%d %d”, &n1, &n2);। আমরা এটি চাইলে এভাবেও লিখতে পারতাম: value = scanf(“%d %d”, &n1,

&n2);। তোমরা প্রিন্ট করলে দেখবে value-এর মান 2। `while(1 == scanf("%s", country))` লাইনে যেটি ঘটছে তা হলো, যতক্ষণ একটি country-এর নাম scanf দিয়ে ইনপুট নেওয়া হচ্ছে, ফাংশনটি 1 রিটার্ন করছে, আর লুপের ভেতরের কন্ডিশন সত্য হচ্ছে (1 == 1), তাই লুপের কাজ চলতে থাকবে।

আরেকটি জিনিস খেয়াল করো যে country-এর আগে কোন & চিহ্ন ব্যবহার করা হয়নি। তোমরা &country লিখে দেখো প্রোগ্রামটি কী আচরণ করে। তবে %s ব্যবহারের একটি সমস্যা হচ্ছে স্ট্রিংয়ে কোনো হোয়াইটস্পেস ক্যারেক্টার (যেমন: স্পেস, ট্যাব ইত্যাদি) থাকা যাবে না, এমন কিছু পেলে scanf ওই ক্যারেক্টার পর্যন্ত একটি স্ট্রিং ধরে নেয়। যেমন, ইনপুট যদি হয় this is তবে scanf প্রথমে this কেই স্ট্রিং হিসেবে নেবে, তারপরে যদি আবার scanf ফাংশন কল করা হয়, তবে is কে সে স্ট্রিং হিসেবে ইনপুট নিয়ে নেবে। এই সমস্যা এড়ানোর জন্য আমরা gets ফাংশন ব্যবহার করতে পারি। নিচের উদাহরণটি দেখো:

```
#include <stdio.h>

int main()
{
    char ara[100];

    while(NULL != gets(ara)) {
        printf("%s\n", ara);
    }

    return 0;
}
```
প্রোগ্রাম: ৯.৬

এই প্রোগ্রামটিও চলতে থাকবে যতক্ষণ না তুমি ctrl + z (মানে কি-বোর্ডে ctrl ও z একসঙ্গে) চাপো, লিনাক্সের জন্য ctrl + d। ctrl + z বা ctrl + d দিলে gets ফাংশনটি NULL রিটার্ন করে। আরেকটি জিনিস লক্ষ করো যে আমি char ara[100]; ডিক্লেয়ার করে

শুরুতেই বলে দিয়েছি স্ট্রিংয়ের সর্বোচ্চ দৈর্ঘ্য হবে 100।

আরেকটি ব্যাপার। string_length ফাংশনের ভেতরে আসলে দুটি ভেরিয়েবল ব্যবহার না করলেও চলে। আমরা ফাংশনটি এভাবেও লিখতে পারি:

```
int string_length(char str[])
{
    int i;

    for(i = 0; str[i] != '\0'; i++);

    return i;
}
```

এখন তোমাদের কাজ হবে string_length ফাংশনটি for লুপ ব্যবহার না করে while লুপ ব্যবহার করে লেখা।
আমাদের পরবর্তী প্রোগ্রামের লক্ষ্য হবে দুটি স্ট্রিং জোড়া দেওয়া বা concatenate করা। যেমন একটি স্ট্রিং যদি হয় “bangla” এবং আরেকটি স্ট্রিং যদি হয় “desh” তবে দুটি জোড়া দিয়ে “bangladesh” বানাতে হবে।

প্রথমেই স্ট্রিংগুলো ডিক্লেয়ার করে নেই: char str1[] = "bangla", str2[] = "desh", str3[12];
আমাদের লক্ষ হচ্ছে str3 তে “bangladesh” রাখা। খুব সুবিধা হতো যদি আমরা এমন কিছু লিখতে পারতাম:
str3 = str1 + str2;

কিন্তু 'সি'-তে এভাবে দুটি অ্যারে বা স্ট্রিং যোগ করা যায় না। তাই একটি একটি করে str1-এর উপাদানগুলো str3 তে কপি করতে হবে, তারপর str2-এর উপাদানগুলো str3 তে কপি করতে হবে।

```
#include <stdio.h>
```

```c
int main()
{
    char str1[] = "bangla", str2[] = "desh", str3[12];

    int i, j, length1 = 6, length2 = 4;

    for(i = 0, j = 0; i < length1; i++, j++) {
        str3[j] = str1[i];
    }

    for(i = 0, j = 0; i < length2; i++, j++) {
        str3[j] = str2[i];
    }

    str3[j] = '\0';

    printf("%s\n", str3);

    return 0;
}
```
প্রোগ্রাম: ৯.৭

প্রোগ্রামটি চালাও। আউটপুট কী আসা উচিত? bangladesh। কিন্তু আউটপুট এসেছে desh। আসলে আমরা কিছু একটা ভুল করেছি। তোমাদের এখন সেই ভুলটি ঠিক করার চেষ্টা করা উচিত। অন্তত তিরিশ মিনিট চেষ্টার পরও যদি ভুল বের করতে না পারো তবে আবার বইটি পড়া শুরু করো।

```
for(i = 0, j = 0; i < length1; i++, j++) {
    str3[j] = str1[i];
}
```

এখানে আমরা শুরুতেই i-এর মান 0 করেছি কারণ i কে আমরা str1-এর ইনডেক্স হিসেবে ব্যবহার করব। j কে ব্যবহার করব str3-এর ইনডেক্স হিসেবে তাই j-এর মানও 0 করা হয়েছে। তারপর একে একে str1-এর উপাদানগুলো str3 তে কপি করছি এবং i ও j-এর মান 1 করে বাড়াচ্ছি (i++, j++)। লুপ শেষ হওয়ার পরে i ও j প্রত্যেকের মান হবে 6।
এখন পরের লুপে আমরা str2 কে str3-তে কপি করব। এখন str2-এর ইনডেক্স হিসেবে যদি i ব্যবহার করি, তবে তার মান লুপের শুরুতেই আবার 0 করে দিতে হবে। আমরা সেটি করেছি। কিন্তু ভুল করেছি সেই সঙ্গে j-এর মান 0 করে দিয়ে। j-এর মান 0 করলে তো str2-এর প্রথম (0 তম) উপাদান str3-এর প্রথম (0 তম) উপাদান হিসেবে কপি হবে, কিন্তু আমরা তো সেটি চাই না। আমরা চাই str2-এর প্রথম উপাদান হবে str3-এর সপ্তম উপাদান। তাহলে j-এর মান 0 করা যাবে না। তাই দ্বিতীয় লুপটি হবে এমন:

```
for(i = 0; i < length2; i++, j++) {
    str3[j] = str2[i];
}
```

আরেকটি ব্যাপার লক্ষ করো। দ্বিতীয় লুপ থেকে বের হবার পরে str3-এর শেষ ঘরে '\0' অ্যাসাইন করেছি (str3[j] = '\0';) যাতে স্ট্রিংটা যে ওখানেই শেষ, এটি কম্পাইলার বুঝতে পারে।

আমাদের পরবর্তী প্রোগ্রাম হবে দুটি স্ট্রিংয়ের মধ্যে তুলনা করা। অর্থাৎ দুটি স্ট্রিংয়ের মধ্যে ছোট, বড়, সমান নির্ণয় করা। সংখ্যার ক্ষেত্রে যেমন >, <, >=, <=, == চিহ্ন ব্যবহার করে তুলনা করা যায়, স্ট্রিংয়ের ক্ষেত্রে সেই ব্যবস্থা নাই। কিন্তু স্ট্রিংয়ের ক্ষেত্রে প্রায়ই আমাদের এই তুলনা করার দরকার পড়বে। যেমন ধরো, সর্টিংয়ের ক্ষেত্রে যেখানে ছোট থেকে বড় বা বড় থেকে ছোট ক্রমানুসারে সাজাতে হবে (alphabetical ordering)।
স্ট্রিংয়ে ছোট-বড় আবার কী? বেশি কথা বলে ব্যাখ্যা না করে কিছু উদাহরণ দিই, তাহলেই বুঝতে পারবে।

'aaa'-এর চেয়ে 'aab' বড়। আবার 'ba' ও 'ca'-এর মধ্যে 'ca' বড়।

এই প্রোগ্রামে আমরা একটি ফাংশন লিখব string_compare() যেটির কাজ হবে দুটি স্ট্রিংয়ের মধ্যে তুলনা করে প্রথমটি দ্বিতীয়টির চেয়ে বড় হলে 1 রিটার্ন করবে, ছোট হলে -1 আর দুটি সমান হলে 0 রিটার্ন করবে।

ফাংশনের রিটার্ন টাইপ হবে ইন্টিজার এবং প্যারামিটার হবে দুটি char টাইপের অ্যারে।

```
int string_compare(char a[], char b[])
{

}
```

আমাদের মূল কাজ হবে a-এর প্রথম উপাদানের সঙ্গে b-এর প্রথম উপাদান, a-এর দ্বিতীয় উপাদানের সঙ্গে b-এর দ্বিতীয় উপাদান এভাবে তুলনা করতে থাকা। যখনই a-এর কোনো উপাদান b-এর কোনো উপাদানের চেয়ে ছোট হবে, আমরা সঙ্গে সঙ্গে বলে দিতে পারি যে a, b-এর চেয়ে ছোট। সুতরাং -1 রিটার্ন করে ফাংশন থেকে বের হয়ে আসব। একইভাবে যখনই a-এর কোনো উপাদান b-এর কোনো উপাদানের চেয়ে বড় হবে, সঙ্গে সঙ্গে 1 রিটার্ন করে ফাংশন থেকে বের হয়ে আসব কারণ a, b-এর চেয়ে বড়। কিন্তু যদি সবগুলোই সমান হয়? তখন আমরা 0 রিটার্ন করব। তাতে বুঝব যে স্ট্রিং দুটি সমান।

```
int string_compare(char a[], char b[])
{
    int i, j;


    for(i = 0; a[i] != '\0' && b[i] != '\0'; i++) {
        if(a[i] < b[i]) {
            return -1;
```

```
        }
        if(a[i] > b[i]) {
            return 1;
        }
    }

    if(string_length(a) == string_length(b)) {
        return 0;
    }

    if(string_length(a) < string_length(b)) {
        return -1;
    }

    if(string_length(a) > string_length(b)) {
        return 1;
    }
}
```

স্ট্রিংয়ের বেসিক জিনিসগুলো নিয়ে আলোচনা করলাম। তবে মজার ব্যাপার হচ্ছে সি ল্যাঙ্গুয়েজে একটি হেডার ফাইল আছে, যার নাম string.h এবং ওইখানে বেশিরভাগ স্ট্রিং-সংক্রান্ত কাজের জন্য ফাংশন তৈরি করে দেওয়া আছে (যেমন: strcmp, strlen, strcpy ইত্যাদি)। তোমাদের দিয়ে কাজগুলো আমি আবার করালাম বলে দুঃখ পাওয়ার কোনো কারণ নেই, আমার ওপর রাগ করারও কিছু নেই। মৌলিক জিনিসগুলো শিখে রাখা সব সময়ই গুরুত্বপূর্ণ, যা তোমার প্রোগ্রামিং চিন্তাকে বিকশিত করবে।

এখন আমরা আরেকটি প্রোগ্রাম লিখব যেটি ইনপুট হিসেবে একটি স্ট্রিং নেবে (যেখানে অনেকগুলো শব্দ থাকবে)। এই স্ট্রিংয়ের সর্বোচ্চ দৈর্ঘ্য হবে 1000। শব্দগুলোর মাঝখানে এক বা একাধিক স্পেস থাকবে। আউটপুট হিসেবে প্রতিটি শব্দ আলাদা লাইনে প্রিন্ট করতে হবে। বিরামচিহ্নগুলো (punctuation) প্রিন্ট করা যাবে না এবং শব্দের প্রথম অক্ষর হবে বড় হাতের।

অনেক শর্ত দিয়ে ফেললাম। তবে প্রোগ্রামটি খুব কঠিন কিছু নয়। নিজে নিজে চেষ্টা করতে পারো।

আর না পারলে এখন চলো দেখি কীভাবে সমাধান করা যায়।
প্রথম কথা হচ্ছে, ইনপুট নেব কীভাবে? বুঝতেই পারছ যে ইনপুটে যেহেতু স্পেস থাকবে, scanf("%s") ব্যবহার করা যাবে না। তাই আমরা gets() ব্যবহার করব। তার পরের কথা হচ্ছে একটি শব্দে কোন কোন ক্যারেক্টার থাকতে পারে? যেহেতু বলা নেই, আমরা ধরে নিই 'a' থেকে 'z', 'A' থেকে 'Z' আর '0' থেকে '9' থাকবে।

তার পরের প্রশ্ন হচ্ছে, আমরা কখন বুঝব বা আমাদের প্রোগ্রামকে কীভাবে বোঝাবো যে একটি শব্দ শুরু হয়েছে?-এর জন্য আমরা একটি ভেরিয়েবল রাখতে পারি। ভেরিয়েবলের নাম যদি দিই is_word_started তাহলে এর মান 0 হলে বুঝব শব্দ শুরু হয়নি, শব্দ শুরু হলে এর মান আমরা 1 করে দেব। আবার শব্দ শেষ হলে 0 করে দেব। যখন দেখব শব্দ শুরু হয়ে গেছে (is_word_started-এর মান 1) কিন্তু কোনো ক্যারেক্টারের মান 'a' – 'z' বা 'A' – 'Z', বা '0' – '9' এই রেঞ্জের মধ্যে নেই, তখনই বুঝব শব্দটি শেষ। তোমরা যদি এর আগে প্রোগ্রামটি চেষ্টা করার পরও লিখতে না পারো, এখন চেষ্টা করলে পারবে আশা করি। আমি এখন কোডটি লিখে দেব তবে সেটি দেখার আগে অবশ্যই নিজে করার চেষ্টা করতে হবে।

```
#include <stdio.h>
#include <string.h>

int main()
{
    char s[1002], word[100];
    int i, j, length, is_word_started;

    gets(s);

    length = strlen(s);
```

```
is_word_started = 0;

for (i = 0, j = 0; i < length; i++) {
    if (s[i] >= 'a' && s[i] <= 'z') {
        if (is_word_started == 0) {
            is_word_started = 1;
            word[j] = 'A' + s[i] - 'a'; // first character is capital
            j++;
        }
        else {
            word[j] = s[i];
            j++;
        }
    }
    else if (s[i] >= 'A' && s[i] <= 'Z') {
        if (is_word_started == 0) {
            is_word_started = 1;
        }
        word[j] = s[i];
        j++;
    }
    else if (s[i] >= '0' && s[i] <= '9') {
        if (is_word_started == 0) {
            is_word_started = 1;
        }
        word[j] = s[i];
        j++;
    }
```

```
        else {
            if (is_word_started == 1) {
                is_word_started = 0;
                word[j] = '\0';
                printf("%s\n", word);
                j = 0;
            }
        }
    }

    return 0;
}
```

প্রোগ্রাম: ৯.৮

প্রোগ্রামটি বুঝতে কি একটু সমস্যা হচ্ছে? সে পরে দেখা যাবে, আগে প্রোগ্রামটি চটপট কম্পিউটারে টাইপ করে ফেলো, কম্পাইল ও রান করো। যারা লিনাক্স ব্যবহার করছ তারা gets() ব্যবহারের কারণে কম্পাইলার থেকে একটি সতর্ক সংকেত (warning) পেতে পারো, পাত্তা দিয়ো না।

ইনপুট হিসেবে যেকোনো কিছু লিখতে পারো। যেমন: This is a test.। আউটপুট কী?
আউটপুট হচ্ছে এই রকম:
This
Is
A

কী মুশকিল! test গেল কোথায়?
এখন তোমার কাজ হবে test-এর নিখোঁজ হওয়ার রহস্যটা তদন্ত করা। তারপর আমি প্রোগ্রামটি ব্যাখ্যা করব।

তোমরা দেখো প্রোগ্রামে আমি স্ট্রিংয়ের দৈর্ঘ্য নির্ণয়ের জন্য strlen ফাংশন ব্যবহার করেছি। আর-এর জন্য আমাকে string.h হেডার ফাইলটি include করতে হয়েছে। ইনপুট হিসেবে স্ট্রিংটা নিলাম s-এ। আর word রাখার জন্য একটি অ্যারে ডিক্লেয়ার করে রেখেছি। তারপর আমি i = 0 থেকে length পর্যন্ত একটি লুপ চালিয়েছি s-এর ভেতরের প্রতিটি ক্যারেক্টার পরীক্ষা করার জন্য।

`if (s[i] >= 'a' && s[i] <= 'z')` দিয়ে পরীক্ষা করলাম এটি ছোট হাতের অক্ষর নাকি। যদি ছোট হাতের অক্ষর হয় তবে একটি শব্দের প্রথম অক্ষর কি না সেটি জানতে হবে। কারণ প্রথম অক্ষর হলে ওটাকে আবার বড় হাতের অক্ষরে রূপান্তর করতে হবে। সেই পরীক্ষাটা আমরা করেছি: `if (is_word_started == 0)` দিয়ে। এটি সত্য হওয়া মানে শব্দ শুরু হয়নি, এটিই প্রথম অক্ষর। তাই আমরা is_word_started-এর মান 1 করে দেব। আর word[j]তে s[i]-এর বড় হাতের অক্ষরটা নেব। তারপর j-এর মান এক বাড়াতে হবে।

`else if (s[i] >= 'A' && s[i] <= 'Z')` এবং `else if (s[i] >= '0' && s[i] <= '9')` এই দুটি শর্তের ভেতরেই আমরা একই কাজ করি। s[i]কে word[j]তে কপি করি। তাই চাইলে দুটি শর্তকে একসঙ্গে এভাবেও লিখতে পারতাম:

```
else if ((s[i] >= 'A' && s[i] <= 'Z') || (s[i] >= '0' && s[i] <= '9'))
```

তার পরের else-এর ভেতরে ঢোকার মানে হচ্ছে আগের if এবং else if-এর শর্তগুলো মিথ্যা হয়েছে। তাই s[i]-এর ভেতরে যেই ক্যারেক্টার আছে সেটি word-এ রাখা যাবে না। এবং যদি word ইতিমধ্যে শুরু হয়ে গিয়ে থাকে, সেটি শেষ করতে হবে এবং word টি প্রিন্ট করতে হবে। আর যদি word শুরু না হয়ে থাকে তাহলে কিছু করার দরকার নেই।

```
else {
    if (is_word_started == 1) {
        is_word_started = 0;
        word[j] = '\0';
        printf("%s\n", word);
        j = 0;
    }
```

```
}
```

তোমরা কি test-রহস্য সমাধান করতে পেরেছ? তোমরা চেষ্টা করতে থাকো আর আমি এখন প্রোগ্রামটি অন্যভাবে লিখব (এর সঙ্গে test রহস্যের কোনো সম্পর্ক নেই সেটি বলে রাখলাম)।

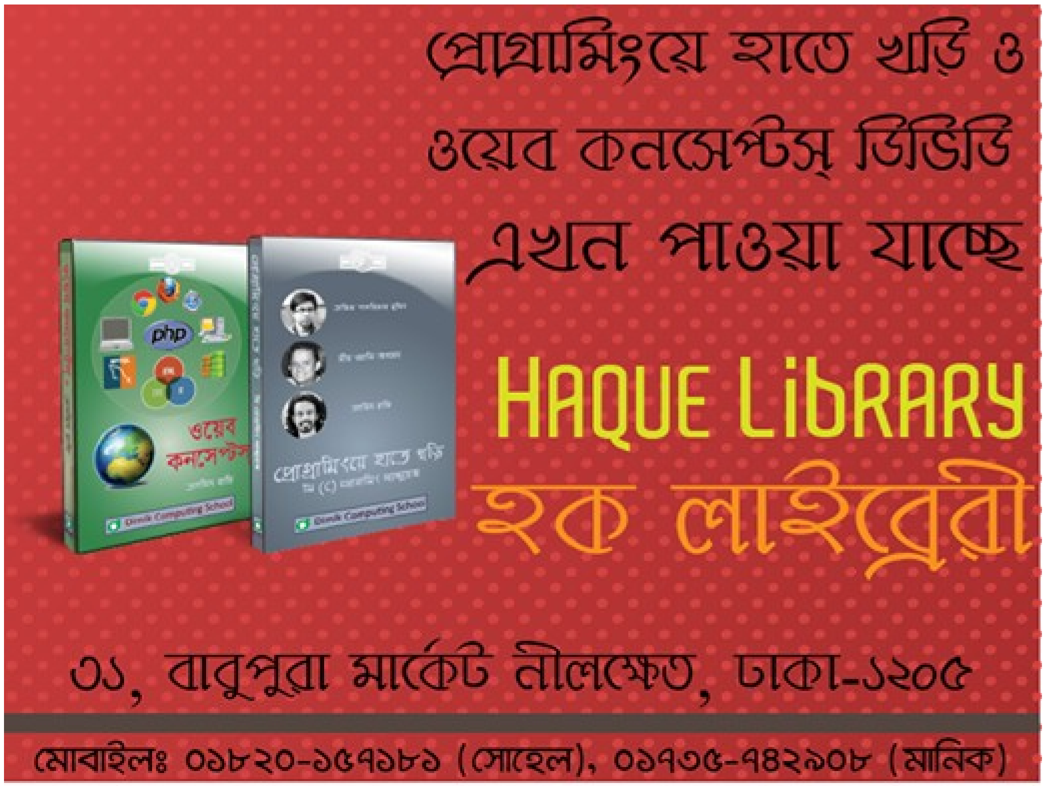
এখন আমি যেটি করব, প্রোগ্রামটি এমনভাবে লিখব যাতে word অ্যারেটিই ব্যবহার করতে না হয়! একটু চিন্তা করে দেখো। আসলে তো এই অ্যারেটি নিয়ে আমরা কিছু করছি না প্রিন্ট করা ছাড়া। তাই এর আসলে কোনো দরকার নেই।

```
#include <stdio.h>
#include <string.h>

int main()
{
    char s[1002], ch;
    int i, length, is_word_started;

    gets(s);
    length = strlen(s);
    is_word_started = 0;

    for (i = 0; i < length; i++) {
        if (s[i] >= 'a' && s[i] <= 'z') {
            if (is_word_started == 0) {
                is_word_started = 1;
                ch = 'A' + s[i] - 'a';
                printf("%c", ch);
            }
```

```
            else {
                printf("%c", s[i]);
            }
        }
          else if ((s[i] >= 'A' && s[i] <= 'Z') || (s[i] >= '0' && s[i] <=
'9')) {
            if (is_word_started == 0) {
                is_word_started = 1;
            }
            printf("%c", s[i]);
        }
        else {
            if (is_word_started == 1) {
                is_word_started = 0;
                printf("\n");
            }
        }
    }

    printf("\n");

    return 0;
}
```
প্রোগ্রাম: ৯.৯

এখন প্রোগ্রামটি বুঝতে চেষ্টা করো এবং বিভিন্ন ইনপুট দিয়ে পরীক্ষা করে দেখো। যেমন:
This is test number 9.9

স্ট্রিং-সংক্রান্ত সমস্যাগুলো দেখতে জটিল মনে হলেও আসলে সহজ। আর এ ধরনের সমস্যা সমাধানের যত চর্চা করবে দক্ষতা তত বাড়বে।



অধ্যায় দশঃ মৌলিক সংখ্যা

মৌলিক সংখ্যা (Prime Number) গণিতবিদদের কাছে যেমন প্রিয়, তেমনই প্রোগ্রামারদেরও অনেক প্রিয় একটি বিষয়। তোমাদের বিভিন্ন সময়ে এই মৌলিক সংখ্যাসংক্রান্ত নানা সমস্যার সমাধান করতে হবে। মৌলিক সংখ্যা জিনিসটি যে গুরুত্বপূর্ণ সেটি বোঝার আরেকটি উপায় হলো, এই বইতে বিষয়টির জন্য আমি একটি পৃথক অধ্যায় বরাদ্দ করেছি।

মৌলিক সংখ্যা হচ্ছে সেসব সংখ্যা যারা 1-এর চেয়ে বড় পূর্ণসংখ্যা এবং সেটি কেবল 1 এবং ওই সংখ্যাটি দ্বারাই নিঃশেষে বিভাজ্য হবে। খুবই সহজ-সরল জিনিস। এখন কোনো সংখ্যা মৌলিক কি না সেটি বের করার জন্য একটি প্রোগ্রাম লিখে ফেলা যাক।

```
#include <stdio.h>

int is_prime(int n)
{
    int i;

    if (n < 2) {
        return 0;
    }

    for(i = 2; i < n; i++) {
        if(n % i == 0) {
            return 0;
        }
    }

    return 1;
}
```

```
int main()
{
    int n;

    while(1) {
        printf("Please enter a number (enter 0 to exit): ");
        scanf("%d", &n);
        if(n == 0) {
            break;
        }
        if(1 == is_prime(n)) {
            printf("%d is a prime number.\n", n);
        }
        else {
            printf("%d is not a prime number.\n", n);
        }
    }

    return 0;
}
```
প্রোগ্রাম: ১০.১

মৌলিক সংখ্যা নির্ণয়ের জন্য আমরা একটি ফাংশন লিখেছি যেটির প্যারামিটার হচ্ছে একটি ইন্টিজার নম্বর n। ফাংশনে আমরা n কে 2 থেকে n-1 পর্যন্ত সংখ্যাগুলো দিয়ে ভাগ করার চেষ্টা করেছি একটি লুপের সাহায্যে। যদি এর মধ্যে কোনো সংখ্যা দিয়ে n নিঃশেষে বিভাজ্য হয়, তবে আমরা সঙ্গে সঙ্গেই বলে দিতে পারি যে সেটি মৌলিক সংখ্যা নয় এবং ফাংশনটি 0 রিটার্ন করে। আর যদি সব সংখ্যা দিয়ে ভাগ করার পরও দেখা যায় যে কোন সংখ্যাই n কে নিঃশেষে ভাগ করতে পারেনি, তখন আমরা এই সিদ্ধান্তে আসতে পারি যে n

একটি মৌলিক সংখ্যা। আর তখন ফাংশন থেকে 1 রিটার্ন করি। আমরা মৌলিক সংখ্যা নির্ণয় করা শিখে গেলাম!

আমি প্রোগ্রামটি লেখার সময় যে পথ অবলম্বন করেছি সেটি হচ্ছে খুব সহজ-সরল পথ। প্রোগ্রামটিকে মোটেও ইফিশিয়েন্ট (efficient) বানানোর চেষ্টা করিনি। তোমরা খুব সহজেই ব্যাপারটি বুঝতে পারবে। প্রোগ্রামে ইনপুট হিসেবে 2147483647 দাও। এটি যে মৌলিক সংখ্যা সেটি বের করতে বেশ সময় লাগে। কারণ তখন 2147483647 কে 2 থেকে 2147483646 পর্যন্ত সব সংখ্যা দিয়ে ভাগ করার ব্যর্থ চেষ্টা করা হয়। প্রোগ্রামটিকে আরও ইফিশিয়েন্ট করতে হবে। একটি বুদ্ধি তোমাদের মাথায় এর মধ্যেই নিশ্চয়ই এসে গেছে। সেটি হচ্ছে 2 থেকে n-1 পর্যন্ত সব সংখ্যা দিয়ে ভাগ করার চেষ্টা না করে 2 থেকে n/2 পর্যন্ত সংখ্যাগুলো দিয়ে ভাগ করার চেষ্টা করলেই হয়। তাহলে প্রোগ্রামের গতি দ্বিগুণ হয়ে যাবে।

এখন তোমরা আরেকটি বিষয় লক্ষ করো। কোন সংখ্যা যদি 2 দিয়ে নিঃশেষে বিভাজ্য না হয়, তবে সেটি অন্য কোন জোড় সংখ্যা দিয়ে নিঃশেষে বিভাজ্য হওয়ার প্রশ্নই আসে না। তাই 2 বাদে অন্য জোড় সংখ্যাগুলো (4, 6, 8, ...) দিয়ে ভাগ করার চেষ্টা করাটা আসলে বোকামি। জোড় সংখ্যা দিয়ে বিভাজ্যতার পরীক্ষাটা আমরা ফাংশনের শুরুতেই করে নিতে পারি।

এখন আমাদের ফাংশনটির চেহারা দাঁড়াবে এই রকম:

```
int is_prime(int n)
{
    int i;

    if (n < 2) {
        return 0;
    }

    if(n == 2) {
        return 1;
    }
```

```
    if(n % 2 == 0) {
        return 0;
    }

    for(i = 3; i <= n / 2; i = i + 2) {
        if(n % i == 0) {
            return 0;
        }
    }

    return 1;
}
```

প্রথমে আমরা পরীক্ষা করেছি n-এর মান 2 কি না। যদি 2 হয় তবে বলে দিয়েছি যে n মৌলিক সংখ্যা। তারপরে আমরা পরীক্ষা করেছি n জোড় সংখ্যা কি না। যদি জোড় হয়, তবে n মৌলিক সংখ্যা না, কেবল 2 ই একমাত্র জোড় মৌলিক সংখ্যা যেটির পরীক্ষা আমরা একেবারে শুরুতেই করে ফেলেছি। তারপর আমরা 3 থেকে n / 2 পর্যন্ত সব বেজোড় সংখ্যা দিয়ে n কে ভাগ করার চেষ্টা করেছি। এখন তোমরা বিভিন্ন ইনপুট দিয়ে প্রোগ্রামটি পরীক্ষা করে দেখো। 2147483647 দিয়ে পরীক্ষা করলে বুঝতে পারবে যে প্রোগ্রামের গতি আগের চেয়ে বেড়েছে কিন্তু তার পরও একটু সময় লাগছে। আমার কম্পিউটারে চার সেকেন্ডের মতো সময় লাগছে। কিন্তু এত সময় তো দেওয়া যাবে না।
তোমাদের যাদের গাণিতিক বুদ্ধিশুদ্ধি বেশি, তারা একটু চিন্তা করলেই প্রোগ্রামটির গতি বাড়ানোর একটি উপায় বের করে ফেলতে পারবে। সেটি হচ্ছে n-এর উৎপাদক বের করার জন্য আসলে n / 2 পর্যন্ত সব সংখ্যা দিয়ে পরীক্ষা করার দরকার নেই। n-এর বর্গমূল পর্যন্ত পরীক্ষা করলেই হয়। n = p x q হলে, p বা q যেকোনো একটি সংখ্যা অবশ্যই n-এর বর্গমূলের সমান বা তার ছোট হবে। বর্গমূল নির্ণয়ের জন্য আমরা math.h হেডার ফাইলের sqrt() ফাংশনটি ব্যবহার করব।

আমাদের প্রোগ্রামটি দাঁড়াচ্ছে এই রকম:

```c
#include <stdio.h>
#include <math.h>

int is_prime(int n)
{
    int i, root;

    if(n == 2) {
        return 1;
    }

    if(n % 2 == 0) {
        return 0;
    }

    root = sqrt(n);

    for(i = 3; i <= root; i = i + 2) {
        if(n % i == 0) {
            return 0;
        }
    }

    return 1;
}

int main()
{
```

```
    int n, m;

    while(1) {
        printf("Please enter a number (enter 0 to exit): ");
        scanf("%d", &n);

        if(n == 0) {
            break;
        }
        if(1 == is_prime(n)) {
            printf("%d is a prime number.\n", n);
        }
        else {
            printf("%d is not a prime number.\n", n);
        }
    }

    return 0;
}
```
প্রোগ্রাম: ১০.২

এখন তোমরা প্রোগ্রামটি চালিয়ে বিভিন্ন ইনপুট দিয়ে পরীক্ষা করে দেখো। একটি কথা বলে দিই। প্রোগ্রামটায় একটি বাগ আছে (মানে ভুল আছে)। সেটি খুঁজে বের করে ঠিক করে ফেলো।

প্রাইম নম্বর বের করতে পেরে তোমরা নিশ্চয়ই বেশ খুশি? কিন্তু আমাদের চেষ্টা এখানেই থেমে থাকবে না। আমরা এখন দেখব আরেকটি চমৎকার পদ্ধতি, গ্রিক গণিতবিদ ইরাতোসথেনেস (Eratosthenes) আজ থেকে দুই হাজার বছরেরও আগে এই পদ্ধতি আবিষ্কার করেছিলেন। এজন্য-এর নাম হচ্ছে সিভ অব ইরাতোসথেনেস (Sieve of Eratosthenes)। পদ্ধতিটি ব্যাখ্যা করা যাক।

ধরো, আমরা 2 থেকে 40 পর্যন্ত সব মৌলিক সংখ্যা বের করব। শুরুতে সব সংখ্যা লিখে ফেলি:

2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 , 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40.

এখন দেখো, তালিকার প্রথম সংখ্যা হচ্ছে 2। এবারে 2-এর সব গুণিতক (2 বাদে, মানে 2-এর চেয়ে বড়গুলো আরকী) বাদ দিয়ে দাও। তাহলে থাকবে:

2, 3, 5, 7, 9, 11, 13, 15, 17, 19 , 21, 23, 25, 27, 29, 31, 33, 35, 37, 39.

এখন তালিকার দ্বিতীয় সংখ্যা 3-এর সব গুণিতক বাদ দাও।

2, 3, 5, 7, 11, 13, 17, 19, 23, 25, 29, 31, 35, 37.

এখন তালিকার তৃতীয় সংখ্যা 5-এর সব গুণিতক বাদ দাও।

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37.

পরবর্তী সংখ্যা হচ্ছে 7 কিন্তু সেটির গুণিতক খোঁজার চেষ্টা করা বৃথা। কারণ তালিকার সর্বোচ্চ সংখ্যা 37-এর বর্গমূল 7-এর চেয়ে ছোট। সুতরাং 7-এর যে গুণিতকগুলো তালিকায় ছিল সেগুলো ইতিমধ্যে তালিকা থেকে বাদ পড়েছে। কারণটি বুঝতে সমস্যা হচ্ছে? দেখো 7-এর গুণিতকগুলো ছিল 14, 21, 28, 35। 7-এর সঙ্গে যেসব সংখ্যা গুণ করে ওই গুণিতকগুলো পাওয়া যায় সেগুলো সবই 7-এর চেয়ে ছোট সংখ্যা এবং তাদের গুণিতকগুলো আমরা ইতিমধ্যেই বাদ দিয়ে দিয়েছি।

এবারে ইমপ্লিমেন্ট করার পালা। আমরা তালিকা রাখার জন্য একটি অ্যারে ব্যবহার করব। ধরা যাক, তার নাম হচ্ছে ara। অ্যারেটি এমনভাবে তৈরি করতে হবে, যাতে কোনো একটি সংখ্যা n-এর অবস্থা (অর্থাৎ সেটি মৌলিক কি না) ara[n] দিয়ে প্রকাশ করা যায়। যদি ara[n]-এর মান 1 হয়, তবে n মৌলিক সংখ্যা আর ara[n]-এর মান 0 হলে n মৌলিক সংখ্যা নয়। ইমপ্লিমেন্টেশনের আগে অ্যালগরিদমটা লেখা যাক:

ধাপ ১: ধরা যাক, অ্যারেতে n টি উপাদান আছে। শুরুতে অ্যারের সব উপাদানের মান 1 বসাই।

ধাপ ২: অ্যারের প্রতিটি উপাদানের জন্য সেটির মান 1 কি না তা পরীক্ষা করি। যদি 1, হয় তবে তৃতীয় ধাপে যাই।

ধাপ ৩: ওই সংখ্যাকে 2 থেকে m পর্যন্ত ক্রমিক সংখ্যাগুলো দিয়ে গুণ করি এবং গুণফল যত হবে, অ্যারের তত নম্বর উপাদানে শূন্য (0) বসাই। অর্থাৎ সেটি যে মৌলিক নয় তা চিহ্নিত করি। এখানে m-এর মান এমন হবে যেন ঐ সংখ্যার সঙ্গে m-এর গুণফল n-এর চেয়ে ছোট বা সমান হয়।

এখন তোমরা কোডটি লেখার চেষ্টা করো। কমপক্ষে তিন ঘণ্টা নিজে চেষ্টা করার পর এবারে আমার কোড দেখো।

```
#include <stdio.h>
#include <math.h>

const int size = 40;
int ara[size];


void print_ara()
{
    int i;

    for(i = 2; i < size; i++) {
        printf("%4d", ara[i]);
    }
    printf("\n");
    for(i = 2; i < size; i++) {
        printf("----");
    }
    printf("\n");
    for(i = 2; i < size; i++) {
        printf("%4d", i);
    }
    printf("\n\n\n");
}

void sieve()
```

```
{
    int i, j, root;

    for(i = 2; i < size; i++) {
        ara[i] = 1;
    }

    root = sqrt(size);
    print_ara();
    for(i = 2; i <= root; i++) {
        if(ara[i] == 1) {
            for(j = 2; i * j <= size; j++) {
                ara[i * j] = 0;
            }
            print_ara();
        }
    }
}

int is_prime(int n)
{
    int i;

    if(n < 2) {
        return 0;
    }

    return ara[n];
```

```
}

int main()
{
    int n, m;

    sieve();

    while(1) {
        printf("Please enter a number (enter 0 to exit): ");
        scanf("%d", &n);

        if(n == 0) {
            break;
        }
        if(n >= size) {
            printf("The number should be less than %d\n", size);
            continue;
        }

        if(1 == is_prime(n)) {
            printf("%d is a prime number.\n", n);
        }
        else {
            printf("%d is not a prime number.\n", n);
        }
    }
```

```
    return 0;
}
```
প্রোগ্রাম: ১০.২

প্রতিবার অ্যারের অবস্থা বোঝানোর জন্য আমি একটি ফাংশন ব্যবহার করেছি, print_ara()। তোমরা দেখো এবারে ইনপুট নেওয়ার আগেই আমরা sieve() ফাংশন কল করে অ্যারেটি তৈরি করে ফেলেছি। তারপর যতবারই ইনপুট নাও, কোনো চিন্তা নেই, ইনপুট যদি n হয় তবে ara[n]-এর মান পরীক্ষা করলেই চলে, মান যদি 1 হয় তবে n মৌলিক সংখ্যা, যদি 0 হয় তবে n মৌলিক সংখ্যা নয়। কত পর্যন্ত সংখ্যা হিসাব করতে চাও সেটি size-এ বসিয়ে দিলেই হবে। এখন এই প্রোগ্রামে গতি নিয়ে কোনো সমস্যা নেই। খুবই ফাস্ট (fast)। কিন্তু আর কোনো সমস্যা তোমাদের চোখে পড়ছে? তোমরা কি বুঝতে পারছ যে প্রোগ্রামটি অনেক বেশি মেমোরি খরচ করে? ধরো, আমরা যদি 100 কোটি পর্যন্ত সংখ্যা মৌলিক কি না সেটি বের করতে চাই, তাহলে তো আমাদের 100 কোটির একটি অ্যারে দরকার হবে। 'সময় বাঁচাব না মেমোরি' সমস্যায় প্রোগ্রামারদের প্রায়ই পড়তে হয়। আমাদের সমস্যার ক্ষেত্রে আমরা একটি মাঝামাঝি সমাধানে পৌঁছতে পারি। n-এর সর্বোচ্চ মান যত হবে তার বর্গমূলটিকে size-এর মান হিসেবে নিতে পারি। তোমাকে যদি বলা হয়, n-এর মান সর্বোচ্চ 100000000 (দশ কোটি) পর্যন্ত হতে পারে তাহলে তুমি এর বর্গমূল অর্থাৎ 10000 পর্যন্ত সংখ্যাগুলোর জন্য sieve ফাংশন ব্যবহার করে মৌলিক সংখ্যাগুলো বের করবে। তারপর কী করবে? নাহ্, আর কিছু বলা যাবে না, তোমরাই চিন্তা করে ঠিক করো কী করবে।

আরেকটি কথা বলে দেওয়া দরকার। একটি ইন্টিজার কিন্তু চার বাইট জায়গা দখল করে, যেখানে একটি ক্যারেক্টার করে এক বাইট। সুতরাং ইন্টিজারের পরিবর্তে তোমরা ক্যারেক্টারের অ্যারে ব্যবহার করে মেমোরি খরচ চার ভাগের এক ভাগে নামিয়ে আনতে পারো। আমাদের তো আসলে ইন্টিজার অ্যারের দরকার নেই, কারণ অ্যারেতে কেবল দুই ধরনের মান থাকবে 0 বা 1। এটি ছাড়াও আমার লেখা sieve ফাংশনে আরও বেশ কিছু উপায় আছে ইফিসিয়েন্সি বাড়ানোর। এর মধ্যে একটি হচ্ছে গুণের বদলে যোগ করা। তোমরা সেটি করার চেষ্টা করো।

বইটি কমপক্ষে দুইবার ভালোভাবে পড়লে এবং সাথে যেই কাজগুলো করতে বলা হয়েছে সেগুলো ঠিকভাবে করলে তুমি প্রোগ্রামিং শেখার জন্য তৈরি হয়ে যাবে!

অধ্যায় এগারঃ আবারও অ্যারে

গণিত শিক্ষকের জন্য লেখা প্রোগ্রামটির কথা মনে আছে তো? সেই যে আমরা তিনটি অ্যারে ব্যবহার করে প্রোগ্রাম লিখেছিলাম ছাত্র-ছাত্রীদের গণিত পরীক্ষার মোট নম্বর বের করার জন্য। মনে না থাকলে প্রোগ্রামটি আবার দেখে নাও।

আমরা প্রথম সাময়িক পরীক্ষার নম্বর রেখেছিলাম একটি অ্যারেতে, দ্বিতীয় সাময়িক পরীক্ষার নম্বর আরেকটি অ্যারেতে, ফাইনাল পরীক্ষার নম্বর আরেকটি অ্যারেতে আর মোট নম্বর আরও একটি অ্যারেতে – মোট চারটি অ্যারে ব্যবহার করেছি। তো এখন যদি পুরো স্কুলের ফলাফল নির্ণয়ের জন্য প্রোগ্রাম লেখতে হয়, তবে সব ক্লাসের সব ছাত্র-ছাত্রীর সব বিষয়ের সব পরীক্ষার জন্য একটি করে অ্যারে ব্যবহার করা খুবই ঝামেলার কাজ হয়ে যাবে। তাই মোটামুটি সব প্রোগ্রামিং ল্যাঙ্গুয়েজেই মাল্টি-ডাইমেনশনাল অ্যারের ব্যবস্থা আছে। এতক্ষণ আমরা যেসব অ্যারে ব্যবহার করেছি, তার সবগুলোই ছিল এক ডাইমেনশনের অ্যারে।

এখন ওই প্রোগ্রামটি আমরা একটু অন্যভাবে লিখব, একটি মাত্র অ্যারে ব্যবহার করে। আপাতত কষ্ট কমানোর জন্য ধরি ক্লাসে মোট দশজন ছাত্র-ছাত্রী আছে। নিচের ছকে তাদের নম্বরগুলো লিখলাম:

	Roll: 1	Roll: 2	Roll: 3	Roll: 4	Roll: 5	Roll: 6	Roll: 7	Roll: 8	Roll: 9	Roll: 10
First terminal exam	80	70	92	78	58	83	85	66	99	81
Second terminal exam	75	67	55	100	91	84	79	61	90	97
Final exam	98	67	75	89	81	83	80	90	88	77
Total marks										

Total Marks সারির ঘরগুলো ফাঁকা, কারণ এগুলো এখনো হিসাব করিনি। প্রথম সাময়িক পরীক্ষার 25%, দ্বিতীয় সাময়িক পরীক্ষার 25% এবং ফাইনাল পরীক্ষার 50% নম্বর যোগ করে মোট নম্বর বের করতে হবে। এখন দেখো, আমাদের ছকে তাদের নম্বরগুলো রাখতে হয়েছে 4 টা সারি (row) এবং 10 টা কলামে। আমরা আগে যেই প্রোগ্রাম লিখেছিলাম, তাতে প্রথম রো-এর জন্য একটি অ্যারে, দ্বিতীয় রো-এর জন্য একটি, তৃতীয় রো-এর জন্য একটি এবং চতুর্থ রো-এর জন্য একটি অ্যারে ব্যবহার করেছিলাম। এখন

ব্যবহার করব একটি 2-D অ্যারে (টু ডাইমেনশনাল অ্যারে)।

2-D অ্যারে ডিক্লেয়ার করার নিয়ম হচ্ছে: data type array name [number of rows][number of columns];

যেমন ওপরের ছকটা যদি marks নামের একটি 2-D অ্যারেতে রাখতে চাই, তবে লিখতে হবে: int marks[4][10];

এখন, অ্যারের প্রথম রো হচ্ছে marks[0], দ্বিতীয় রো হচ্ছে marks[1], তৃতীয় রো হচ্ছে marks[2] এবং চতুর্থ রো হচ্ছে marks[3]। আবার marks[0][0] হচ্ছে প্রথম রো-এর প্রথম কলাম, marks[0][1] হচ্ছে প্রথম রো-এর দ্বিতীয় কলাম, marks[0][5] হচ্ছে প্রথম রো-এর ষষ্ঠ কলাম, marks[1][0] হচ্ছে দ্বিতীয় রো-এর প্রথম কলাম, marks[2][3] হচ্ছে তৃতীয় রো-এর চতুর্থ কলাম। আশা করি, বুঝতে পেরেছ।

এখন বলো তো, যার রোল নম্বর 10 তার দ্বিতীয় সাময়িক পরীক্ষার নম্বর কোন ঘরে আছে? আর marks[0][0] ঘরে কার এবং কোন পরীক্ষার নম্বর আছে?

marks[0][0]-এ থাকবে রোল 1-এর প্রথম সাময়িক পরীক্ষার নম্বর আর marks[1][9]-এ থাকবে রোল 10-এর দ্বিতীয় সাময়িক পরীক্ষার নম্বর। অ্যারেতে সংখ্যাগুলো এভাবে রাখতে পারি:

int marks[4][10] = {{80, 70, 92, 78, 58, 83, 85, 66, 99, 81}, {75, 67, 55, 100, 91, 84, 79, 61, 90, 97}, {98, 67, 75, 89, 81, 83, 80, 90, 88, 77}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}};

এখানে খেয়াল করেছ যে আমরা আসলে একটি অ্যারের ভেতর চারটি এক ডাইমেনশনের অ্যারে রেখেছি। marks[0]তে আছে প্রথম সাময়িক পরীক্ষায় সবার নম্বর, marks[1]-এ দ্বিতীয় সাময়িক পরীক্ষার নম্বর, marks[2]-এ ফাইনাল পরীক্ষার নম্বর এবং marks[3]তে মোট নম্বর (যেহেতু এখনো এটি হিসাব করিনি, তাই সব 0 লিখে দিলাম)।

এখন সম্পূর্ণ প্রোগ্রামটি তোমার কম্পিউটারে টাইপ করে কম্পাইল ও রান করো।

```
#include <stdio.h>

int main()
{
    int marks[4][10] = {{80, 70, 92, 78, 58, 83, 85, 66, 99, 81}, {75, 67,
55, 100, 91, 84, 79, 61, 90, 97}, {98, 67, 75, 89, 81, 83, 80, 90, 88, 77},
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0}};

    int col;

    for(col = 0; col < 10; col++) {
        marks[3][col] = marks[0][col] / 4.0 + marks[1][col] / 4.0 + marks[2]
[col] / 2.0;
        printf("Roll NO: %d   Total Marks: %d\n", col + 1, marks[3][col]);
    }

    return 0;
}
```

প্রোগ্রাম: ১১.১

নম্বরগুলো আগে না লিখে যদি আমরা ব্যবহারকারীর কাছ থেকে ইনপুট হিসেবে নিতে চাইতাম তাহলে কী করতে হতো?

```
int marks[4][10];
int i, j;

for (i = 0; i < 4; i++) {
```

```
    for (j = 0; j < 10; j++) {
        scanf("%d", &ara[i][j]);
    }
}
```

এভাবে নেস্টেড লুপের সাহায্যে আমরা ইনপুট নিতে পারি। প্রথম লুপটি ব্যবহার করা হয়েছে রো-এর জন্য আর দ্বিতীয় লুপটি কলামের জন্য। যখন i = 0, অর্থাৎ প্রথম রো-এর জন্য আমরা j = 0 থেকে 9 পর্যন্ত সব ঘরের ইনপুট নিচ্ছি, তারপর আবার i = 1 (দ্বিতীয় রো)-এর জন্য j = 0 থেকে 9 (প্রতি কলাম) পর্যন্ত সব ঘরের মান ইনপুট নেওয়া হচ্ছে।

এখন আমরা 1 থেকে 10 পর্যন্ত সংখ্যাগুলোর নামতা বের করার জন্য 2-D অ্যারে ব্যবহার করে একটি প্রোগ্রাম লিখব। একের নামতা হবে প্রথম রো-তে, দুইয়ের নামতা দ্বিতীয় রো-তে ... দশের নামতা দশম রো-তে থাকবে। তোমরা কি প্রোগ্রামটি নিজে নিজে লেখার চেষ্টা করবে? এক ঘণ্টার মধ্যেও যদি না হয়, তবে আমার কোডটি দেখো। প্রোগ্রামিং শেখার সময় অনেক সহজ প্রোগ্রাম লিখতেও প্রচুর সময় লাগে, তাতে হতাশ হবার কিছু নেই।

```
#include <stdio.h>

int main()
{
    int namta[10][10];

    int row, col;

    for (row = 0; row < 10; row++) {
        for(col = 0; col < 10; col++) {
            namta[row][col] = (row + 1) * (col + 1);
        }
    }
```

```
    for (row = 0; row < 10; row++) {
        for(col = 0; col < 10; col++) {
            printf("%d x %d = %d\n", (row + 1), (col + 1), namta[row][col]);
        }
        printf("\n");
    }

    return 0;
}
```
প্রোগ্রাম: ১১.২

সম্পূর্ণ আউটপুট স্ক্রিনে না-ও আটতে পারে, তাতে চিন্তার কিছু নেই।

এখন তোমাদের জন্য একটি সমস্যা। ওপরের প্রোগ্রামটি পরিবর্তন করো যাতে আউটপুট হিসেবে আমরা দেখতে পারি যে namta অ্যারেতে মোট কয়টি জোড় ও কয়টি বেজোড় সংখ্যা আছে। সেই সঙ্গে অ্যারেতে যতগুলো সংখ্যা আছে, তার যোগফলও বের করতে হবে। আশা করি, প্রোগ্রামটি ঠিকঠাক লিখতে পেরেছ। যদি কোনো সমস্যা হয়, তবে তুমি তোমার বন্ধুর সাহায্য নিতে পারো।
আচ্ছা, কেউ যদি বলে, সার্কভুক্ত সাতটি দেশের নাম একটি অ্যারেতে রাখতে, তাহলে কী করবে? একটি char type অ্যারেতে একটি দেশের নাম রাখা যায়। যেমন: char country[] = “Bangladesh”;। তাহলে সাতটি দেশের নাম রাখার জন্য আমাদের একটি 2-D অ্যারে ব্যবহার করতে হবে। এই অ্যারেতে মোট কয়টি রো থাকবে? সাতটি। কলাম কয়টি থাকবে? জানি না। আসলে একেক দেশের নামের দৈর্ঘ্য তো একেক রকম। তাই আমরা একটি কাজ করতে পারি, কলামে 100 টি ক্যারেক্টার রাখার ব্যবস্থা করতে পারি, কারণ সার্কের কোনো দেশের নামে তো 100 টির বেশি ক্যারেক্টার নেই, কম আছে।

```
#include <stdio.h>

int main()
```

```
{
    char saarc[7][100] = {"Bangladesh", "India", "Pakistan", "Sri Lanka",
"Nepal", "Bhutan", "Maldives"};

    int row;

    for (row = 0; row < 7; row++) {
        printf("%s\n", saarc[row]);
    }

    return 0;
}
```

প্রোগ্রাম: ১১.৩

প্রোগ্রামটি তোমার কম্পিউটারে লিখে রান করাও। এখন তোমরা বলো তো, saarc[3][3], saarc[0][5] ও saarc[5][0] – এই তিনটি ঘরে কী কী ক্যারেক্টার আছে? একটু পরে একটি প্রোগ্রাম লিখব, তার সঙ্গে তোমার উত্তর মিলিয়ে নেবে।
আমরা যদি ওই অ্যারের প্রতিটি ক্যারেক্টার আলাদাভাবে প্রিন্ট করতে চাই, তবে প্রোগ্রামটি এভাবে লিখতে পারি:

```
#include <stdio.h>
#include <string.h>

int main()
{
    char saarc[7][100] = {"Bangladesh", "India", "Pakistan", "Sri Lanka",
"Nepal", "Bhutan", "Maldives"};

    int row, col, name_length;
```

```
    for (row = 0; row < 7; row++) {
        name_length = strlen(saarc[row]);
        for(col = 0; col < name_length; col++) {
            printf("%c ", saarc[row][col]);
        }
        printf("\n");
    }

    return 0;
}
```

প্রোগ্রাম: ১১.৪

আবার যদি দেখতে চাই কোন ঘরে কী আছে, তাহলে রো এবং কলাম নম্বরসহ প্রিন্ট করতে পারি।

```
#include <stdio.h>
#include <string.h>

int main()
{
    char saarc[7][100] = {"Bangladesh", "India", "Pakistan", "Sri Lanka",
"Nepal", "Bhutan", "Maldives"};

    int row, col, name_length;

    for (row = 0; row < 7; row++) {
        name_length = strlen(saarc[row]);
        for(col = 0; col < name_length; col++) {
```

```
            printf("(%d, %d) = %c, ", row, col, saarc[row][col]);
        }
        printf("\n");
    }

    return 0;
}
```

প্রোগ্রাম: ১১.৫

এবারে নিচের ছকটি দেখো।

6	4	7	8	9
3	7	1	9	9
8	6	4	2	7
2	4	2	5	9
4	1	6	7	3

এখন 2-D অ্যারে ব্যবহার করে তোমাদের দুটি প্রোগ্রাম লিখতে হবে। প্রথম প্রোগ্রামটির কাজ হবে প্রতিটি রো-এর সংখ্যাগুলোর যোগফল বের করা আর দ্বিতীয় প্রোগ্রামের কাজ হবে প্রতিটি কলামের সংখ্যাগুলোর যোগফল বের করা।

প্রথম প্রোগ্রামের আউটপুট হবে এই রকম:

```
Sum of row 1: 34
Sum of row 2: 29
Sum of row 3: 27
```

```
Sum of row 4: 22
Sum of row 5: 21
```

দ্বিতীয় প্রোগ্রামের আউটপুট হবে এই রকম:

```
Sum of column 1: 23
Sum of column 2: 22
Sum of column 3: 20
Sum of column 4: 31
Sum of column 5: 37
```

তোমাদের অনেকেরই দ্বিতীয় প্রোগ্রামটি লিখতে একটু সময় লাগতে পারে, কিন্তু হতাশার কোন কারণ নেই। সময় লাগাই স্বাভাবিক। এবারে নিচের ছকটি দেখো। আগের ছকটির সঙ্গে-এর কোথায় যেন একটু মিল রয়েছে!

6	3	8	2	4
4	7	6	4	1
7	1	4	2	6
8	9	2	5	7
9	9	7	9	3

আসলে আগের ছকের রোগুলো নতুন ছকের কলাম, আর আগের ছকের কলামগুলো নতুন ছকের রো। যেমন আগের ছকের প্রথম রো-টি ছিল: 6, 4, 7, 8, 9। আর এই ছকের প্রথম কলাম হচ্ছে: 6, 4, 7, 8, 9। একইভাবে আগের ছকের দ্বিতীয় রো নতুন ছকের দ্বিতীয় কলামের সঙ্গে মেলে। এখন আমরা একটি প্রোগ্রাম লিখব, যেটি একটি 5 x 5 অ্যারেকে (অর্থাৎ 5 রো এবং 5 কলামবিশিষ্ট অ্যারে), আরেকটি 5 x 5 অ্যারেতে এমনভাবে কপি করবে, যাতে প্রথম অ্যারের রোগুলো হয় দ্বিতীয় অ্যারের কলাম আর প্রথম অ্যারের কলামগুলো হয় দ্বিতীয় অ্যারের রো। মানে ওপরের ছক দুটির মত আরকি। যেমন প্রথম অ্যারের প্রথম রো যদি হয়: 1, 2, 3, 4, 5 তাহলে দ্বিতীয় অ্যারের প্রথম কলাম হবে 1, 2, 3, 4, 5। তোমার কি বিষয়টি একটু জটিল মনে হচ্ছে? তাহলে কিছুক্ষণ বিশ্রাম নিয়ে

তারপর নিচের প্রোগ্রামটি কম্পাইল করো, রান করো, আউটপুট দেখো এবং কীভাবে কাজ করছে বোঝার চেষ্টা করো।

```
#include <stdio.h>
#include <string.h>

int main()
{
    int ara1[5][5] = {{1, 2, 3, 4, 5}, {10, 20, 30, 40, 50}, {100, 200, 300,
400, 500}, {1000, 2000, 3000, 4000, 5000}, {10000, 20000, 30000, 40000,
50000}};
    int ara2[5][5];

    int r, c;

    printf("Content of first array (ara1): \n");

    for (r = 0; r < 5; r++) {
        for(c = 0; c < 5; c++) {
            printf("%d  ", ara1[r][c]);
        }
        printf("\n");
    }

    printf("\n");

    // now start copy
    for (r = 0; r < 5; r++) {
        for(c = 0; c < 5; c++) {
```

```
            ara2[c][r] = ara1[r][c];
        }
    }

    printf("Content of second array (ara2): \n");

    for (r = 0; r < 5; r++) {
        for(c = 0; c < 5; c++) {
            printf("%d  ", ara2[r][c]);
        }
        printf("\n");
    }

    return 0;
}
```
প্রোগ্রাম: ১১.৬

তোমরা যদি এতক্ষণ 2-D অ্যারের সব উদাহরণ এবং যেসব প্রোগ্রাম নিজে লিখতে বলেছি, সেগুলো সব কম্পিউটারে রান করে থাকো এবং বুঝে থাকো (বুঝতে হলে অবশ্যই চিন্তা করতে হবে) তবে তোমাদের আর 2-D অ্যারে নিয়ে সমস্যা হওয়ার কথা নয়। অ্যারে কিন্তু 3-D, 4-D কিংবা আরও বেশি ডাইমেনশনের হতে পারে, তবে সেগুলো নিয়ে এই বইতে আর আলোচনা করব না।

অধ্যায় বারঃ বাইনারি সংখ্যা

আমরা তো দৈনন্দিন জীবনে নানা হিসাব-নিকাশের জন্য দশভিত্তিক (decimal) সংখ্যা পদ্ধতি ব্যবহার করি। কিন্তু কম্পিউটার ব্যবহার করে দুইভিত্তিক বা বাইনারি (binary) সংখ্যা পদ্ধতি। দশভিত্তিক সংখ্যা পদ্ধতিতে আছে মোট দশটি অঙ্ক 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 আর বাইনারিতে দুটি, 0 আর 1। আমরা এই অধ্যায়ে বাইনারি সংখ্যা পদ্ধতির কিছু মৌলিক জিনিস দেখব আর বাইনারি থেকে ডেসিমাল এবং ডেসিমাল থেকে বাইনারি সংখ্যায় রূপান্তর করা শিখব। ডেসিমালে আমরা গণনা করি এভাবে: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, … 19, 20, 21, …, 98, 99, 100, 101 …। দেখো, যখনই আমরা ডান দিকের ঘরে (এককের ঘরে) দশটি অঙ্ক ব্যবহার করে ফেলি, তখন তার বাঁয়ে দশকের ঘরের অঙ্কের মান এক বাড়াই (আর যদি না থাকে তাহলে 1 বসাই বা 0-এর সঙ্গে 1 যোগ করি আর কি, কারণ 9 আর 09 কিন্তু একই কথা, তাই 09-এর পরবর্তি সংখ্যা হচ্ছে 10), আবার দশকের ঘরে 0 থেকে 9 সব অঙ্ক ব্যবহার করে ফেলার পরে শতকের ঘরের অঙ্কের মান এক বাড়াই (আর যদি না থাকে তাহলে 1 বসাই বা 0-এর সঙ্গে 1 যোগ করি আর কি)। তেমনই বাইনারিতে আমরা গণনা করব এইভাবে: 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011 …। যেহেতু অঙ্ক মাত্র দুটি, তাই দুটি অঙ্কের ব্যবহার হয়ে গেলেই বাঁ দিকের ঘরে এক বসাতে হয় বা 0-এর সঙ্গে 1 যোগ করতে হয় (বাঁ দিকে তো আমরা ইচ্ছামত শূন্য বসাতে পারি)।

বাইনারি সিস্টেমে অবশ্য আমরা এককের ঘর, দশকের ঘর, শতকের ঘর, সহস্রের ঘর না বলে বলব একের ঘর, দুইয়ের ঘর, চারের ঘর, আটের ঘর। কেন বল তো? একটু চিন্তা করো।

ডেসিমালে যেমন 10 লিখতে দুটি অঙ্ক লাগে, 100 লিখতে তিনটি, 1000 লিখতে চারটি, তেমনই বাইনারিতে দুই লিখতে দুটি (10), চার লিখতে তিনটি (100), আট লিখতে চারটি (1000), ষোল লিখতে পাঁচটি (10000) অঙ্ক ব্যবহার করতে হয়। ডেসিমালে ডান দিকের প্রথম অঙ্ক ($10^0 = 1$) হচ্ছে এককের ঘর, দ্বিতীয় অঙ্ক ($10^1 = 10$) হচ্ছে দশকের ঘর, তৃতীয় অঙ্ক ($10^2 = 100$) হচ্ছে শতকের ঘর, তেমনই বাইনারিতে ডানদিকের প্রথম অঙ্ক ($2^0 = 1$) হচ্ছে একের ঘর, পরের অঙ্ক ($2^1 = 2$) হচ্ছে দুইয়ের ঘর, তারপর ($2^2 = 4$) হচ্ছে চারের ঘর, এই রকম।

দশভিত্তিক সংখ্যায় যেমন যোগ, বিয়োগ, গুণ, ভাগ করা যায়, তেমনই বাইনারিতে করা যায়। আসলে যোগ করতে পারলে কিন্তু বাকি কাজ করা কোনো ব্যাপার নয়। আবার বাইনারিতে ভগ্নাংশের ব্যাপার আছে, তবে আমি কেবল পূর্ণসংখ্যা নিয়েই আলোচনা করব।

যোগের ক্ষেত্রে মূল হিসাবগুলো হচ্ছে:
0 + 0 = 0,
0 + 1 = 1,
1 + 0 = 1,
1 + 1 = 10।

ডেসিমালের মতোই হিসাব, 1 + 1 এর ক্ষেত্রে দেখো, দুইয়ের (10) শূন্য এল প্রথমে, হাতে থাকে এক, সেটি পরে লিখলাম।

101 + 101 = কত?

প্রথমে একের ঘরের যোগ, 1 + 1 = 10। তাই যোগফলের একের ঘরে বসবে 0 আর হাতে থাকল 1 (carry)। এবারে দুইয়ের ঘরে, 0 + 0 = 0, এখন এই 0-এর সঙ্গে হাতের 1 যোগ করতে হবে। তাহলে যোগফলের দুইয়ের ঘরে বসবে 1। এবারে চারের ঘরের যোগ করলে পাই, 1 + 1 = 10। হাতে কিছু নেই (কোনো carry নেই)। সুতরাং চারের ঘরে বসবে 0 আর 1 বসবে আটের ঘরে। যোগফল: 1010। এবারে বলো 1011 + 1011 = কত? যোগ করে যদি দেখো যোগফল 10110 হয়নি, তাহলে তুমি যোগে কোথাও ভুল করেছ।

বিয়োগের ক্ষেত্রেও ডেসিমালের মতো হিসাব হবে।
0 – 0 = 0,
1 – 0 = 1,
1 – 1 = 0,
0 – 1 = 1।
শেষেরটা খেয়াল করো, 23 – 15 করার সময় আমরা কী করি? তখন 3-এর বাঁয়ে একটি কাল্পনিক 1 ধরে নিই (বা 1 ধার করি), তারপর 13 – 5 = 8 লেখি। আর যেই একটি ধার করলাম, সেটি পরের ঘরে 1-এর সঙ্গে যোগ করে দিই। তেমনই বাইনারিতে 0 – 1 করতে গেলে 0-এর বাঁয়ে একটি এক ধরব, তখন সংখ্যাটি হবে 10 (দুই), এই দুই থেকে এক বাদ দিলে এক থাকবে। পরের ঘরে একটি এক যোগ করতে হবে (যেই সংখ্যাটি বিয়োগ হচ্ছে তার সঙ্গে)।

110 – 101 = কত?

একের ঘরে 0 থেকে 1 বাদ দিলে থাকে 1, এখানে 1 ধার করতে হয়েছে। তাই 101-এর দুইয়ের ঘরে সেটি যোগ করে দেব। তাহলে দুইয়ের ঘরে 1 – 1 = 0, চারের ঘরে 1 – 1 = 0। তাই বিয়োগফল হবে: 001 বা 1। যোগ-বিয়োগ পারলে গুণ-ভাগ না পারার কারণ নেই। ডেসিমালের মতোই নিয়ম।

আবার কোনো ডেসিমাল সংখ্যাকে আমরা নির্দিষ্ট অঙ্ক x 10 ^ওই অঙ্কের অবস্থান^-এর যোগফল হিসেবে যেমন লিখতে পারি, বাইনারি সংখ্যাকেও নির্দিষ্ট অঙ্ক x 2 ^ওই অঙ্কের অবস্থান^-এর যোগফল হিসেবে লেখা যায়। যেমন: $1903 = 1 \times 10^3 + 9 \times 10^2 + 0 \times 10^1 + 3 \times 10^0$।

বাইনারি: $10110 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$। ইংরেজিতে একে বলে Exponential Expression।

এখন কোনো বাইনারি সংখ্যার মান যদি ডেসিমালে বের করতে চাই, তবে প্রথমে বাইনারি সংখ্যাটিকে এক্সপোনেনশিয়াল এক্সপ্রেশন আকারে লিখতে হবে। তারপর গুণফলগুলো ডেসিমালে হিসাব করতে হবে। নিচের উদাহরণটি দেখো:

$10110 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 1 \times 16 + 0 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1$
$= 16 + 0 + 4 + 2 + 0 = 22$

অর্থাৎ বাইনারি 10110 = ডেসিমাল 22। আমরা অনেকক্ষণ কোনো প্রোগ্রামিং করছি না, চলো বাইনারি সংখ্যার ডেসিমাল মান বের করার একটি প্রোগ্রাম লিখে ফেলি। পদ্ধতি তো জানা হয়ে গেছে। এখন গুরুত্বপূর্ণ প্রশ্ন হচ্ছে, বাইনারি সংখ্যা রিড করব কী দিয়ে? আমরা স্ট্রিং ব্যবহার করতে পারি।

```
char binary[] = “10110”;

int len = 5; // স্ট্রিংয়ের দৈর্ঘ্য 5।
int decimal = 0; // এখনো কোনো হিসাব করিনি, তাই ধরলাম ডেসিমাল মান 0।
```

এবারে আমরা একটি লুপের সাহায্যে বাইনারি সংখ্যার প্রতিটি অঙ্কের সঙ্গে $2^{\text{ওই অঙ্কের অবস্থান}}$ গুণ করে সেটি ডেসিমালের সঙ্গে যোগ করে দেব। প্রথম ক্যারেক্টার অর্থাৎ binary[0]তে তো '1' আছে,-এর অবস্থান কত বলো তো?-এর অবস্থান হচ্ছে 4। তারপরের অঙ্কের বেলায় অবস্থানের মান এক কমবে, এভাবে একেবারে শেষের অঙ্কের বেলায় অবস্থান হবে 0।

```
int position = 4;

int indx;

for(indx = 0; indx < len; indx++) {
    decimal = decimal + pow(2, position);
    position--;
}
```

লুপ থেকে বের হলে আমরা সম্পূর্ণ বাইনারি সংখ্যার ডেসিমাল মান পেয়ে যাব। এখানে দেখো, আমি pow ফাংশন ব্যবহার করেছি। এটির কাজ বলা আছে math.h হেডার ফাইলে। a^b-এর মান বের করার জন্য pow(a, b) বলে দিলেই হয়। তাহলে আমাদের পুরো প্রোগ্রামটি দাঁড়াচ্ছে এই রকম:

```
#include <stdio.h>
#include <string.h>
#include <math.h>

int main()
{
    char binary[65];
    int len, decimal, power, i;
```

```
    printf("Enter the binary number: ");
    scanf("%s", binary);

    decimal = 0;
    len = strlen(binary);
    power = len - 1;

    for(i = 0; i < len; i++) {
        decimal += pow(2, power);
        power--;
    }

    printf("Decimal value is %d\n", decimal);

    return 0;
}
```
প্রোগ্রাম: ১২.১

প্রোগ্রাম কম্পাইল করে রান করো। ইনপুট যদি 10110 দাও, তাহলে আউটপুট কত আসে? আউটপুট আসে 31 কিন্তু আউটপুট তো আসা উচিত 22। তাহলে আমরা কোথাও ভুল করেছি। তোমরা নিজে নিজে ভুলটি বের করার চেষ্টা করো।

আমাদের তো আসলে pow(2, position)কে বাইনারি সংখ্যার ওই position-এর অঙ্কটি দিয়ে গুণ করার কথা, সেটি আমরা করতে ভুলে গেছি। অর্থাৎ আমাদের লিখতে হবে:

decimal += binary[i] * pow(2, power);

একটি ব্যাপার খেয়াল করছো তো? 10110-এর একের ঘরের অঙ্কটি আমাদের অ্যারের শেষ ক্যারেক্টার, আর ষোলোর ঘরের অঙ্কটি

হচ্ছে অ্যারের প্রথম ক্যারেক্টার। অ্যারেতে সংখ্যাটি আছে এইভাবে: ['1', '0', '1', '1', '0']। তাই binary[0]-এর সঙ্গে গুণ হবে pow(2, 4), binary[1]-এর সঙ্গে গুণ হবে pow(2, 3), ..., এভাবে binary[4]-এর সঙ্গে গুণ হবে pow(2, 0)। এখন প্রোগ্রামটি ঠিক করে নিয়ে তারপর চালাও। ইনপুট 10110-এর জন্য কী আউটপুট?

আমি তো আউটপুট দেখতে পাচ্ছি Decimal value is 1510। ভুলটি কোথায় হলো? সব তো ঠিকই করলাম। তোমরা আবার বিরক্ত হয়ে যাচ্ছ না তো? টেস্ট ক্রিকেট খেলার সময় যেমন ধৈর্যের প্রয়োজন, প্রোগ্রামিংও তেমনই ধৈর্যের খেলা।

ভুলটি যে decimal += binary[i] * pow(2, power); স্টেটমেন্টে হয়েছে তাতে কোনো সন্দেহ নেই। কারণ আমরা এখানেই একটু পরিবর্তন করেছি। লক্ষ করো, binary[i]-এর মান হয় '0' বা '1' (মানে ক্যারেক্টার '0' বা ক্যারেক্টার '1')। এখন কম্পিউটার '0' বলতে বোঝে 48 আর '1' বলতে বোঝে 49। ঝামেলাটা এখানেই হয়েছে। এখন এই '0'কে 0 আর '1'কে 1 বোঝাব কীভাবে?

'0' – '0' = 48 – 48 = 0।
'1' – '0' = 49 – 48 = 1।

বুদ্ধিটা দারুণ না? আমরা binary[i] না লিখে (binary[i] – '0') লিখলেই ঝামেলা শেষ। এবারে প্রোগ্রাম ঠিকঠাক কাজ করবে (যদি না তুমি নতুন কোনো ভুল করে থাকো)।

এবারে আমরা দেখব ডেসিমাল থেকে বাইনারিতে রূপান্তর। একটি উদাহরণের সাহায্যে পদ্ধতিটা দেখাই। ধরো 95 কে বাইনারিতে রূপান্তর করতে হবে। এখন আমাদের বের করতে হবে n-এর সর্বোচ্চ মান, যেখানে 2^n <= 95। দুইয়ের পাওয়ারগুলো হচ্ছে 1, 2, 4, 8, 16, 32, 64, 128, ...। এখানে আমরা দেখতে পাচ্ছি 64 < 95 বা 2^6 < 95। তাহলে n-এর মান 6। আর আমাদের বাইনারি সংখ্যাটি হবে সাত অঙ্কের (0 থেকে 6 মোট সাতটি অঙ্ক)। যেহেতু 64 < 95, তাই এই সংখ্যাটি নেওয়া যায়। তাহলে চৌষট্টির ঘরে (বাঁ থেকে প্রথম বা ডান থেকে সপ্তম) হবে 1 (1xxxxxx)। এখন n-এর মান 1 কমাই। 64 + 2^5 = 64 + 32 = 96, যা কিনা 95-এর চেয়ে বড়। তাই একে নেওয়া যাবে না। অতএব বত্রিশের ঘরে 0 বসাই (10xxxxx)। এবারে n-এর মান আবার এক কমাই, n-এর মান এখন 4। 64 + 2^4 = 64 + 16 = 80 < 95। সুতরাং ষোলোর ঘরে হবে 1 (101xxxx)। এখন n-এর মান এক কমাই, n = 3। 80 + 2^3 = 80 + 8 = 88 < 95। তাই আটের ঘরেও 1 বসবে (1011xxx)। এরপর একইভাবে, n = 2-এর জন্য 88 + 2^2 = 88 +

4 = 92 < 95। চারের ঘরেও 1 বসবে (10111xx)। তারপর n = 1, 92 + 2^1 = 92 + 2 = 94 < 95। দুইয়ের ঘরেও 1 (101111x)। এখন n = 0, 94 + 2^0 = 94 + 1 = 95। তাই একের ঘরেও 1। সুতরাং বাইনারি সংখ্যাটি হচ্ছে 1011111। তোমরা এখন এই পদ্ধতিতে কোনো দশভিত্তিক সংখ্যাকে বাইনারিতে রূপান্তর করার প্রোগ্রাম লিখে ফেলো এবং বিভিন্ন মান দিয়ে পরীক্ষা করে দেখো।

এখন একই কাজ আমরা একটু অন্যভাবে করব। নিচের টেবিলটি দেখো:

	ভাগফল	ভাগশেষ
95/ 2	47	1
47/2	23	1
23/2	11	1
11/2	5	1
5/2	2	1
2/2	1	0
1 /2	0	1

এবারে ভাগশেষ কলামের অঙ্কগুলো শেষ থেকে প্রথম ক্রমে লেখলেই আমরা বাইনারি নম্বরটা পেয়ে যাব: 1011111। আর ভাগের কাজটি আমরা ততক্ষণ করব যতক্ষণ না ভাগফল 0 পাচ্ছি।

এই পদ্ধতিতেও তোমরা ডেসিমাল থেকে বাইনারি রূপান্তরের জন্য একটি কোড লিখে ফেলো। রূপান্তরের কোডটি main ফাংশনে না করে আলাদা একটি ফাংশনে করবে।

তুমি যদি এই পিডিএফ বইটি কারো কাছ থেকে কপি করে থাকো, কিংবা ইন্টারনেট থেকে ফ্রি ডাউনলোড করে থাকো, তুমি চাইলে লেখকের পরিশ্রমের মূল্য পরিশোধ করতে পারো। এই পিডিএফ বইটির দাম একেবারেই হাতের নাগালে। মাত্র **25** টাকা বিকাশ-এর মাধ্যমে পাঠিয়ে দাও এই নাম্বারেঃ **01622624182**, তোমরা যত বেশি বই কিনবে, সেটি হার্ডকপিই হোক, কিংবা ইবুক (যেমন এই পিডিএফ), লেখকরা তত বেশি উৎসাহ পাবেন এবং আরো বই লেখার ব্যাপারে আগ্রহী হবেন।

বিকাশের মাধ্যমে পেমেন্ট করার নিয়মঃ

01. Go to your bKash Mobile Menu by dialing *247#
02. Choose “Payment”
03. Enter the Merchant bKash Account Number you want to pay to (01622624182)
04. Enter the amount you want to pay (25)
05. Enter a reference against your payment (cpbook pdf)
06. Enter the Counter Number (enter 0)
07. Now enter your bKash Mobile Menu PIN to confirm

১) প্রথমে *247# নাম্বারে ডায়াল করে বিকাশ মেনু আনতে হবে।
২) তারপরে 'পেমেন্ট' নির্বাচন করতে হবে।
৩) এবারে মার্চেন্ট বিকাশ একাউন্ট নাম্বার হিসেবে 01622624182 দিতে হবে।
৪) তারপরে টাকার পরিমাণ দিতে হবে (এই পিডিএফ বইয়ের জন্য ২৫ টাকা)
৫) রেফারেন্স হিসেবে cpbook pdf লিখে দিতে হবে।
৬) কাউন্টার নাম্বার ০ দিবে।
৭) এবারে তোমার বিকাশ মোবাইল মেনু পিন (PIN) দিতে হবে।

তোমার বিকাশ একাউন্ট না থাকলে একটি খুলে নিতে পারো (এটি খুলতে কোনো টাকা লাগে না) কিংবা অন্য কারো বিকাশ একাউন্ট ব্যবহার করতে পারো।

dimik

অনলাইনে সফটওয়্যার, ওয়েব ও মোবাইল অ্যাপ্লিকেশন তৈরির কোর্স, সম্পূর্ণ বাংলায় নিয়ে আসছে দ্বিমিক কম্পিউটিং স্কুল: http://dimikcomputing.com

অধ্যায় তেরঃ কিছু প্রোগ্রামিং সমস্যা

এই অধ্যায়ে আমরা কয়েকটি সহজ সমস্যা দেখব ও সমাধানের চেষ্টা করব।
আমাদের প্রথম সমস্যা হচ্ছে, বিভিন্ন ধরনের আকৃতি তৈরি করা। নিচের ছবিগুলো দেখো।

```
ccccccc
cccccc
ccccc
cccc
ccc
cc
c
cc
ccc
cccc
ccccc
cccccc
ccccccc
```

ছবি ১৩.১

```
ccccccccccc
 ccccccccc
  ccccccc
   ccccc
    ccc
     c
    ccc
   ccccc
  ccccccc
 ccccccccc
ccccccccccc
```

ছবি ১৩.২

```
ccccccc
 cccccc
  ccccc
   cccc
    ccc
     cc
      c
     cc
    ccc
   cccc
  ccccc
 cccccc
ccccccc
```

ছবি ১৩.৩

```
c                  c
cc                cc
ccc              ccc
cccc            cccc
ccccc          ccccc
cccccc        cccccc
ccccccc      ccccccc
cccccccc    cccccccc
ccccccccc  ccccccccc
cccccccccccccccccccc
```

ছবি ১৩.৪

তোমাদের চারটি প্রোগ্রাম লিখতে হবে এই চার ধরনের আকৃতি তৈরি করার জন্য। কেবল printf ফাংশন ব্যবহার করলেই হবে না, লুপ ব্যবহার করতে হবে। তাহলে লুপ বা নেস্টেড লুপ, এবং 'c' ও ' ' (স্পেস ক্যারেক্টার) প্রিন্ট করে তোমরা প্রোগ্রামগুলো লিখে ফেলতে

পারো। আরও খেলাধুলা করার ইচ্ছা হলে আরও নানান রকম আকৃতি তৈরির চেষ্টা করতে পার।

প্যালিনড্রোম (palindrome) কী জিনিস, তোমরা জান? কোনো শব্দকে উল্টাভাবে (মানে শেষ থেকে শুরু) লিখলে যদি সেটি আর নতুন শব্দটি একই রকম হয় তবে সেটি একটি প্যালিনড্রোম। যেমন: madam। এটিকে শেষ থেকে শুরু পর্যন্ত লিখলেও madam হবে। এখন একটি প্রোগ্রাম লিখব যেটিতে কোনো শব্দ ইনপুট দিলে সেটি প্যালিনড্রোম কি না বলে দেবে।

এজন্য আমরা কী করতে পারি? প্রথমে শব্দটি স্ট্রিং হিসেবে একটি অ্যারেতে ইনপুট নেব। তারপর আরেকটি অ্যারেতে সেটি উল্টাভাবে রাখব। তারপর যদি দুটি একই স্ট্রিং হয়, তবে সেটি প্যালিনড্রোম। তাহলে প্রোগ্রামটি লিখে ফেলি:

```
#include <stdio.h>
#include <string.h>

int main()
{
    char word[80], reverse_word[80];
    int i, j, len;

    scanf("%s", word);

    len = strlen(word);

    for(i = 0, j = len - 1; i < len; i++, j--) {
        reverse_word[i] = word[j];
    }
    reverse_word[i] = '\0';

    printf("%s\n", reverse_word);
```

```
    if (0 == strcmp(word, reverse_word)) {
        printf("%s is a palindrome.\n", word);
    }
    else {
        printf("%s is not a palindrome.\n", word);
    }

    return 0;
}
```

প্রোগ্রাম: ১৩.১

কী মজা! আমি প্রোগ্রামটি লিখে দিলাম। তবে আমি এখানে বেশ কিছু বোকামি করেছি, যার মধ্যে অন্যতম হচ্ছে একটি অতিরিক্ত অ্যারে ব্যবহার করা। সুতরাং তোমাদের এখন প্রোগ্রামটি এমনভাবে লিখতে হবে, যাতে কেবল একটি অ্যারে ব্যবহার করেই কাজ হয়। আর তখন strcmp ফাংশনটিরও দরকার হবে না। প্রোগ্রামটি লিখতে সময় বেশি লাগতে পারে, লাগুক, অসুবিধা নেই। তবে লিখতে হবে ঠিকঠাক, এটিই হলো কথা।

তোমরা তো ফ্যাক্টরিয়াল (factorial) জিনিসটির সঙ্গে পরিচিত? এটি একটি গাণিতিক অপারেশন যা কোনো ধনাত্মক পূর্ণসংখ্যার ক্ষেত্রে ব্যবহার করা যায়। n একটি ধনাত্মক পূর্ণ সংখ্যা হলে-এর ফ্যাক্টরিয়ালকে প্রকাশ করা হয় n! দিয়ে এবং n! = n * (n – 1) * (n – 2) * … * 3 * 2 * 1। যেমন 4! = 4 * 3 * 2 * 1 = 24। আবার 6! = 6 * 5 * 4 * 3 * 2 * 1 = 720। 1! = 1। 0! = 1 (0-এর ক্ষেত্রে ব্যতিক্রমটি লক্ষ করো, কিছু বিশেষ সুবিধার জন্য 0-এর ফ্যাক্টরিয়ালের মান 1 ধরা হয়)। এখন তোমরা কোনো ধনাত্মক পূর্ণসংখ্যার ফ্যাক্টরিয়াল বের করার প্রোগ্রামটি লিখে ফেলো। সহজ প্রোগ্রাম, একটি লুপ ব্যবহার করেই করা যায়। এখন বিভিন্ন সংখ্যা দিয়ে প্রোগ্রামটি টেস্ট করে দেখো ফ্যাক্টরিয়াল ঠিকঠাক বের করতে পারে কি না। প্রোগ্রামে তুমি যদি ডাটা টাইপ int ব্যবহার করে থাক তবে 12-এর চেয়ে বড় কোনো পূর্ণ সংখ্যার ফ্যাক্টরিয়ালের মান ঠিকমতো দেখাবে না (ক্যালকুলেটরে করে মিলিয়ে দেখতে পারো)। কারণ হচ্ছে 12-এর চেয়ে বড় কোনো পূর্ণ সংখ্যার জন্য সেই সংখ্যার ফ্যাক্টরিয়ালের মান রেঞ্জের বাইরে চলে যায়।

এখন তোমাদের একটি মজার সমস্যা সমাধান করতে হবে। কোনো পূর্ণসংখ্যা n (যেখানে 1 < n < 100, মানে n-এর মান 2 থেকে 99 পর্যন্ত হতে পারে)-এর ফ্যাক্টরিয়ালকে মৌলিক সংখ্যার গুণফল হিসেবে প্রকাশ করলে কোন মৌলিক সংখ্যা কতবার আছে সেটি বের করতে হবে। যেমন, আমরা জানি, 5! = 120 = 2 * 2 * 2 * 3 * 5। এখানে 2 আছে 3 বার, 3 আছে 1 বার আর 5 আছে 1 বার। তাই ইনপুট 5 হলে আউটপুট হবে: 5! = (2, 3), (3, 1), (5, 1)। তোমরা কি একটি ব্যাপার বুঝতে পারছ যে শুরুতে n-এর ফ্যাক্টরিয়ালের মান বের করে তারপর মৌলিক উৎপাদকে ভাঙতে গেলে একটি ঝামেলা হয়ে যাবে? কারণ n-এর মান সর্বোচ্চ হতে পারে 99 আর ইন্টিজারে তো 12-এর চেয়ে বড় কোনো সংখ্যার ফ্যাক্টরিয়ালের মান রাখা যায় না। আসলে এই প্রোগ্রামের জন্য n!-এর মান বের করার কোনো দরকার নেই। শুধু একটু গাণিতিক যুক্তি-বুদ্ধি খাটাও। আর 2 থেকে 99 পর্যন্ত মৌলিক সংখ্যাগুলো একটি অ্যারেতে রেখে নাও। প্রোগ্রামটি ঠিকভাবে করতে তোমাদের অনেকেরই দু-তিন দিন সময় লেগে যেতে পারে, এতে হতাশ হওয়ার কিছু নেই।

এখন আমরা একটি প্রোগ্রাম লিখব। যার উদ্দেশ্য হবে কোনো অ্যারেতে কিছু সংখ্যা থাকলে সেগুলোকে ছোট থেকে বড় ক্রমে সাজানো। যেমন, কোনো অ্যারে যদি এমন হয়: int ara[] = {3, 1, 5, 2, 4}, তবে আমাদের প্রোগ্রাম সেই অ্যারের সংখ্যাগুলো এমনভাবে সাজাবে, যাতে ara[] = {1, 2, 3, 4, 5} হয়। প্রোগ্রামটি একটু পরে লিখব, তার আগে ঠিক করে নেই যে সেটি কীভাবে কাজ করবে। তোমার কাছে পাঁচটি সংখ্যা আছে: 3, 1, 5, 2, 4। ছোট থেকে বড় ক্রমে সাজাতে হবে। তুমি প্রথমে কী করবে? প্রথমে সবচেয়ে ছোট সংখ্যাটি খুঁজে বের করে তাকে শুরুতে লিখবে: 1। তখন বাকি থাকে চারটি সংখ্যা: 3, 5, 2, 4। এখন এই চারটির মধ্যে সবচেয়ে ছোট সংখ্যাটি 1-এর পরে লিখবে: 1, 2। বাকি রইল 3, 5, 4। এদের মধ্যে সবচেয়ে ছোট 3। তাই তুমি লিখবে : 1, 2, 3। এখন বাকি 5, 4। এই দুটি সংখ্যার মধ্যে সবচেয়ে ছোট 4। সেটি তুমি 3-এর পরে লিখবে: 1, 2, 3, 4। এখন বাকি একটি সংখ্যা, 5। সেটি তুমি 4-এর পরে লিখবে। 1, 2, 3, 4, 5। তোমার সাজানোর কাজ হয়ে গেল। একে সর্টিং (sorting) বলে। বিভিন্ন উপায়ে এটি করা যায়। তবে আমরা একটি সহজ-সরল উপায়ে করলাম। এখন প্রোগ্রামটি লিখব কীভাবে?

প্রথমে একটি অ্যারেতে সংখ্যাগুলো রাখো: int ara1[] = {3, 1, 5, 2, 4};
এখন আরেকটি অ্যারে নাও: int ara2[5]; অ্যারেটি এখনো খালি। তাই একটি ভেরিয়েবলে ইনডেক্স 0 লিখে রাখো।
int index_2 = 0;
এখন একটি একটি করে ara2 তে সংখ্যাগুলো রাখতে হবে। তার জন্য একটি লুপ দরকার।
for(index_2 = 0; index_2 < 5; index_2++) // মানে 0 থেকে 4 পর্যন্ত প্রতিটি ঘরে আমরা সংখ্যা বসাব।

এই লুপের ভেতরে আরেকটি লুপ দরকার যেটি দিয়ে আমরা ara1-এর সবচেয়ে ছোট সংখ্যা খুঁজে বের করব।
minimum = 100000; // এমন একটি বড় সংখ্যা অ্যাসাইন করলাম যেটি ara1-এর যেকোনো সংখ্যার চেয়ে বড়।
for (i = 0; i < 5; i++) {
    if (ara1[i] < minimum) {
        minimum = ara1[i];
    }
}
এখন ara1-এর ক্ষুদ্রতম সংখ্যাটি minimum এ চলে এল। সেটি এখন ara2 তে রাখি:
ara2[index_2] = minimum।

সবশেষে ara2-এর সব সংখ্যা প্রিন্ট করে দেখব।

এবারে চলো, পুরো প্রোগ্রামটি লিখে কম্পাইল ও রান করে দেখি আউটপুট কী আসে।

```
#include <stdio.h>

int main()
{
    int ara1[] = {3, 1, 5, 2, 4};
    int ara2[5];

    int i, minimum, index_2;

    for (index_2 = 0; index_2 < 5; index_2++) {
        minimum = 10000;
        for (i = 0; i < 5; i++) {
```

```
            if (ara1[i] < minimum) {
                minimum = ara1[i];
            }
        }
        ara2[index_2] = minimum;
    }

    for (i = 0; i < 5; i++) {
        printf("%d\n", ara2[i]);
    }

    return 0;
}
```
প্রোগ্রাম: ১১.২

কী সুন্দর প্রোগ্রাম! আউটপুট কী? আউটপুটও খুব সুন্দর, একে একে পাঁচটি 1।

```
1
1
1
1
1
```

কিন্তু আমরা তো এমন আউটপুট চাইনি। কোথাও গোলমাল হয়েছে। এখন আমার কোডে দেখো তো কোনো ভুল বের করা যায় কি না।

একটি ঝামেলা হয়েছে। ভেতরের লুপে (যেখানে সবচেয়ে ছোট সংখ্যা বের করা হয়) কিন্তু সব সময়ই minimum-এর মান 1 আসবে, কারণ 1 হচ্ছে ওই পাঁচটির মধ্যে সবচেয়ে ছোট সংখ্যা। এজন্য দ্বিতীয় অ্যারেতে পাঁচটি সংখ্যাই 1 হয়ে যাচ্ছে। তাই আমরা যখন

minimum বের করব, তখন অ্যারের যেই ঘরে সবচেয়ে ছোট সংখ্যা পাব সেই ঘরের মান একটি অনেক বড় সংখ্যা দিয়ে দেব। এজন্য একটি ভেরিয়েবল রাখি minimum_index। আর লুপটি এখন এমন হবে:

```
minimum = 10000;
for (i = 0; i < 5; i++) {
   if (ara1[i] < minimum) {
         minimum = ara1[i];
         minimum_index = i;
   }
}
```

এখন minimum-এর মান আমরা পেয়ে গেছি এবং সেই সঙ্গে এটিও জানি যে এটি আসলে আছে ara1[minimum_index] ঘরে।

```
ara1[minimum_index] = 10000;
```

তাহলে প্রোগ্রামটি ঠিক করে আবার চালাই:

```
#include <stdio.h>

int main()
{
    int ara1[] = {3, 1, 5, 2, 4};
    int ara2[5];

    int i, minimum, index_2, minimum_index;

    for (index_2 = 0; index_2 < 5; index_2++) {
        minimum = 10000;
        for (i = 0; i < 5; i++) {
```

```
            if (ara1[i] < minimum) {
                minimum = ara1[i];
                minimum_index = i;
            }
        }
        ara1[minimum_index] = 10000;
        ara2[index_2] = minimum;
    }

    for (i = 0; i < 5; i++) {
        printf("%d\n", ara2[i]);
    }

    return 0;
}
```
প্রোগ্রাম: ১১.৪

এখন প্রোগ্রামটি আউটপুট ঠিকঠাক দেখাবে। আচ্ছা, সব কাজই তো আমি করে দিলাম। তোমাদের কাজটি কী? তোমাদের কাজ হবে প্রোগ্রামটি এমনভাবে লেখা যাতে দ্বিতীয় অ্যারের প্রয়োজন না হয়। শুরুতে যে অ্যারেটি আছে তার ভেতরেই সর্টিং করতে হবে। এজন্য সবচেয়ে ছোট সংখ্যাটি অ্যারের প্রথম ঘরে নিয়ে আসো আর যে ঘর থেকে সবচেয়ে ছোট সংখ্যা পেয়েছ সেখানে প্রথম ঘরের সংখ্যাটি রাখো। এখন তোমার অ্যারের প্রথম ঘরে আছে সবচেয়ে ছোট সংখ্যা। এবারে বাকি চারটি ঘরের মধ্যে সবচেয়ে ছোট সংখ্যাটি অ্যারের দ্বিতীয় ঘরে রাখো এবং যে ঘর থেকে ওই সংখ্যাটি পেয়েছ সেখানে দ্বিতীয় ঘরের সংখ্যাটি রাখো। আর কিছু বলা যাবে না।

রোবট নিয়ে এখন আমরা একটি প্রোগ্রাম লিখব। কোনো একটি N x N গ্রিডে একটি রোবট আছে। শুরুতে তার একটি অবস্থান আছে। আমরা সেটিকে কিছু কমান্ড দেব, এক ঘর ডানে, বাঁয়ে, ওপরে ও নিচে যাওয়ার কমান্ড।

(0, 0)	(0, 1)	(0, 2)	(0, 3)	(0, 4)	(0, 5)	(0, 6)	(0, 7)	(0, 8)
(1, 0)		(1, 2)						
(2, 0)	(2, 1)	**R** (2, 2)	(2, 3)					
(3, 0)		(3, 2)						
(4, 0)								
(5, 0)								
(6, 0)								
(7, 0)								
(8, 0)								(8, 8)

গ্রিডটি দেখো। ওপরের একেবারে বাঁ দিকের ঘর হচ্ছে (0, 0)। ওপরের একেবারে ডানদিকের ঘর হচ্ছে (0, 8)। নিচের একেবারে বাঁ

দিকের ঘর হচ্ছে (8, 0)। নিচের একেবারে ডান দিকের ঘর হচ্ছে (8, 8)। ধরা যাক, এই মুহূর্তে রোবটটি আছে (2, 2) ঘরে। এক ঘর ওপরে যেতে বললে সে যাবে (1, 2) ঘরে। নিচে যেতে বললে যাবে (3, 2) ঘরে। ডানে আর বাঁয়ে যেতে বললে যথাক্রমে (2, 3) ও (2, 1) ঘরে যাবে। কমান্ডগুলো হচ্ছে U (up), D (down), L (left), R (right), S (stop)। এখন তোমাকে যদি শুরুর অবস্থান আর কমান্ডগুলো বলে দিই, তাহলে রোবটের শেষ অবস্থান (stop করার পর অবস্থান) বের করতে হবে।

তোমরা কি প্রোগ্রামটি নিজে লেখার জন্য কিছুক্ষণ চেষ্টা করবে?

তোমরা নিশ্চয়ই বুঝতে পারছ যে একটি 2-D অ্যারে দরকার হবে এই প্রোগ্রামে। আসলে কিন্তু এখানে অ্যারের কোনোই দরকার নেই। এটি সাধারণ যোগ-বিয়োগের প্রোগ্রাম। মনে করি, শুরুর অবস্থান হচ্ছে (x, y)। এখন U কমান্ড দিলে একঘর ওপরে যাবে, তখন x-এর মান এক কমে যাবে, y-এর মানের কোনো পরিবর্তন হবে না। D কমান্ড দিলে এক ঘর নিচে যাবে, তখন x-এর মান এক বেড়ে যাবে, y-এর মানের কোনো পরিবর্তন হবে না। R কমান্ড দিলে y-এর মান এক বাড়বে, x-এর মান অপরিবর্তিত থাকবে। L কমান্ড দিলে y-এর মান এক কমবে, x-এর মান অপরিবর্তিত থাকবে। তাহলে আমাদের পুরো প্রোগ্রামটি দাঁড়াবে এই রকম:

```
#include <stdio.h>

int main()
{
    int x, y;
    char c;

    printf("Please enter the initial position: ");
    scanf("%d %d", &x, &y);

    while (1) {
        scanf("%c", &c);
```

```
        if (c == 'S') {
            break;
        }
        else if (c == 'U') {
            x--;
        }
        else if (c == 'D') {
            x++;
        }
        else if (c == 'R') {
            y++;
        }
        else if (c == 'L') {
            y--;
        }
    }

    printf("Final position of the robot is: %d, %d\n", x, y);

    return 0;
}
```

প্রোগ্রাম: ১১.৫

আউটপুট কী হবে সেটি নির্ভর করবে তোমার ইনপুটের ওপর। যেমন:

Please enter the initial position: 2 2

D

R

D
R
S
Final position of the robot is: 4, 4

বেশ সহজ সরল প্রোগ্রাম। কিন্তু এখন যদি বলি যে গ্রিডে কিছু কিছু ঘরে যাওয়া নিষেধ এবং ওই ঘরগুলোতে যেতে বললে রোবটটি কিছুই করবে না (অর্থাৎ ওই কমান্ডকে উপেক্ষা করবে), তখন আমরা প্রোগ্রামটি কীভাবে লিখব? যেমন একটি উদাহরণ দিই। ধরা যাক, (0, 4) ঘরটি নিষিদ্ধ (blocked)। যদি রোবটের অবস্থান হয় (0, 3) ঘরে এবং তাকে 'R' কমান্ড দেওয়া হয়, তখন তার অবস্থানের কোনো পরিবর্তন হবে না। কারণ এক ঘর ডানে (মানে (0, 4) ঘরে) যাওয়া সম্ভব নয়।

এই সমস্যার সমাধান করতে যে প্রোগ্রামটি লিখতে হবে, তাতে কিন্তু একটি 2-D অ্যারে ব্যবহার করতে হবে। এই অ্যারের সাহায্যে আমরা বুঝব যে কোন ঘরে যাওয়া যাবে আর কোন ঘরে যাওয়া যাবে না। সেটি কীভাবে? খুবই সহজ। যেসব ঘরে যাওয়া যাবে অ্যারের ওই ঘরগুলোতে 1 আর যেসব ঘরে যাওয়া যাবে না সেগুলোতে 0 রাখব।

প্রথমে 10 x 10 গ্রিডের জন্য একটি 2-D অ্যারে ডিক্লেয়ার করি:

```
int grid[10][10];
```

তারপর শুরুতে ধরে নিই সব ঘরে যাওয়া যাবে।

```
for (i = 0; i < 10; i++) {
    for (j = 0; j < 10; j ++) {
        grid[i][j] = 1;
    }
}
```

এখন কোন কোন ঘরগুলোতে যাওয়া যাবে না তা ব্যবহারকারীর কাছ থেকে ইনপুট নিই:

```
printf("Please enter the number of blocked cells: ");
scanf("%d", &n);
printf("Now enter the cells: ");
for (i = 0; i < n; i++) {
    scanf("%d %d", &x, &y);
    grid[x][y] = 0;
}
```

এখন কোনো ঘরে যাওয়া যাবে কি না, সেটি বোঝার জন্য একটি শর্ত পরীক্ষা করলেই হবে।

```
if (grid[x][y] == 1) {
    যদি সত্য হয়, তবে (x, y) ঘরে যাওয়া যাবে।
}
```

এখন তোমরা সম্পূর্ণ প্রোগ্রামটি নিজে নিজে লিখে ফেলো।

অধ্যায় চোদ্দঃ শেষের শুরু

আমরা বইয়ের শেষ অধ্যায়ে চলে এসেছি। তোমরা যদি আগের অধ্যায়গুলো ঠিকমতো পড়ে থাকো, উদাহরণগুলো নিজে নিজে কম্পিউটারে চালিয়ে দেখে থাকো এবং যখনই আমি তোমাদেরকে কোনো প্রোগ্রাম নিজে লিখতে বলেছি, সেগুলো নিজে লেখার চেষ্টা করে থাকো, তাহলে তোমাকে অভিনন্দন! তুমি প্রোগ্রামিং শেখার জন্য প্রস্তুত হয়ে গেছ। যদি বলতে পারতাম তুমি প্রোগ্রামিং শিখে ফেলেছ তবে তোমাদেরও ভালো লাগত, আমারও ভালো লাগত, কিন্তু মিথ্যা কথা বলে কী লাভ?

প্রোগ্রামিং হচ্ছে চর্চার বিষয়। মুখস্থ করে পরীক্ষায় অনেক ভালো রেজাল্ট করা যায়, এমনকি কলেজ-বিশ্ববিদ্যালয়ে প্রোগ্রামিং পরীক্ষাতেও মুখস্থ করে অনেকেই বেশ ভালো নম্বর পায়। তবে এই ভালো নম্বর পাওয়ার সঙ্গে ভালো প্রোগ্রামার হওয়ার আসলে কোন সম্পর্ক নেই। প্রোগ্রামিং হচ্ছে একধরনের দক্ষতা (skill) এবং কেবল নিয়মিত অনুশীলনের মাধ্যমেই এই দক্ষতা অর্জন করা সম্ভব। এর জন্য ভালো ছাত্র হওয়ার দরকার নেই, জিনিয়াস হওয়ারও কোনো দরকার নেই। দরকার হচ্ছে প্রোগ্রামিংকে ভালোবাসা। যখন তুমি প্রোগ্রামিং করতে বসলে খাওয়াদাওয়ার কথা ভুলে যাবে, রাতে কোনো প্রোগ্রামিং সমস্যা নিয়ে কাজ শুরু করলে আর কিছুক্ষণ পরে দেখবে বাইরে ভোরের আলো ফুটছে, কিংবা ভুলে বাথরুমের স্যান্ডেল পরে ক্লাসে চলে যাবে, তখন বুঝবে যে তুমি প্রোগ্রামার হয়ে যাচ্ছ।

এখন তোমার উচিত হবে বইটি আরেকবার পড়া এবং সঙ্গে সঙ্গে প্রোগ্রামগুলো আবার করা। তারপর তোমরা আরও বেশি সি শিখতে চাইলে সি-এর কোন বই পড়তে পারো। তোমরা যদি প্রোগ্রামিং কন্টেস্টের ব্যাপারে উৎসাহী হও তবে সি প্লাস প্লাস (C++) শেখা শুরু করে দিতে পারো কোন বই থেকে। আবার জাভা (Java), সি শার্প (C#), পিএইচপি (PHP) কিংবা পাইথন (Python) শিখতে পারো। কোনোটি শিখতেই তেমন ঝামেলা পোহাতে হবে না কারণ তুমি প্রোগ্রামিংয়ের মৌলিক জিনিসগুলো এতক্ষণে আয়ত্বে এনে ফেলেছ। বই ও ওয়েবসাইটের তালিকা আমি পরিশিষ্ট অংশে লিখেছি।

একজন দক্ষ প্রোগ্রামার হতে গেলে যে জিনিসগুলো লাগবে তা হচ্ছে–
১) এক বা একাধিক প্রোগ্রামিং ল্যাঙ্গুয়েজে ভালো দখল,
২) ভালো একটি IDE ব্যবহারের দক্ষতা,
৩) প্রোগ্রামিংয়ের মৌলিক বিষয়গুলো সম্পর্কে স্বচ্ছ ধারনা,

৪) গণিত ও যুক্তিতে দক্ষতা,
৫) অবজেক্ট ওরিয়েনটেড প্রোগ্রামিংয়ে (OOP– Object Oriented Programming) দক্ষতা,
৬) ডাটা স্ট্রাকচার ও অ্যালগরিদমের মৌলিক জ্ঞান ও তা প্রয়োগের ক্ষমতা,
৭) যোগাযোগে দক্ষতা (Communication Skills),
৮) ইন্টারনেট ঘেঁটে অল্প সময়ে কোনো সমস্যার সমাধান বের করা বা দ্রুত কোন নতুন বিষয় শিখে নেওয়ার দক্ষতা,
৯) একটি সমস্যার পিছনে লেগে থাকার মানসিকতা,
১০) প্রোগ্রামিংয়ের প্রতি ভালোবাসা।

তোমাদের প্রোগ্রামিং জীবন আনন্দময় হোক, তোমাদের নিজের জীবন আনন্দময় হোক, তোমাদের কারণে তোমাদের আশেপাশের মানুষদের জীবন আনন্দময় হোক। সবাইকে শুভেচ্ছা।

তুমি যদি এই পিডিএফ বইটি কারো কাছ থেকে কপি করে থাকো, কিংবা ইন্টারনেট থেকে ফ্রি ডাউনলোড করে থাকো, তুমি চাইলে লেখকের পরিশ্রমের মূল্য পরিশোধ করতে পারো। এই পিডিএফ বইটির দাম একেবারেই হাতের নাগালে। মাত্র **25** টাকা বিকাশ-এর মাধ্যমে পাঠিয়ে দাও এই নাম্বারেঃ **01622624182**, তোমরা যত বেশি বই কিনবে, সেটি হার্ডকপিই হোক, কিংবা ইবুক (যেমন এই পিডিএফ), লেখকরা তত বেশি উৎসাহ পাবেন এবং আরো বই লেখার ব্যাপারে আগ্রহী হবেন।

বিকাশের মাধ্যমে পেমেন্ট করার নিয়মঃ

01. Go to your bKash Mobile Menu by dialing *247#
02. Choose “Payment”
03. Enter the Merchant bKash Account Number you want to pay to (01622624182)
04. Enter the amount you want to pay (25)
05. Enter a reference against your payment (cpbook pdf)
06. Enter the Counter Number (enter 0)
07. Now enter your bKash Mobile Menu PIN to confirm

১) প্রথমে *247# নাম্বারে ডায়াল করে বিকাশ মেনু আনতে হবে।
২) তারপরে 'পেমেন্ট' নির্বাচন করতে হবে।
৩) এবারে মার্চেন্ট বিকাশ একাউন্ট নাম্বার হিসেবে 01622624182 দিতে হবে।
৪) তারপরে টাকার পরিমাণ দিতে হবে (এই পিডিএফ বইয়ের জন্য ২৫ টাকা)
৫) রেফারেন্স হিসেবে cpbook pdf লিখে দিতে হবে।
৬) কাউন্টার নাম্বার ০ দিবে।
৭) এবারে তোমার বিকাশ মোবাইল মেনু পিন (PIN) দিতে হবে।

তোমার বিকাশ একাউন্ট না থাকলে একটি খুলে নিতে পারো (এটি খুলতে কোনো টাকা লাগে না) কিংবা অন্য কারো বিকাশ একাউন্ট ব্যবহার করতে পারো।

পরিশিষ্ট একঃ প্রোগ্রামিং প্রতিযোগিতা

প্রোগ্রামিং প্রতিযোগিতা হচ্ছে প্রোগ্রামারদের মধ্যে লড়াই। এর মানে কিন্তু এই নয় যে প্রোগ্রামাররা একে অপরের সঙ্গে মারামারি করবে আর শেষ পর্যন্ত যে টিকে থাকবে সে-ই বিজয়ী। আসলে প্রোগ্রামিং প্রতিযোগিতা হচ্ছে একটি পরীক্ষার মতো যেখানে প্রত্যেককে একটি নির্দিষ্ট সময়ে নির্দিষ্টসংখ্যক প্রোগ্রামিং সমস্যার সমাধান করতে দেওয়া হবে। যে সবচেয়ে বেশি সমস্যার নির্ভুল সমাধান করবে সে বিজয়ী হবে। আর দুজন যদি সমানসংখ্যক সমস্যার সমাধান করে, তবে তাদের মধ্যে যে কম সময়ে করেছে সে বিজয়ী। তবে স্কুল-কলেজের পরীক্ষার সঙ্গে এর পার্থক্য হচ্ছে, এখানে বই থেকে সরাসরি প্রশ্ন করা হয় না। তাই মুখস্থ করার কোনো সুযোগ নেই। বিচারকেরা অনেক সময় নিয়ে প্রোগ্রামিং প্রতিযোগিতার সমস্যা তৈরি করেন। এর মধ্যে সহজ সমস্যাও থাকে আবার খুব কঠিন সমস্যাও থাকে।

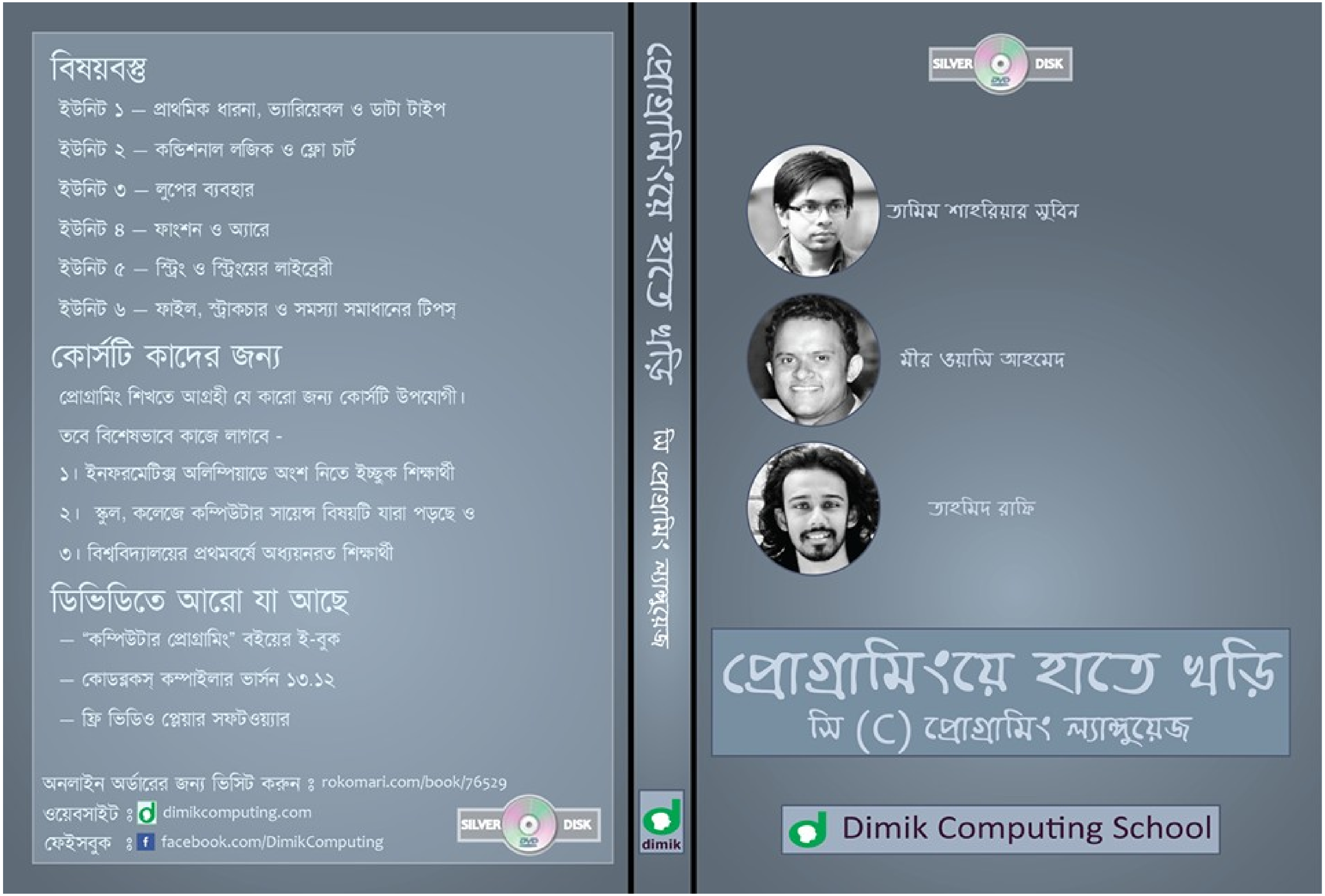
স্কুল-কলেজের ছাত্রছাত্রীদের জন্য সবচেয়ে বড় প্রতিযোগিতা হচ্ছে আইওআই (IOI– International Olympiad in Informatics)। 1989 সাল থেকে প্রতিবছর এ প্রতিযোগিতা অনুষ্ঠিত হচ্ছে। একেক বছর একেক দেশে প্রতিযোগিতা অনুষ্ঠিত হয়। বিগত চার বছর যাবৎ বাংলাদেশ এ প্রতিযোগিতায় অংশগ্রহণ করে আসছে। এখন পর্যন্ত আমাদের সেরা অর্জন হচ্ছে 2009 সালে আবিরুল ইসলামের রৌপ্য পদক (সিলভার মেডেল)। IOI তে অংশগ্রহণ করার জন্য বাংলাদেশ দল গঠনের কাজটি করা হয় দুই ধাপে। প্রথমে বিভাগীয় ইনফরমেটিক্স অলিম্পিয়াড। তারপর বিভাগীয় পর্যায়ের বিজয়ীদের নিয়ে জাতীয় ইনফরমেটিক্স অলিম্পিয়াড অনুষ্ঠিত হয়। জাতীয় অলিম্পিয়াডের বিজয়ীদের মধ্য থেকেই দলের সদস্য বাছাই করা হয়।

বিশ্ববিদ্যালয় পর্যায়ের ছাত্রছাত্রীদের জন্য সবচেয়ে বড় প্রোগ্রামিং প্রতিযোগিতা হচ্ছে এসিএম আইসিপিসি (ACM ICPC– ACM International Collegiate Programming Contest)। এর জন্য দল বাছাই অনেকটা বিশ্বকাপ ফুটবলের মতো হয়। প্রতি মহাদেশ থেকে প্রতিযোগিতার মাধ্যমে দল নির্বাচন করা হয়। একটি দলে তিনজন সদস্য এবং একজন প্রশিক্ষক থাকেন। মজার ব্যাপার হচ্ছে দলের সদস্যদের কিন্তু কম্পিউটার বিজ্ঞানের শিক্ষার্থী হতে হবে, এমন কোনো কথা নেই। যেকোনো বিভাগের শিক্ষার্থী এই প্রতিযোগিতায় অংশগ্রহণ করতে পারে। বাংলাদেশের প্রতিযোগিরা ঢাকায় ICPC Regional Contest-এ অংশগ্রহণ করে। এছাড়া ভারত ও আশেপাশের দেশের ICPC Regional Contest-এও বাংলাদেশের প্রতিযোগীদের অংশগ্রহণের সুযোগ রয়েছে এবং প্রায়ই আমাদের দেশের কয়েকটি দল ওইসব প্রতিযোগিতায় অংশগ্রহণ করে। ICPC Regional Contest-এ বিজয়ী দলগুলো সুযোগ পায় চূড়ান্ত পর্বে (ICPC World Finals) অংশগ্রহণ করার। 1998 সালের পর থেকে প্রতি বছরই বাংলাদেশ থেকে কমপক্ষে

একটি দল চূড়ান্ত পর্বে অংশগ্রহণের যোগ্যতা লাভ করে যা আমাদের দেশের প্রোগ্রামারদের কৃতিত্বের পরিচয় বহন করে। তোমরা এ বিষয়ে উইকিপিডিয়াতে আরও তথ্য পাবে এই লিংকে: http://en.wikipedia.org/wiki/ACM_ICPC_Dhaka_Site।

এছাড়া ইন্টারনেটে অনুষ্ঠিত হয় আরও নানা ধরনের প্রোগ্রামিং প্রতিযোগিতা যেখানে স্কুল-কলেজ-বিশ্ববিদ্যালয়ের ছাত্র, শিক্ষক ও পেশাজীবীরা অংশ্রগ্রহণ করতে পারেন। এদের মধ্যে গুরুত্বপূর্ণ তিনটি হচ্ছে Google Code Jam (http://code.google.com/codejam), Topcoder (http://www.topcoder.com/tc) এবং Codechef (http://www.codechef.com/)। এই প্রতিযোগিতাগুলো অত্যন্ত কঠিন, তাই এতে অংশগ্রহণের জন্য পর্যাপ্ত দক্ষতা থাকতে হবে। তবে এসব প্রতিযোগিতায় কিন্তু বাংলাদেশের প্রোগ্রামাররা বেশ ভালো অবস্থানে রয়েছে।

প্রোগ্রামিং প্রতিযোগিতায় ভালো করতে হলে প্রোগ্রামিংয়ে দক্ষতার পাশাপাশি অ্যালগরিদম ও গণিতে বিশেষভাবে দক্ষ হতে হয়। জ্যামিতি, কম্বিনেটরিক্স, সংখ্যাতত্ত্ব ইত্যাদির যথেষ্ট জ্ঞানের পাশাপাশি সমস্যা সমাধানের দক্ষতা অর্জন করতে হয়। এজন্য লেখাপড়ার পাশাপাশি নিয়মিত প্রোগ্রামিং সমস্যা সমাধানের কোনো বিকল্প নেই। আর বিভিন্ন ওয়েবসাইটে নিয়মিত প্রোগ্রামিং প্রতিযোগিতার আয়োজন করা হয় যেখানে তুমি অংশগ্রহণ করতে পারো ইন্টারনেটের মাধ্যমে।


বিষয়বস্তু
ইউনিট ১ — প্রাথমিক ধারনা, ভ্যারিয়েবল ও ডাটা টাইপ
ইউনিট ২ — কন্ডিশনাল লজিক ও ফ্লো চার্ট
ইউনিট ৩ — লুপের ব্যবহার
ইউনিট ৪ — ফাংশন ও অ্যারে
ইউনিট ৫ — স্ট্রিং ও স্ট্রিংয়ের লাইব্রেরী
ইউনিট ৬ — ফাইল, স্ট্রাকচার ও সমস্যা সমাধানের টিপস্
কোর্সটি কাদের জন্য
প্রোগ্রামিং শিখতে আগ্রহী যে কারো জন্য কোর্সটি উপযোগী।
তবে বিশেষভাবে কাজে লাগবে -
১। ইনফরমেটিক্স অলিম্পিয়াডে অংশ নিতে ইচ্ছুক শিক্ষার্থী
২। স্কুল, কলেজে কম্পিউটার সায়েন্স বিষয়টি যারা পড়ছে ও
৩। বিশ্ববিদ্যালয়ের প্রথমবর্ষে অধ্যয়নরত শিক্ষার্থী
ডিভিডিতে আরো যা আছে
— "কম্পিউটার প্রোগ্রামিং" বইয়ের ই-বুক
— কোডব্লকস্ কম্পাইলার ভার্সন ১৩.১২
— ফ্রি ভিডিও প্লেয়ার সফটওয়্যার
অনলাইন অর্ডারের জন্য ভিজিট করুন ঃ rokomari.com/book/76529
ওয়েবসাইট ঃ dimikcomputing.com
ফেইসবুক ঃ facebook.com/DimikComputing
SILVER DISK
প্রোগ্রামিংয়ে হাতে খড়ি
সি প্রোগ্রামিং ল্যাঙ্গুয়েজ
dimik
SILVER DISK
তামিম শাহরিয়ার সুবিন
মীর ওয়াসি আহমেদ
তাহমিদ রাফি
প্রোগ্রামিংয়ে হাতে খড়ি
সি (C) প্রোগ্রামিং ল্যাঙ্গুয়েজ
Dimik Computing School

পরিশিষ্ট দুইঃ প্রোগ্রামিং ক্যারিয়ার

গণিত যেমন কেবল গণিতবিদেরাই ব্যবহার করেন না, বরং বিজ্ঞানের সব শাখায় রয়েছে এর ব্যবহার, তেমনই প্রোগ্রামিংও কিন্তু কেবল কম্পিউটার বিজ্ঞানী বা কম্পিউটার ইঞ্জিনিয়ারদের জন্য নয়। বিশ্ববিদ্যালয়ে পড়তে গেলে বিজ্ঞান ও প্রকৌশলের সব বিভাগের শিক্ষার্থীদের জন্য প্রোগ্রামিং জানাটা খুব গুরুত্বপূর্ণ।

পেশা হিসেবে প্রোগ্রামিংয়ের আলাদা একটি গুরুত্ব আছে আমাদের জন্য। যেহেতু বিভিন্ন ধরনের প্রতিযোগিতার মাধ্যমে নিজেকে মেলে ধরার অনেক সুযোগ এখানে রয়েছে ,তাই বাংলাদেশ থেকে লেখাপড়া করে সরাসরিই বিশ্বের নামকরা সফটওয়্যার নির্মাতা প্রতিষ্ঠান যেমন– গুগল, মাইক্রোসফট, ফেসবুক ইত্যাদিতে কাজ করার সুযোগ তৈরি হয়েছে। প্রতিবছরই বাংলাদেশ থেকে কয়েকজন প্রোগ্রামার নিজের মেধা ও জ্ঞানকে কাজে লাগিয়ে এই সুযোগের সদ্ব্যবহার করছেন। অনেক ক্ষেত্রেই ওইসব প্রতিষ্ঠানে কাজ করার জন্য আবেদন করার প্রয়োজন হয় না, তারা নিজে থেকেই বিভিন্ন দেশের সেরা প্রোগ্রামারদের খুঁজে বের করে।

বিশ্ববিখ্যাত সব প্রতিষ্ঠানে কাজ করা ছাড়াও প্রোগ্রামারদের জন্য আরেকটি সুবিধা হচ্ছে Telecommuting। অর্থাৎ কোনো অফিসে না গিয়ে কাজ করার সুযোগ। উন্নত বিশ্বের অনেক কোম্পানি তাদের নিজ দেশে প্রোগ্রামারদের দুষ্প্রাপ্যতার কারণে উন্নয়নশীল অর্থনীতির দেশের প্রোগ্রামারদের কাজের সুযোগ দেয়, আর সে ক্ষেত্রে নিজ দেশে বসেই কাজ করা যায়। কারণ ওই কোম্পানিগুলো জানে যে আমাদের মত দেশের অর্থনীতি উন্নয়নশীল হলেও প্রোগ্রামাররা মোটেও অদক্ষ নন, বরং বিশ্বমানের প্রোগ্রামার। বাংলাদেশের বেশকিছু প্রোগ্রামার এখন বাংলাদেশে বসেই ইন্টারনেটের মাধ্যমে কাজ করছেন আমেরিকা, কানাডা ও ইউরোপের বিভিন্ন দেশের সফটওয়্যার কোম্পানিতে।

আরেকটি মজার ব্যাপার হচ্ছে, কেউ যদি ধরাবাঁধা চাকরি করতে না চায় তবে তার জন্য ফ্রিল্যান্স প্রোগ্রামিংয়ের সুযোগ রয়েছে। ইন্টারনেটে অনেক ওয়েবসাইট আছে যেখানে ছোট-মাঝারি-বড় বিভিন্ন ধরনের সফটওয়্যারের প্রজেক্ট থাকে যেগুলোতে বিড (bid) করে কাজ করা যায়। বাংলাদেশে এখন শত শত প্রোগ্রামার ফ্রিল্যান্স প্রোগ্রামিংয়ের সঙ্গে জড়িত। এর জন্য কেবল কম্পিউটার ও ইন্টারনেট সংযোগ থাকলেই চলবে। ফ্রিল্যান্স কাজ করার জন্য বিপুল ধৈর্যের প্রয়োজন। আর ইংরেজি ভাষায় যোগাযোগের দক্ষতা থাকতে হয়। তবে ব্যক্তিগতভাবে আমি মনে করি, ছাত্রাবস্থায় এ ধরনের কাজ না করাই ভালো। কারণ ছাত্রজীবনে লেখাপড়া করার ও মৌলিক বিষয়গুলো আয়ত্ত্ব করার যে সময় ও সুযোগ মেলে, জীবনের পরবর্তী পর্যায়ে কখনোই সেই সুযোগ পাওয়া যায় না। তাই

তোমাদের প্রতি আমার পরামর্শ থাকবে যে ছাত্রজীবনে অর্থ উপার্জনের দিকে মনোযোগ না দিয়ে প্রচুর লেখাপড়া এবং সঙ্গে সঙ্গে নানা ধরনের সামাজিক ও সাংস্কৃতিক কর্মকাণ্ডে জড়িত থাকার চেষ্টা করবে, যেগুলো তোমার ভালো লাগে।

সব শেষ কথা হচ্ছে, প্রোগ্রামিং এমন একটি কাজ যেখানে সব সময়ই তোমার নিজেকে উন্নত করার সুযোগ আছে। তাই লেখাপড়া করার মানসিকতা থাকতে হবে, পড়তে হবে নানা বইপত্র, ঘাঁটতে হবে ইন্টারনেট। নিজে কোনো সমস্যায় পড়লে প্রথমেই ইন্টারনেট ঘেঁটে দেখবে যে সমস্যাটির সমাধান ইতিমধ্যে কেউ করে রেখেছে কি না। বিভিন্ন ফোরাম ও ব্লগে সাধারণত প্রোগ্রামাররা কিছু কমন সমস্যার সমাধান দিয়ে রাখে। তবে প্রোগ্রামিং শেখার সময় কিন্তু সমাধানের জন্য ইন্টারনেট ঘাঁটবে না, নিজে চেষ্টা করবে।

ক্যারিয়ার হিসেবে ওয়েব ডেভেলাপমেন্ট খুব আকর্ষনীয়। ওয়েবের কাজ জানা মানুষের রয়েছে দেশীয় এবং আন্তর্জাতিক বাজারে ব্যাপক চাহিদা। আর তাই ওয়েব সম্পর্কে থাকা চাই স্বচ্ছ ধারনা। এই ভিডিওগুলোতে সেই ধারনা সহজভাবে দেওয়ার চেষ্টা করেছেন তাহমিদ রাফি। স্কুল-কলেজ-বিশ্ববিদ্যালয় পড়ুয়া শিক্ষার্থীরা এই ভিডিওগুলো থেকে উপকৃত হবে, আর সেই সাথে ওয়েবের কাজের নানান দিক সম্পর্কে সম্যক জ্ঞান লাভ করবে। এখান থেকে তারা তাদের পরবর্তী করণীয় সম্পর্কে দিক নির্দেশনাও পেয়ে যাবে।

বিষয়বস্তু

ইউনিট ১ — প্রাথমিক ধারনা

ওয়ার্ল্ড ওয়াইড ওয়েবের ইতিহাস, সার্ভার ক্লায়েন্ট, আইপি এড্রেস, ডোমেইন নেম সার্ভার, ইউনিফর্ম রিসোর্স লোকেটর

ইউনিট ২ — ওয়েব সাইটের গঠন : ক্লায়েন্ট অংশ

এইচ টি এম এল (HTML), এইচ টি এম এল ট্যাগ (Tag), সি এস এস (CSS), জাভাস্ক্রিপ্ট (Javascript)

ইউনিট ৩ — ওয়েব সাইটের গঠন : সার্ভার অংশ

জাভাস্ক্রিপ্ট লাইব্রেরী, সি এস এস ৩ (CSS3), এইচ টি এম এল ৫ (HTML5), ওয়েব এপ্লিকেশন ফ্রেমওয়ার্ক, কনটেন্ট ম্যানেজমেন্ট সিস্টেম (CMS), জুমলা, ওয়ার্ডপ্রেস, চাকুরীর বাজার

ইউনিট ৪ — আধুনিক টেকনোলজি

ওয়েব সার্ভার, সার্ভার সাইড স্ক্রিপ্টিং, ডাটাবেস, এপাচি (Apache), পি এইচ পি (PHP), হোস্টিং সার্ভিস

ডিভিডিতে আরো যা আছে

— ক্লাসের সবগুলো লেকচার স্লাইডের সফট কপি (পিডিএফ ভার্সন)
— দরকারী লিঙ্কসমূহ
— ফ্রি ভিডিও প্লেয়ার সফটওয়্যার

অনলাইন অর্ডারের জন্য ভিসিট করুন ঃ rokomari.com/book/76530
ওয়েবসাইট ঃ dimikcomputing.com
ফেইসবুক ঃ facebook.com/DimikComputing

SILVER DISK

ওয়েব কনসেপ্টস্ * তাহমিদ রাফি

dimik

SILVER DISK

ওয়েব কনসেপ্টস্

তাহমিদ রাফি

Dimik Computing School

পরিশিষ্ট তিনঃ বই ও ওয়েবসাইটের তালিকা

তুমি যদি ইতিমধ্যে এই বইটি পড়ে ফেলো এবং এবারে ভালোভাবে সি শিখতে চাও, তবে Herbert Schildt-এর Teach Yourself C বইটি পড়তে পারো। আবার Brian Kernighan ও Dennis Ritchie-এর লেখা The C Programming Language বইটিও পড়তে পারো। লেখকদের একজন, Dennis Ritchie, সি ল্যাঙ্গুয়েজ ডিজাইন করেছেন। আর কেউ যদি তোমার কাছে জানতে চায় শুরুতে সি শিখতে হলে কোন ইংরেজি বইটি ভালো তবে Stephen G. Kochan-এর Programming in C বইটির কথা বলে দেবে। এটি সি শেখার জন্য চমৎকার ও সহজ একটি বই। Schaums Outlines সিরিজের Programming with C বইটিও ভালো। বইতে প্রচুর উদাহরণ আর অনুশীলনী আছে।

সি শেখার পরে তুমি সি প্লাস প্লাস বা জাভা শিখতে পারো। সি প্লাস প্লাস শেখার জন্য ভালো বই হচ্ছে Teach Yourself C++ (লেখক: Herbert Schildt) আর জাভার জন্য Java How to Program (লেখক: Paul Deitel and Harvey Deitel)। তারপর অন্য ল্যাঙ্গুয়েজ শিখতে গেলে আর বই কেনার দরকার নেই। ইন্টারনেটে প্রচুর টিউটোরিয়াল আছে। সেগুলো পড়ে শিখে ফেলবে।

তুমি যদি কম্পিউটার বিজ্ঞানে পড়তে চাও, কিংবা প্রোগ্রামিং কন্টেস্টে ভালো করতে চাও, তাহলে তোমার Discrete Mathematics ভালো করে শিখতে হবে। এর জন্য Kenneth H. Rosen-এর Discrete Mathematics বইটি খুব ভালো। আগাগোড়া পড়ে ফেলবে। সঙ্গে সঙ্গে অনুশীলনীর সমস্যাগুলো সমাধানের চেষ্টা করবে। Discrete Mathematics শেখার পরে শিখতে হবে অ্যালগরিদম। অ্যালগরিদম শেখার শুরু আছে কিন্তু শেষ নেই। আর শুরু করার জন্য তোমরা পড়তে পারো Introduction to Algorithms (লেখক: Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein) এটি অ্যালগরিদমের মৌলিক বিষয়গুলো শেখার জন্য আমার দেখা সবচেয়ে ভালো বই।

প্রোগ্রামিং প্রতিযোগিতার জন্য কিছু লিংক:

- http://projecteuler.net/ এখানে অনেক মজার সমস্যা আছে যেগুলোর বেশিরভাগই প্রোগ্রাম লিখে সমাধান করতে হয়। এখানে প্রোগ্রাম জমা দেওয়া লাগে না, কেবল প্রোগ্রাম দিয়ে বের করা উত্তরটা জমা দিতে হয়।
- http://www.spoj.pl/ এখানেও অনেক ভালো সমস্যা আছে। সমাধান করে প্রোগ্রাম জমা দিলে প্রোগ্রাম সঠিক হয়েছি কি না তা জানা যায়। এই ওয়েবসাইটের একটি বৈশিষ্ট্য হচ্ছে সি, সি প্লাস প্লাস, জাভা, পার্ল, পাইথন, রুবি, পিএইচপি ইত্যাদি

ব্যবহার করে প্রোগ্রাম লেখা যায়।

- http://uva.onlinejudge.org/ এই সাইটে নিয়মিত অনলাইন প্রোগ্রামিং প্রতিযোগিতার আয়োজন করা হয়। এ ছাড়াও অনুশীলনের জন্য প্রচুর সমস্যা দেওয়া আছে। নতুন প্রোগ্রামারদের জন্য এটি বেশ ভালো জায়গা।
- http://ace.delos.com/usacogate এটি যদিও আমেরিকার ইনফরমেটিক্স অলিম্পিয়াড ট্রেনিং প্রোগ্রাম, কিন্তু সাইটে যেকোনো দেশের প্রোগ্রামাররাই রেজিস্ট্রেশন করে অনুশীলন করতে পারে। তোমরা যারা প্রোগ্রামিং প্রতিযোগিতায় ভালো করতে চাও, তাদের অবশ্যই এখানে অনুশীলন করা উচিত।
- http://www.topcoder.com/tc এখানেও নিয়মিত অনলাইন প্রোগ্রামিং প্রতিযোগিতা অনুষ্ঠিত হয়। এখানে ভালো ফলাফল করলে আবার টাকাও দেয় (কী আনন্দ!)। এ ছাড়া এখানে অনেক ভালো টিউটোরিয়াল ও আর্টিকেল আছে। এটি অভিজ্ঞ প্রোগ্রামারদের জন্য বেশ ভালো একটি সাইট।
- http://codeforces.com/ এই সাইটে নিয়মিত বিভিন্ন ধরনের প্রোগ্রামিং কন্টেস্ট হয়। অভিজ্ঞ প্রোগ্রামারদের জন্য ভালো।
- http://www.codechef.com/ এটিও প্রোগ্রামিং প্রতিযোগিতার জন্য একটি ভালো ওয়েবসাইট এবং অভিজ্ঞ প্রোগ্রামারদের জন্য।
- http://ioinformatics.org আন্তর্জাতিক ইনফরমেটিক্স অলিম্পিয়াডের অফিসিয়াল ওয়েবসাইট।
- http://cm.baylor.edu/welcome.icpc এসিএম আইসিপিসির অফিসিয়াল ওয়েবসাইট।

প্রোগ্রামিং ছাড়াও বিজ্ঞান ও গণিতের নানা বিষয়ের জন্য এই ফোরামে অংশগ্রহণ করতে পারো: http://matholympiad.org.bd/forum/।

আর সবচেয়ে গুরুত্বপূর্ণ ওয়েবসাইট হচ্ছে www.google.com। এটি আসলে একটি সার্চ ইঞ্জিন। যখনই কোন কিছু জানতে ইচ্ছা করবে, google-এ সার্চ করলে তুমি সেই বিষয়ের নানা তথ্যসমৃদ্ধ ওয়েবসাইটের লিংক পেয়ে যাবে।

পরবর্তী পেজ থেকে তোমাদের অনুশীলনের জন্য ৪০টি প্রোগ্রামিং সমস্যা দেওয়া হলো। সমস্যাগুলো তৈরি করেছেন তামিম শাহরিয়ার সুবিন, মোহাম্মাদ মাহমুদুর রহমান, তানভীরুল ইসলাম, ইকরাম মাহমুদ ফাহিম, মাহবুব মোযাদ্দেজ সৌরভ, শুভানন রাযিক, যোবায়ের হাসান। আরো কিছু সমস্যা বাংলাদেশ ইনফরমেটিক্স অলিম্পিয়াডের ট্রেনিং প্রোগ্রাম থেকে সংগৃহীত হয়েছে। সমস্যাগুলোর সমাধানের জন্য বইয়ের প্রোগ্রামিং জ্ঞানই যথেষ্ট।

সমস্যাগুলো সমাধান করে সেগুলো সঠিক হলো কী না, সেটি তোমরা যাচাই করতে পারো কম্পিউটার প্রোগ্রামিং বইয়ের ওয়েবসাইটে। লিঙ্কঃ http://cpbook.subeen.com/p/blog-page_11.html

আর সমস্যাগুলো কিভাবে সমাধান করতে হবে, তার একটি উদাহরণ এখানে ভিডিও-এর মাধ্যমে দেখানো হয়েছেঃ http://cpbook.subeen.com/2012/12/problem-solving-guideline-1.html

আর প্রোগ্রামিং বিষয়ক বিভিন্ন অনলাইন কোর্স তুমি ফ্রি-তে করতে পারো দ্বিমিক কম্পিউটিং স্কুল থেকে। দ্বিমিকের ওয়েবসাইটের ঠিকানা হচ্ছে http://dimikcomputing.com এবং
ফেসবুক পেজের লিঙ্কঃ https://www.facebook.com/DimikComputing।
তুমি চাইলে কোর্সের ডিভিডিগুলো দ্বিমিক-এর ওয়েবসাইট থেকে কিনে নিতে পারো। বিকাশের মাধ্যমে পেমেন্ট করলে সেগুলো কুরিয়ারের মাধ্যমে পাঠানো হবে। আর বিকাশ-এ টাকা পাঠাতে না পারলে রকমারি ডট কম-থেকেও কিনতে পারো (ফোন নাম্বারঃ ০১৫ ১৯৫২ ১৯৭১)। এছাড়া ঢাকার নীলক্ষেতের হক লাইব্রেরি-তে ডিভিডিগুলো পাওয়া যায়।

প্রোগ্রামিং সমস্যা ১ - ধনাত্মক - ঋণাত্মক

শূণ্যের চেয়ে বড় সংখ্যাগুলো হচ্ছে ধনাত্মক আর শূণ্যের চেয়ে ছোট সংখ্যাগুলো হচ্ছে ঋণাত্মক সংখ্যা। অনেকগুলো সংখ্যা দেওয়া থাকবে, কয়টা ধনাত্মক আর কয়টা ঋণাত্মক সেটি বের করতে হবে।

ইনপুট

প্রথম লাইনে একটি সংখ্যা দেওয়া থাকবে। সংখ্যাটির মান যত হব ততটি লাইনে একটি করে সংখ্যা থাকবে।

আউটপুট

আউটপুটে দুটি সংখ্যা থাকবে। প্রথমে থাকবে মোট ধনাত্মক সংখ্যা, তারপরে একটি স্পেস ক্যারেক্টার, তারপরে মোট ঋণাত্মক সংখ্যা।

উদাহরণ

ইনপুট:

```
5
10
23
-8
2
-9765
```

আউটপুট:

```
3 2
```

সমস্যাটি সমাধান করে যাচাই করা যাবে এই লিঙ্কেঃ http://cpbook.subeen.com/2012/11/positive-negative.html

প্রোগ্রামিং সমস্যা ২ - সংখ্যা গণনা

এক লাইনে অনেকগুলো সংখ্যা দেওয়া থাকবে। সংখ্যাগুলোর মধ্যে এক বা একাধিক স্পেস ক্যারেক্টার থাকবে। লাইনে মোট কয়টি সংখ্যা আছে সেটি বের করতে হবে।

ইনপুট

প্রথম লাইনে একটি সংখ্যা থাকবে। তারপর সেই সংখ্যাটির মান যত, ততটি লাইন থাকবে। প্রতি লাইনে এক বা একাধিক সংখ্যা থাকবে যাদের পরমমান 10000000-এর বেশি হবে না। একটি লাইনের সংখ্যাগুলোর মাঝে এক বা একাধিক স্পেস থাকবে।

আউটপুট

প্রতি লাইনে কয়টি সংখ্যা আছে সেটি প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট:

```
4
1  -2 10000 -50 20 7 445
9
-98 876 65
223   9876452  212
```

আউটপুট:

```
7
1
3
3
```

সমস্যাটি সমাধান করে যাচাই করা যাবে এই লিঙ্কেঃ

প্রোগ্রামিং সমস্যা ৩ - **আয়তন-১**

একটি আয়তাকার বাক্সের আয়তন বাক্সটির দৈর্ঘ্য, প্রস্থ ও উচ্চতার গুণফলের সমান। তোমাকে যে কোন বাক্সের এই তিনটি পরিমাপ বলে দিলে তাদের আয়তন নির্ণয় করার একটি প্রোগ্রাম লিখতে পারবে কি?

ইনপুট

প্রথম লাইনে একটি সংখ্যা থাকবে। তারপর সেই সংখ্যাটির মান যত, ততটি লাইন থাকবে। প্রতি লাইনে তিনটি করে সংখ্যা থাকবে যার প্রতিটির মান-ই ১ থেকে ১০০-এর মধ্যে সীমাবদ্ধ। একটি লাইনের সংখ্যাগুলোর মাঝে এক বা একাধিক স্পেস ক্যারেক্টার থাকবে। প্রতি লাইনের সংখ্যা তিনটি একটি বাক্সের দৈর্ঘ্য, প্রস্থ ও উচ্চতার মান নির্দেশ করে।

আউটপুট

প্রতি লাইনের জন্য সেই লাইনে যে বাক্সের মাপ তিনটি দেয়া আছে তার আয়তন প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট:

```
2
2 2 2
3 5 10
```

আউটপুট:

```
8
150
```

সমস্যাটি সমাধান করে যাচাই করা যাবে এই লিঙ্কেঃ

প্রোগ্রামিং সমস্যা ৪- **ASCII যোগ**

আমরা জানি কম্পিউটারে যে কোনো অক্ষরকেই একটি ASCII সংখ্যা দ্বারা প্রকাশ করা হয়। তোমাকে কতগুলো তিন অক্ষরের শব্দ দেয়া হল। এখন তোমার কাজ হল এর প্রতিটি অক্ষরের ASCII মানের যোগফল বের করা।

ইনপুট

প্রথম লাইনে একটি সংখ্যা থাকবে। তারপর সেই সংখ্যাটির মান যত, ততটি লাইন থাকবে। প্রতি লাইনে একটি করে তিন অক্ষরের ইংরেজি শব্দ দেয়া থাকবে।

আউটপুট

প্রতি লাইনের জন্য সেই লাইনে দেয়া শব্দের অক্ষর তিনটির ASCII মানের যোগফল প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট:

```
2
CAT
dog
```

আউটপুট:

```
216
314
```

সমস্যাটি সমাধান করে যাচাই করা যাবে এই লিঙ্কেঃ

প্রোগ্রামিং সমস্যা ৫ - **বৃহত্তম ও ক্ষুদ্রতম**

তোমার ক্লাসের ৫ জনের অংক পরীক্ষার নম্বর দেয়া হল। কে সবচেয়ে বেশী পেয়েছে আর কে সবচেয়ে কম বের করতে পারবে?

ইনপুট

প্রথম লাইনে একটি সংখ্যা থাকবে। তারপর সেই সংখ্যাটির মান যত, ততটি লাইন থাকবে। প্রতি লাইনে থাকবে ০ থেকে ১০০-এর মধ্যে ৫টি করে সংখ্যা যা একটি পরীক্ষায় পাওয়া ৫ জনের নম্বর।

আউটপুট

প্রতি লাইনের জন্য সেই লাইনে দেয়া নম্বরগুলোর মধ্যে প্রথমে সবচেয়ে বেশীটি এর পরে সবচেয়ে কমটি প্রিন্ট করতে হবে। সংখ্যা ২টির মধ্যে ঠিক একটি স্পেস প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট:

```
2
1 2 3 4 5
0 10 50 80 100
```

আউটপুট:

```
5 1
100 0
```

সমস্যাটি সমাধান করে যাচাই করা যাবে এই লিঙ্কেঃ

প্রোগ্রামিং সমস্যা ৬ - পূর্ণবর্গ সংখ্যা

একটি ধনাত্মক সংখ্যা দেওয়া থাকবে, বলতে হবে সংখ্যাটি পূর্ণবর্গ কিনা ।

ইনপুট

প্রথম লাইনে একটি সংখ্যা দেওয়া থাকবে। সংখ্যাটির মান যত হবে ততটি লাইনে একটি করে সংখ্যা থাকবে। সব সংখ্যা 0-এর চেয়ে বড় এবং 10000001-এর চেয়ে ছোট।

আউটপুট

ইনপুটের সংখ্যাটি পূর্ণবর্গ হলে প্রিন্ট করতে হবে YES আর নাহলে প্রিন্ট করতে হবে NO।

উদাহরণ

ইনপুট:

```
5
4
10
64
99
25
```

আউটপুট:

```
YES
NO
YES
NO
YES
```

সমস্যাটি সমাধান করে যাচাই করা যাবে এই লিঙ্কেঃ

প্রোগ্রামিং সমস্যা ৭ - গড় - ১

তোমার ক্লাসের ৫ জনের অংক পরীক্ষার নম্বর দেয়া হল। এই ৫ জনের নম্বরের গড় বের করতে হবে।

ইনপুট

প্রথম লাইনে একটি সংখ্যা থাকবে। তারপর সেই সংখ্যাটির মান যত, ততটি লাইন থাকবে। প্রতি লাইনে থাকবে ০ থেকে ১০০-এর মধ্যে ৫টি করে সংখ্যা যা একটি পরীক্ষায় পাওয়া ৫ জনের নম্বর।

আউটপুট

প্রতি লাইনের জন্য সেই লাইনে দেয়া নম্বরগুলোর গড় প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট:

```
2
1 2 3 4 5
0 10 50 80 100
```

আউটপুট:

```
3
48
```

প্রোগ্রামিং সমস্যা ৮ - গড় - ২

তোমার ক্লাসের সামাজিক বিজ্ঞান পরীক্ষার ফল হাতে পেয়ে গিয়েছ। কেউ খুশি, কেউ বেজার। কিন্তু তোমার খুশি বা বেজার হওয়ার সময় নেই। কারণ তোমার হাতে কাজ আছে। তোমার একটি প্রোগ্রাম লিখতে হবে যেটি দিয়ে সামাজিক বিজ্ঞান পরীক্ষায় তোমার ক্লাসের সব শিক্ষার্থীর গড় নম্বর বের করতে হবে।

ইনপুট

প্রথম লাইনে একটি সংখ্যা থাকবে। সংখ্যাটির মান যত, প্রথম লাইনের পরে ততটি লাইন থাকবে। প্রতি লাইনে প্রথম সংখ্যাটি হচ্ছে মোট শিক্ষার্থীর সংখ্যা। তারপর থাকবে একটি স্পেস ক্যারেক্টার। তারপর সব শিক্ষার্থীর সামাজিক বিজ্ঞান পরীক্ষায় প্রাপ্ত নম্বর থাকবে এবং প্রতিটি সংখ্যা এক বা একাধিক স্পেস ক্যারেক্টার দিয়ে পৃথক করা থাকবে।

আউটপুট

প্রথম লাইন বাদে প্রতি লাইন ইনপুটের জন্য আউটপুটে একটি সংখ্যা প্রিন্ট করতে হবে যেটি হচ্ছে পরীক্ষার নম্বরের গড়। দশমিকের পরে দুই ঘর পর্যন্ত প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট:

```
4
2 50 80
7 56 78 43 87 66 49 88
1 90
3 50 50 100
```

আউটপুট:

```
65.00
66.71
90.00
66.67
```

প্রোগ্রামিং সমস্যা ৯ - পরীক্ষার খাতায় মৌলিক সংখ্যা

"মৌলিক সংখ্যা হচ্ছে সেসব সংখ্যা যারা 1-এর চেয়ে বড় পূর্ণসংখ্যা এবং সেটি কেবল 1 এবং ওই সংখ্যাটি দ্বারাই নিঃশেষে বিভাজ্য হবে।" তোমার ক্লাসের ছাত্র-ছাত্রীদের মধ্যে কয়জনের গনিত পরীক্ষায় প্রাপ্ত নম্বর মৌলিক সংখ্যা তা বের করতে হবে।

ইনপুট

প্রথম লাইনে একটি সংখ্যা থাকবে। তারপর সেই সংখ্যাটির মান যত, ততটি লাইন থাকবে। প্রতি লাইনে ০ থেকে ১০০-এর মধ্যে ১০ টি করে সংখ্যা দেয়া থাকবে যা গনিত পরীক্ষায় ১০ জনের প্রাপ্ত নম্বর।

আউটপুট

প্রতি লাইনে একটি করে সংখ্যা থাকবে যা মোট কতজন এর প্রাপ্ত নম্বর মৌলিক সংখ্যা প্রকাশ করবে।

উদাহরণ

ইনপুট:

```
4
71 99 81 5 43 29 100 51 65 22
55 26 31 67 17 88 93 47 75 1
62 86 9 25 14 36 49 56 77 95
11 97 79 42 33 23 7 84 59 61
```

আউটপুট:

```
4
4
0
7
```

প্রোগ্রামিং সমস্যা ১০ - অঙ্ক গণনা

একটি অঋণাত্মক পূর্ণ সংখ্যা দেওয়া থাকবে। বলতে হবে সংখ্যাটি প্রকাশ করতে সর্বনিম্ন কতগুলো দশমিক অঙ্ক লাগবে।

ইনপুট
প্রথম লাইনে একটি সংখ্যা থাকবে। ওই সংখ্যার মান যত, এর পরে ততগুলো লাইনে একটি করে ইনপুট সংখ্যা থাকবে। প্রতিটি সংখ্যা 10000001-এর চেয়ে ছোট।

আউটপুট
প্রতিলাইনের জন্য সেই লাইনে দেওয়া সংখ্যাটি প্রকাশ করতে সর্বনিম্ন কতগুলো দশমিক অঙ্ক লাগবে তা প্রিন্ট কতে হবে।

উদাহরণ
ইনপুট:

```
4
231
007
4341
000
```

আউটপুট:

```
3
1
4
1
```

প্রোগ্রামিং সমস্যা ১১ - লেফট-রাইট

একটি স্ট্রিং দেওয়া থাকবে। স্ট্রিংটি কিছু দশমিক ডিজিট এবং L, R এই দুইটি ক্যারেকটার দিয়ে গঠিত। স্ট্রিংটির যেসব অবস্থানে L পাওয়া যাবে সেগুলোকে তার ঠিক বামের ক্যারেকটার দিয় বদলে ফেলতে হবে। এবং যেসব স্থানে R পাওয়া যাবে সেগুলোকে বদলে ফেলতে হবে তার ডানের ক্যারেকটার দিয়ে। অর্থাৎ, ইনপুট 34R92L6 থাকলে হয়ে যাবে 3499226।

ইনপুট
প্রথম লাইনে একটি সংখ্যা থাকবে। ওই সংখ্যার মান যত, এর পরে ততগুলো লাইনে একটি করে ইনপুট স্ট্রিং থাকবে। স্ট্রিংটি শুরু হবে একটি ডিজিট দিয়ে এবং শেষ হবে একটি ডিজিট দিয়ে। মধ্যবর্তী কোনো স্থানে ডিজিট ব্যতীত অন্য কোনো ক্যারেকটার পাশাপাশি থাকবে না। প্রতিটি ইনপুট স্ট্রিং এর দৈর্ঘ্য 50 বা তার কম হবে।

আউটপুট
প্রতিলাইনের জন্য সেই লাইনে দেওয়া স্ট্রিংটিকে নিয়ম অনুযায়ী পরিবর্তন করলে যে নতুন স্ট্রিং পাওয়া যাবে সেটি প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট:
```
5
0L7
4R5L9
71
8R4R0
34R92L6
```

আউটপুট:
```
007
45559
71
```

```
84400
3499226
```

প্রোগ্রামিং সমস্যা ১২ - জোড়-বিজোড়-১

কোনো একটি পূর্ণসংখ্যা দেওয়া থাকলে সেটি জোড় না বিজোড় তা বের করতে হবে।

ইনপুট

প্রথম লাইনে একটি সংখ্যা থাকবে। সংখ্যাটির মান যত, ততটি লাইনে একটি করে পূর্ণসংখ্যা (0 থেকে 2147483647-এর মধ্যে) দেওয়া থাকবে।

আউটপুট

প্রতিটি পূর্ণসংখ্যার জন্য, সংখ্যাটি জোড় হলে even আর বিজোড় হলে odd প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট:

```
3
100
0
1111
```

আউটপুট:

```
even
even
odd
```

প্রোগ্রামিং সমস্যা ১৩ - জোড়-বিজোড়-২

কোনো একটি পূর্ণসংখ্যা দেওয়া থাকলে সেটি জোড় না বিজোড় তা বের করতে হবে।

ইনপুট
প্রথম লাইনে একটি সংখ্যা থাকবে। সংখ্যাটির মান যত, ততটি লাইনে একটি করে অঋণাত্মক পূর্ণসংখ্যা দেওয়া থাকবে। একটি সংখ্যায় সর্বোচ্চ একশটি অঙ্ক থাকতে পারে।

আউটপুট
প্রতিটি পূর্ণসংখ্যার জন্য, সংখ্যাটি জোড় হলে even আর বিজোড় হলে odd প্রিন্ট করতে হবে।

উদাহরণ
ইনপুট:

```
3
100
0
1111
```

আউটপুট:

```
even
even
odd
```

প্রোগ্রামিং সমস্যা ১৪ - বাক্স

তোমার হাতে যথেষ্ট পরিমাণে কাজকর্ম নেই দেখে তোমাকে একটি বাক্স আঁকার কাজ দেওয়া হলো। আসলে ব্যপারটি তেমন কিছু কঠিন নয়, তোমাকে বর্গের একটি বাহুর দৈর্ঘ্য বলা হবে আর তুমি চট করে * অক্ষরটি ব্যবহার করে ওই বর্গটি এঁকে ফেলবে। বর্গের কেবল বাহু আঁকলেই হবে না, ভিতরের ঘরগুলোও * অক্ষরটি দিয়ে পূর্ণ করে দিতে হবে। যেহেতু তুমি প্রোগ্রামিং শেখা শুরু করেছ এবং লুপ পর্যন্ত শিখে ফেলেছ, তাই তুমি কাজটি করবে একটি প্রোগ্রাম লিখে।

ইনপুট

প্রথম লাইনে একটি সংখ্যা থাকবে - N এবং তারপরে N-সংখ্যক লাইনে প্রতি লাইনে একটি করে সংখ্যা (M) থাকবে যার মান 1 থেকে 100-এর মধ্যে হবে।

আউটপুট

প্রতিটি M-এর জন্য M X M বর্গ আঁকতে হবে। পুরো বর্গটি * দিয়ে পূর্ণ করে দিতে হবে। প্রতিটি বর্গ একটি ফাঁকা লাইন দিয়ে পৃথক করে দিতে হবে। পৃথক করার কাজে ব্যতিত অন্য কোথাও অতিরিক্ত ফাঁকা লাইন রাখা যাবে না।

উদাহরণ

ইনপুট:

3

1

3

5

আউটপুট:

*

***

***

***

*****

*****

*****

*****

*****

প্রোগ্রামিং সমস্যা ১৫ - অঙ্কমিল

দুইটি দুই অঙ্কের সংখ্যা দেওয়া থাকবে। বলতে হবে কোন অঙ্কগুলো উভয় সংখ্যাতেই আছে। যদি তেমন কোনো অঙ্ক না থাকে, তাহলে N প্রিন্ট করতে হবে।

ইনপুট

প্রথম লাইনে একটি সংখ্যা থাকবে। ওই সংখ্যার মান যত, এর পরে ততগুলো লাইনে দুইটি করে দুই অঙ্কের সংখ্যা থাকবে। সংখ্যাদুটি একটি স্পেস দ্বারা আলাদা।

আউটপুট

প্রতি লাইনের জন্য সেই লাইনে দেওয়া সংখ্যাদুটির মধ্যে মিলে যাওয়া অঙ্ক গুলো প্রিন্ট করতে হবে। যদি একাধিক ভিন্ন অঙ্ক মিলে যায় তাহলে ছোট থেকে বড় ক্রমে প্রিন্ট করতে হবে। আর কোনো মিল পাওয়া না গেলে প্রিন্ট করতে হবে N ।

উদাহরণ

ইনপুট:

```
5
12 25
33 33
74 86
90 90
65 56
```

আউটপুট:

```
2
3
N
09
56
```

প্রোগ্রামিং সমস্যা ১৬ - টমি মিয়ার প্রোবাবিলিটি

টমি মিয়া প্রোবাবিলিটিতে (সম্ভাব্যতা) বিশ্বাস করে। ওকে কেউ যখন জিজ্ঞেস করে, "তুমি ডেমোক্রেসিতে বিশ্বাস করো? তুমি খোদায় বিশ্বাস করো? তুমি ... ", টমি মিয়া চোখ বন্ধ করে দুইপাশে গম্ভীরভাবে মাথা নেড়ে বলে, 'উহু, আমি শুধু প্রোবাবিলিটিতে বিশ্বাস করি'। বেশিরভাগ মানুষ প্রোবাবিলিটি বুঝে না। সেজন্য ওরা আর কথা বাড়ায় না।

তো কেউই খুব অবাক হলো না যখন টমি মিয়া তার বড় ছেলেকে ইংলিশ মিডিয়াম স্কুলে ভর্তি করানোর জন্য প্রিপারেশন নেয়া শুরু করলো। এখন সমস্যা হচ্ছে, ভর্তি পরীক্ষায় ইংরেজি অনুবাদ করতে হবে। টমি মিয়ার ছেলে শব্দগুলোর ইংরেজি অনুবাদ জানে, কিন্তু সে ব্যাকরণ জানে না। এখন সমস্যা হচ্ছে ব্যাকরণ না মানলে ঠিক বাক্য গঠন হয় না। যেমন ধরো, 'তুমি ভাত খাও' এটা তুমি যদি ইংরেজিতে অনুবাদ করে লিখো, 'rice eat you' তাহলে কেউ ভাববে না তুমি কবি। সবাই ভাববে তুমি ব্যাকরণ জানো না, কিংবা ভাববে তুমি চাচ্ছো ভাত তোমাকে খেয়ে ফেলুক!

টমি মিয়া প্রোবাবিলিটিতে বিশ্বাস করে। তো প্রতিটা বাক্যের জন্য টমি মিয়া জানতে চায় তার বড় ছেলের সঠিক হবার প্রোবাবিলিটি কতো।

ইনপুট

প্রথম লাইনে একটি পূর্ণসংখ্যা থাকবে। সংখ্যাটির মান যত, প্রথম লাইনের পরে ততটি লাইন থাকবে এবং প্রতি লাইনে একটা ইংরেজি বাক্য - সেখানে দশটার বেশি শব্দ থাকবে না। প্রতি শব্দে ২০টির বেশি অক্ষর থাকবে না।

আউটপুট

প্রতি লাইনে অনুবাদ সঠিক হবার প্রোবাবিলিটি প্রিন্ট করতে হবে। তুমি ধরে নিতে পারো, প্রতিটা বাক্যের জন্য শুধু একটা মাত্র সঠিক অনুক্রম আছে। অনুক্রম মানে হচ্ছে order। আউটপুট দেখাতে হবে 1/n আকারে যেখানে n একটি পূর্ণসংখ্যা।

উদাহরণ

ইনপুট:

```
2
eat you rice
no way no good
```

আউটপুট:

```
1/6
1/12
```

প্রোগ্রামিং সমস্যা ১৭ - শব্দ ব্যাপারি

আবুল পত্রিকার বিজ্ঞাপন বিক্রি করে। ব্যাপারটা তোমাকে কিভাবে বুঝাই? ধরো তুমি পত্রিকায় বিজ্ঞাপন দিতে চাও "পাত্রী চাই। ছয় ফিট উঁচু কানা বগার জন্য তিন ফিট নিচু উচ্চ মধ্যমার নিম্নবিত্ত পাত্রী দরকার। কানা বগা নিম্নমুখী ধারার উর্ধমুখীতা বুঝিতে সক্ষম এবং দুই মিনিটে যে কোন সমস্যার চটুল সমাধান দিতে পারে।" তাহলে আবুল তোমার শব্দগুলো গুনবে। তারপর তোমাকে দাম বলবে - অত টাকা না দিলে সে বিজ্ঞাপন পত্রিকায় পাঠাবে না। প্রতি শব্দের জন্য আবুল ৪২০ টাকা নেয়। (আমি জানি না কেন। আমি সত্যি জানি না!) তুমি জানো তোমার বিজ্ঞাপন কি হবে। একটা প্রোগ্রাম লিখো বাকি পৃথিবীর সবার জন্য, যারা শব্দ গুনতে জানে না কিন্তু জানতে চায় ওদের বিজ্ঞাপনের জন্য আবুল কত দর হাকাতে পারে।

ইনপুট

প্রথম লাইনে একটি পূর্ণ সংখ্যা থাকবে। তারপরে ওই পূর্ণসংখ্যার মান যত, ততটি লাইন থাকবে। প্রতি লাইনে অনেকগুলো ইংরেজি শব্দ থাকবে (সর্বোচ্চ ১০০০টি)। সব ছোট হাতের অক্ষরে লেখা। শব্দগুলোর মাঝে এক বা একাধিক স্পেস থাকবে। কোন কোন শব্দের শেষে ফুল স্টপ থাকতে পারে।

আউটপুট

প্রতি লাইন ইনপুটের জন্য বিজ্ঞাপনের মূল্য প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট:

```
2
a little pink cat is looking for a mouse to play an online multiplayer game. please find
her a mouse. she promises not to eat it or chase it. she is a good pink cat.
this is a test
```

আউটপুট:

```
14700
1680
```

প্রোগ্রামিং সমস্যা ১৮ - জামা সঙ্কট

নিতাই নিতাইগঞ্জ গিয়ে দেখলো সবাই লাল শার্ট পড়ে আছে। পরদিন ঘুম থেকে উঠে দেখলো সবাই নীল শার্ট পড়ে আছে। পরদিন আবার ঘুম থেকে উঠে দেখে সবাই আবার লাল শার্ট পড়ে আছে। ও চায়ের দোকানে গিয়ে চায়ের অর্ডার দিয়ে দোকানীকে জিজ্ঞেস করলো, 'ব্যাপারটা কি?'। তখন দোকানদার ওকে বললো ওদের সবার দুইটা করে শার্ট। একদিন সবাই একসাথে লাল জামা পড়ে। অন্যদিন সবাই একসাথে নীল জামা পড়ে। এতে নাকি তাদের মধ্যে একাত্নবোধ প্রবল হয়।

নিতাই একাত্নবোধের মানে জানে না। কিন্তু ও খুব চিন্তিত ওর ঈদের দিনের জামা নিয়ে। কারণ ওর নীল বা লাল কোনটার রঙের জামা নেই। আর ওর মাত্র একটা জামা কেনার পয়সা আছে। ঈদের দিন আসতে আরো n দিন বাকি। ঈদের দিন নিতাই কোন রঙের জামা পড়বে, যাতে ওকে একদম বেমানান না লাগে? মানে ধরো, ঈদের দিনে সবাই যদি নীল জামা পড়ে আর একা নিতাই যদি লাল জামা পড়ে থাকে তাহলে তো সমস্যা।

ইনপুট

প্রথম লাইনে একটি সংখ্যা থাকবে। তারপর, সংখ্যাটির মান যত, ততটি লাইন থাকবে। প্রতি লাইনে থাকবে একটি সংখ্যা n। n মানে হচ্ছে ঈদের দিনের কত দিন বাকি।

আউটপুট

ইনপুটের প্রতিটি n-এর জন্য প্রতি লাইনে একটি করে শব্দ প্রিন্ট করতে হবে। জামার রং যদি নীল হয় তাহলে blue। লাল হলে red।

উদাহরণ

ইনপুট:

```
2
1
134344
```

আউটপুট:

```
blue
red
```

'প্রোগ্রামিংয়ে হাতে খড়ি' কোর্সের ডিভিডি পাওয়া যাচ্ছে, বিস্তারিত জানা যাবে
http://dimikcomputing.com -এ

প্রোগ্রামিং সমস্যা ১৯ - পিরামিড

তুমি ফারাও-এর প্রধান আর্কিটেক্ট। এবং তোমার প্রধান কাজ হলো পিরামিড বানানো। অন্যদের সাথে তোমার বানানো পিরামিডের পার্থক্য হলো, তোমার বানানোগুলো তারকাখচিত। যেমন এইটা-

```
   *

  ***

 *****

*******
```

এটা একটা চারতলা পিরামিড। পিরামিডটাকে এমন ভাবে বানানো হয়েছে যেন ডান ও বাম পাশের ভারসাম্য ঠিক থাকে। সে জন্য এটার সবার নিচে আছে 7 টি তারকা। এর উপরের তলায় 5 টি তারকা, এবং বসানো হয়েছে একঘর ডানে সরিয়ে। এর উপরে আছে 3 টি ও 1 টি করে তারকাবিশিষ্ট তলা। অর্থাৎ, এমন ভাবে হিসাব করে পিরামিডটি বানাতে শুরু করতে হবে যেন, সবার উপরের তলায় থাকে মাত্র ১ টি তারকা এবং পিরামিডটির উচ্চতা ফারাও যত তলা চায় ঠিক তত তলাই হয়।

ইনপুট

প্রথম লাইনে একটি সংখ্যা থাকবে। ওই সংখ্যার মান যত, এর পরে ততগুলো লাইনে একটি করে ইনপুট সংখ্যা থাকবে, যা পিরামিডের তলা নির্দেশ করে। প্রতিটি সংখ্যা 1 থেকে 100 এর মধ্যে।

আউটপুট

প্রতি লাইনের জন্য সেই লাইনে দেওয়া সংখ্যাটির সমান সংখ্যক তলা বিশিষ্ট একটি পিরামিড আঁকতে হবে। পিরামিডটিকে নির্দিষ্ট আকৃতি দিতে যতগুলো স্পেস লাগে তার বাইরে বাড়তি কোনো স্পেস প্রিন্ট করা যাবে না। প্রতিটি পিরামিডের নিচে একটি ফাঁকা লাইন দিতে হবে, যেন পরবর্তী পিরামিডটা আলাদা থাকে।

উদাহরণ

ইনপুট:

```
4
5
1
4
2
```

আউটপুট:

```
    *

   ***

  *****

 *******

*********
```

```
*
```

```
   *
  ***
 *****
*******
```

```
 *
***
```

প্রোগ্রামিং সমস্যা ২০ – ম্যাজিক

একটা রোবট ম্যাজিক দেখতে গেছে। জাদুকর নানান রকম খেলা দেখাচ্ছে। এবং তা দেখে বাকি দর্শকরা মাঝে মধ্যেই চমৎকৃত হয়ে হাততালি দিচ্ছে। কিন্তু রোবটটা পড়েছে মুশকিলে। এইসব খেলার মধ্যে ঠিক কখন যে ম্যাজিকটা হচ্ছে, সেটা সে ধরতে পারছে না। যেমন এই খেলাটার কথাই ভাবা যাক। জাদুকর একটা থলি নিয়েছে। এবং খেলার শুরুতেই সবাইকে দেখিয়েছে যে থলিটা একেবারেই ফাঁকা। এরপর সে থলিতে কিছু বল রাখছে, এবং বের করছে। আর তাই দেখেই একটু পর পর তুমুল হাততালি দিচ্ছে বাকি দর্শক! এতসব শোরগোলে রোবটটি চরম বিভ্রান্ত। তোমার কাজ হলো, কখন ম্যাজিকটা হচ্ছে এটা বুঝতে এই বোকা রোবটটিকে সাহায্য করা।

রোবটটি অবশ্য কিছু কাজ এগিয়ে রেখেছে। যাদুকর যখনই থলিতে একটা বল রাখে, রোবটটি 1 লেখে। এবং যখনই কোনো বল বের করে তখন লেখে 0 । এভাবে পুরো খেলায় কতবার থলিতে বল রাখা হলো আর বের করা হলো, সেটাকে সে একটা স্ট্রিং দিয়ে প্রকাশ করে। যেমন, 10110 । এর মানে হলো যাদুকর থলিতে প্রথমে একটা বল রেখেছে, তারপর থলিথেকে একটা বল বের করেছে। তারপর পর পর দুইটা বল রেখেছে। এবং সব শেষে একটা বল বের করে নিয়েছে। যার মধ্যে ম্যাজিকের কিছু নেই।

কিন্তু যদি স্ট্রিংটা হতো এমন 1001 তাহলেই দেখ ম্যাজিক হচ্ছে। কারণ জাদুকর প্রথমে একটা বল রাখছে। তারপর একটা বল বের করে নিয়েছে। কিন্তু এর পর সে আবার একটা বল বের করছে! কিন্তু থলিটা তো শুরুতে ফাঁকাছিলো। এই বাড়তি বলটা এলো কোথা থেকে? এটাই হলো ম্যাজিক। দুঃখের বিষয় হলো, জাদুকরের খেলা দেখে রোবটটা এই স্ট্রিং টা বানাতে পারলেও সেখান থেকে কখন ম্যাজিক হচ্ছে তা বুঝতে পারছে না। তোমার কাজ সেই প্রোগ্রামটা লিখে দেওয়া।

ইনপুট

প্রথম লাইনে একটি সংখ্যা থাকবে। ওই সংখ্যার মান যত, এর পরে ততগুলো লাইনে একটি করে ইনপুট স্ট্রিং থাকবে। স্ট্রিংটি 1 ও 0 দিয়ে গঠিত এবং এর দৈর্ঘ্য 50 এর বেশি নয়।

আউটপুট

প্রতিটি ইনপুট স্ট্রিং এর জন্য যদি স্ট্রিংটিতে ম্যাজিক থাকে তাহলে প্রিন্ট করতে হবে MAGIC আর যদি ম্যাজিক না থাকে তাহলে প্রিন্ট করতে হবে NORMAL ।

উদাহরণ

ইনপুট:

```
4

10110

11001001

1001

1
```

আউটপুট:

```
NORMAL

MAGIC

MAGIC

NORMAL
```

প্রোগ্রামিং সমস্যা ২১ - গণক

আমাদের বিশ্বজিৎ একজন গণক! না, সে 'ভবিষ্যত' গণনা করা তথাকথিত গণক না, বরং তার শখ হলো গণনা করা। তাই সুযোগ পেলেই সে যে কোন কিছু গণনা শুরু করে দেয়, তা সে যতই তুচ্ছ হোক না কেন। তো আমাদের এই বিশ্বজিতের আর একটি ভালো অভ্যাস হলো সে প্রতিদিন কোন একটা সময়ে তার বাড়ির পাশে খাল পারের রাস্তা দিয়ে কিছুক্ষন হাঁটাহাটি করে, আর সেই রাস্তাটা লাল-সাদা ইটের টালি দিয়ে বাঁধাই করা। স্বভাবতই সে প্রায় প্রতিদিনই হাঁটার সময় চেষ্টা করে আজ কতগুলো টালি অতিক্রম করে যাচ্ছে গুণে ফেলতে। কিন্তু দুঃখের বিষয় হলো, প্রায় প্রতিদিনই সে ভুল করে ফেলে কারণ হাঁটতে গেলে সবসময় তো আর পায়ের নিচের টালি গুলোর দিকে তাকিয়ে থাকা যায় না, সামনের দিকেও তাকাতে হয়, পরিচিত কাউকে দেখলে তার দিকে তাকিয়ে একটু মুচকি হাসি দিতে হয়, ইত্যাদি করতে গিয়ে প্রতিবারই সে গণনার তাল হারিয়ে ফেলে। তাই বিশ্বজিৎ চিন্তা করছে তার এই টালি গণনার কাজটা অন্য কোন ভাবে করা যায় কিনা? এর সমাধান হিসেবে সে ঠিক করে নিয়েছে হাঁটার সময় সে প্রতি সেকেন্ডে এক কদম করে এগুবে এবং তার প্রতি পদক্ষেপের দৈর্ঘ্য সবসময় সমান রাখবে, সেই সমান দৈর্ঘ্যের প্রতি পদক্ষেপে কতগুলো করে টালি অতিক্রম হয় তা তো সহজেই গুণে নেয়া যায়। তাহলে সে যদি টালি বাঁধাই করা পথে হাঁটা শুরু করার আগ মুহূর্তের সময় টুকে নেয় (আগ মুহূর্তের সময় মানে সে যদি "১০ টা ১০ মি: ০ সে:" কে 'শুরুর সময়' হিসেবে লিখে তাহলে ১০টা ১০ মি: ১ সে: এ সে ১ পা এগুবে) এবং টালি পথে হাঁটার শেষ করার সময়টা নিয়ে নেয় (যদি ১০টা ১০মি: ০সে: এ হাঁটা শুরু করে ২ পা হেঁটেই তার হাঁটা শেষ করে তাহলে হাঁটার 'শেষ সময়' হবে ১০টা ১০মি: ২সে:) তাহলে কত সময় ধরে সে হেঁটেছে জেনে নিয়ে - সে টালি বাধাই করা পথে কতগুলো পদক্ষেপ দিয়েছে এবং প্রকারন্তরে সে মোট কতগুলো টালি অতিক্রম করেছে তা হিসাব করে ফেলা সম্ভব। বিশ্বজিৎ বেশ কিছুদিন খাতা-কলমে এ হিসাব করেছে কিন্তু বারবার একই রকম হিসাব করাটা কিছুটা একঘেয়েমি ব্যাপার হয়েগেছে তার জন্য। বিশ্বজিৎ শুনেছে তুমি কম্পিউটারের ভাষা জানো, তাই সে তোমার কাছে এসেছে তার এই প্রতিদিনের কাজটি করে দিতে পারে এরকম একটি কম্পিউটার প্রোগ্রাম লিখে দেয়ার জন্য।

ইনপুট

তোমার লেখা প্রোগ্রামটিকে প্রথমে একটি ধনাত্মক পূর্ণ সংখ্যা 'N' (N<20) ইনপুট হিসেবে গ্রহণ করতে পারতে হবে। যা দিয়ে বলে দেয়া হবে বিশ্বজিৎ তোমার প্রোগ্রামটি দিয়ে মোট কতগুলো হিসাব করতে চায়। এরপর প্রতিটি হিসাবের ইনপুট পরপর তিন লাইনে দেয়া হবে। ঐ তিন লাইনের প্রথম লাইনে থাকবে হাঁটা 'শুরুর সময়', ২য় লাইনে থাকবে হাঁটা 'শেষের সময়' এবং ৩য় লাইনে থাকবে একটি ধনাত্মক পূর্ণ সংখ্যা C (C<=5) যা দিয়ে প্রকাশ করা হবে বিশ্বজিৎ প্রতি পদক্ষেপে মোট কতগুলো টালি অতিক্রম করে

গিয়েছিল তার মান। হাঁটা শুরু এবং শেষের সময় প্রকাশ করা হবে মোট তিনটি সংখ্যা দিয়ে, সংখ্যা গুলোর স্বজ্জা হবে এরকম "hh mm ss", যেখানে hh (0<=hh<24) ঘড়ির ঘন্টার মান, mm (0<=mm<60) মিনিটের মান এবং ss (0<=ss<60) সেকেন্ডের মান নির্দেশ করে।

এখানে প্রোগ্রাম লেখার সুবিধার জন্য একটা ব্যাপার মনে রাখা যেতে পারে তা হলো, বিশ্বজিৎ কখনো ২ ঘন্টার বেশি সময় ধরে হাঁটে না।

আউটপুট

হাঁটাহাটির প্রতি তিন লাইন ইনপুট সেটের জন্য তোমার প্রোগ্রাম কে এক লাইনে একটি সংখ্যা আউপুট দেখাতে হবে, যা নির্দেশ করবে বিশ্বজিৎ মোট কত টালি অতিক্রম করে গেছে।

উদাহরণ

ইনপুট:

```
2

10 10 0

10 20 15

2

23 15 25

1 10 20

3
```

আউটপুট:

1230

20685

প্রোগ্রামিং সমস্যা ২২ - মৌলিক উৎপাদক

একটা সংখ্যাকে তার মৌলিক উৎপাদকগুলোতে বিশ্লেষণ করা - ছোট বেলায় এই কাজটা অনেকবার করেছ। তখন তো করতে খাতা আর কলমে, এখন সি দিয়ে কি সেটা করতে পারবে?

ইনপুটঃ

প্রতি লাইনে তোমাকে একটা পজিটিভ সংখ্যা N >= 2 দেওয়া হবে। শুধুমাত্র শেষ লাইনের সংখ্যাটি 2 থেকে ছোট হবে। এই লাইনটি দিয়ে বোঝানো হবে যে তোমার প্রোগ্রামকে এখন থামতে হবে। ধরে নাও N এর সর্বোচ্চ মান হতে পারবে 65536।

আউটপুটঃ

শেষ লাইন ছাড়া অন্য সব লাইনের জন্য তোমাকে সংখ্যাটিকে মৌলিক উৎপাদকে বিশ্লেষণ করে দেখাতে হবে। উৎপাদক গুলোকে অবশ্যই ছোট থেকে বড় ক্রমে সাজাবে, আর তার সাথে তাদের ঘাতও দেখাতে হবে। উদাহরণ থেকে আউটপুট ফরম্যাটিং দেখে নাও। আউটপুটে লাইনের শেষে অতিরিক্ত স্পেস দিও না যেন।

উদাহরণ

ইনপুট:

```
100

22

65536
```

```
65154

11

0
```

আউটপুট:

```
100 = 2^2 * 5^2

22 = 2^1 * 11^1

65536 = 2^16

65154 = 2^1 * 3^1 * 10859^1

11 = 11^1
```

প্রোগ্রামিং সমস্যা ২৩ - গ্রেড নির্ণয়

Marks	Grade
80-100	A+
75-79	A
70-74	A-
65-69	B+
60-64	B
55-59	B-
50-54	C
45-49	D
0-44	F

উপরের স্কেল অনুসরণ করে প্রদত্ত মার্কস এর গ্রেড নির্ণয় করতে হবে।

ইনপুট

ইনপুট ফাইলের প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T (T<=25). এরপরের T সংখ্যক প্রতিটি লাইনে একটি পূর্ণ সংখ্যা M (0 <= M <= 100) থাকবে, যা মার্কস নির্দেশ করবে। M কে যথাযোগ্য গ্রেডে পরিবর্তন করতে করতে হবে।

আউটপুট

প্রতিটি কেসের জন্য একটি করে কেস নম্বর প্রিন্ট করতে হবে যেখানে "Case X: " লেখা থাকবে, যেখানে X হচ্ছে 1 থেকে শুরু করে কেসের নম্বর। এরপরে প্রিন্ট করতে হবে যা কিনা প্রদত্ত মার্কস এর সমমানের গ্রেড।

উদাহরণ

ইনপুট:

```
3

80

42

56
```

আউটপুট:

```
Case 1: A+
```

```
Case 2: F

Case 3: B-
```

প্রোগ্রামিং সমস্যা ২৪ - ছোট থেকে বড়

তিনটি পৃথক সংখ্যা দেয়া থাকবে। এদেরকে ছোট থেকে বড় আকারে প্রিন্ট করতে হবে।

ইনপুট

ইনপুট ফাইলের প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T (T<=100). এরপরে প্রতিটি লাইনে তিনটি করে পূর্ণ সংখ্যা n1, n2, n3 থাকবে যারা প্রত্যেকে স্বতন্ত্র এবং 1000 এর সমান বা ছোট।

আউটপুট

প্রতিটি কেসের জন্য একটি করে কেস নম্বর প্রিন্ট করতে হবে। এরপরে প্রদত্ত তিনটি সংখ্যাকে ছোট থেকে বড় আকারে সাজিয়ে প্রিন্ট করতে হবে। পাশাপাশি দু'টি সংখ্যার মাঝে শুধুমাত্র একটি স্পেস প্রিন্ট করতে হবে। নমুনা আউটপুটে আরো বিস্তারিত দেখতে পারো।

উদাহরণ
ইনপুট:

```
3
3 2 1
1 2 3
10 5 6
```

আউটপুট:

```
Case 1: 1 2 3
Case 2: 1 2 3
Case 3: 5 6 10
```

প্রোগ্রামিং সমস্যা ২৫ - গুণিতক

1 থেকে N এর মধ্যে x এর সকল গুনিতক বের করতে হবে।

ইনপুট

ইনপুট ফাইলের প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T (T≤100). এরপরে T সংখ্যক টেস্ট কেস থাকবে যার প্রতিটিতে দুটি করে পূর্ণ সংখ্যা x এবং N থাকবে যেখানে 1 ≤ x≤ N ≤ 1000000.

আউটপুট

প্রতিটি লাইনে একটি করে কেস নম্বর প্রিন্ট করতে হবে এবং এরপরে 1 থেকে N এর মধ্যে x এর সকল গুনিতকগুলোকে ছোট থেকে বড় আকারে দেখাতে হবে। পাশাপাশি দু'টি সংখ্যাকে শুধুমাত্র একটি স্পেস দিয়ে আলাদা করতে হবে। নমুনা আউটপুটে আরো বিস্তারিত দেখতে পারো।

উদাহরণ
ইনপুট:

```
3
4 11
13 50
2 10
```

আউটপুট:

```
Case 1: 4 8
Case 2: 13 26 39
Case 3: 2 4 6 8 10
```

প্রোগ্রামিং সমস্যা ২৬ - **ডিজিটাল ত্রিভূজ**

এমনভাবে সংখ্যা প্রিন্ট করতে হবে যেনো আউটপুট ত্রিভুজের মতো দেখায়। প্রদত্ত দু'টি সংখ্যার ভিত্তিতে আউটপুট প্রিন্ট করতে হবে।

ইনপুট

ইনপুট ফাইলের প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T (T<=10), পরবর্তী T সংখ্যক লাইনের প্রতিটিতে দুটি করে পূর্ণ সংখ্যা থাকবে, D (0 <= D <= 9) এবং H (1 <= H <= 20), যেখানে D হচ্ছে সেই অঙ্কটি যাকে ব্যবহার করে ত্রিভুজ তৈরি করা হবে এবং H হচ্ছে ত্রিভুজের উচ্চতা।

আউটপুট

প্রতিটি কেসের জন্য প্রথম লাইনে কেস নম্বর প্রিন্ট করতে হবে যেখানে "Case X:" লেখা থাকবে, যেখানে X হচ্ছে 1 থেকে শুরু করে কেসের নম্বর। এর পরবর্তী লাইন থেকে নমুনা আউটপুটের মত করে ত্রিভুজ প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট:

```
2
1 4
3 5
```

আউটপুট:

```
Case 1:
1
11
111
1111
```

```
Case 2:
3
33
333
3333
33333
```

ওয়েবের প্রাথমিক ধারণা নিয়ে আয়োজিত অনলাইন কোর্সের ডিভিডি,
বিস্তারিত: http://dimikcomputing.com

প্রোগ্রামিং সমস্যা ২৭ - লুপের কারবার

সকল triplet (x, y, z) দেখাতে হবে যেখানে x ≤ A, y ≤ B, z ≤ C (0 ≤ A, B, C ≤ 20) এবং x < y < z ছোট থেকে বড় হিসেবে দেখাতে হবে।

ইনপুট

ইনপুট ফাইলের প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T (T≤20). এরপরে পরবর্তী T সংখ্যক লাইনের প্রত্যেকটিতে তিনটি পূর্ণ সংখ্যা A,B,C থাকবে। এদের সীমা উপরে দেয়া আছে।

আউটপুট

প্রতিটি কেসের জন্য শুরুতে কেস নম্বরটি প্রিন্ট করতে হবে "Case X:" ফরম্যাটে, যেখানে X হচ্ছে 1 হতে শুরু করে কেসের নম্বর। এরপরে প্রতিটি ট্রিপলেট প্রতি লাইনে ছোট থেকে বড় হিসেবে প্রিন্ট করতে হবে। ট্রিপলেটগুলি lexicographical ক্রমানুসারে ঊর্ধক্রমে প্রিন্ট করতে হবে। নমুনা আউটপুটে আরো বিস্তারিত দেখতে পারো।

বি:দ্র: Lexicographic ক্রম – একটি ট্রিপলেট (a1, b1, c1) আরেকটি ট্রিপলেট (a2, b2, c2) থেকে lexicographic ক্রমানুসারে ছোটো হবে যদি এবং কেবল যদি নিচের শর্ত গুলোর মাঝে কোনো একটা সত্যি হয়: (i) a1 < a2 (ii) a1 = a2 and b1 < b2 (iii) a1 = a2 and b1 = b2 and c1 < c2)

উদাহরণ

ইনপুট:

3

```
1 1 2

3 2 3

4 4 4
```

আউটপুট:

```
Case 1:

0 1 2

Case 2:

0 1 2

0 1 3

0 2 3

1 2 3

Case 3:

0 1 2

0 1 3

0 1 4
```

0 2 3

0 2 4

0 3 4

1 2 3

1 2 4

1 3 4

2 3 4

প্রোগ্রামিং সমস্যা ২৮ - ভাজক

একটি সংখ্যার সমস্ত গুণনীয়ক (ভাজক) বের করতে হবে।

ইনপুটঃ

ইনপুট ফাইলের প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T (T≤10). এরপরের পরবর্তী T সংখ্যক লাইনের প্রতিটিতে একটি করে পূর্ণ সংখ্যা N থাকবে, (1 ≤ N ≤ 100000)।

আউটপুটঃ

প্রতিটি কেসের জন্য একটি করে লাইন প্রিন্ট করতে হবে, শুরুতে কেইস নম্বর দিতে হবে।

এরপর N এর সকল গুণনীয়ক ছোট থেকে বড় হিসেবে দেখাতে হবে এবং প্রতিটি গুণনীয়ক শুধুমাত্র একবার দেখাতে হবে।

গুণনীয়ক গুলোকে শুধুমাত্র একটি স্পেস দিয়ে আলাদা করতে হবে এবং লাইনের শেষে কোনো অতিরিক্ত স্পেস থাকবে না।

উদাহরণঃ

ইনপুট:

```
3
6
15
23
```

আউটপুট:

Case 1: 1 2 3 6

Case 2: 1 3 5 15

Case 3: 1 23

প্রোগ্রামিং সমস্যা ২৯ - ত্রিভূজ প্রেম

ত্রিভুজের ভুমি এবং উচ্চতা দেয়া থাকবে, এর ক্ষেত্রফলের দ্বিগুন বের করতে হবে।

ইনপুটঃ

ইনপুট ফাইলের প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T (T≤50).

এরপরে T সংখ্যক লাইন থাকবে যার প্রতিটিতে দুটি করে পূর্ণ সংখ্যা B এবং H থাকবে যারা যথাক্রমে ভূমি এবং উচ্চতা নির্দেশ করে। এখানে 1 ≤ B, H ≤ 1000.

আউটপুটঃ

প্রতিটি কেসের জন্য একটি করে লাইন প্রিন্ট করতে হবে যেখানে "Case X: Y" লেখা থাকবে,

যেখানে X হচ্ছে 1 থেকে শুরু করে কেসের নম্বর এবং Y হচ্ছে ত্রিভুজটির ক্ষেত্রফলের দ্বিগুন।

উদাহরণঃ

ইনপুট:

```
3

6 12
```

```
31 56

335 501
```

আউটপুট:

```
Case 1: 72

Case 2: 1736

Case 3: 167835
```

প্রোগ্রামিং সমস্যা ৩০ - কোন ধারা?

তিনটি সংখ্যা দেওয়া থাকবে, তোমাকে বলতে হবে তারা কি সমান্তর ধারা, গুণোত্তর ধারা, উভয়টি হতে পারে, নাকি কোনটিই নয়।

(Given three numbers, find out if they from arithmetic progression, geometric progression, both, or none.)

ইনপুটঃ

প্রথম লাইনের টেস্ট কেসের সংখ্যা দেওয়া T ($T \le 50$)দেওয়া থাকবে, এর পরে T টি লাইন থাকবে। প্রত্যেক লাইনে তিনটি করে সংখ্যা n1, n2, n3 দেওয়া থাকবে
যেখানে $0 \le$ n1, n2, n3 ≤ 1000.

আউটপুটঃ

প্রতিটি টেস্ট কেসের জন্য তোমাকে এক লাইন করে প্রিন্ট করতে হবে, ফরম্যাট হবেঃ "Case X:

"যেখানে X হল কেস নম্বর (১ থেকে শুরু করে) এবং অংশটি তোমাকে তোমার উত্তর অনুযায়ী পরবর্তী চারটি স্ট্রিং এর কোন একটি প্রিন্ট করতে হবেঃ

"None", "Arithmetic Progression", "Geometric Progression", "Both"(অবশ্যই কোটেশন ছাড়া)।

স্যাম্পল ইনপুট আউটপুট সেকশনে আউটপুট কেমন হতে হবে তা দেখানো আছে।

উদাহরণঃ

ইনপুট:

```
4
1 2 5
1 3 5
2 4 8
2 2 2
```

আউটপুট:

```
Case 1: None
Case 2: Arithmetic Progression
Case 3: Geometric Progression
Case 4: Both
```

প্রোগ্রামিং সমস্যা ৩১ - **সময় বহিয়া যায়**

সেকেন্ডে সময় দেয়া থাকবে, একে বছর, মাস, দিন, ঘণ্টা, মিনিট ও সেকেন্ডে পরিণত করে দেখাতে হবে।

তুমি ধরে নিতে পারো যে ১ বছর = ১২ মাস, ১ মাস = ৩০ দিন, ১ দিন = ২৪ ঘন্টা, ১ ঘন্টা = ৬০ মিনিট, ১ মিনিট = ৬০ সেকেন্ড।

ইনপুটঃ

ইনপুট ফাইলের প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T ($T \leq 50$) ।

এরপরে T সংখ্যক লাইন থাকবে, যার প্রতিটিতে একটি ধনাত্মক পূর্ণ সংখ্যা, S অর্থাৎ সেকেন্ডের মান দেয়া থাকবে ($1 \leq S \leq 10^9$) ।

আউটপুটঃ

প্রতিটি কেসের জন্য শুরুতে কেসের নম্বর প্রিন্ট করবে। নিচের নমুনা ইনপুট ও আউটপুটের মত করে।

এরপরে যথাক্রমে বছর, মাস, দিন, ঘণ্টা, মিনিট ও সেকেন্ডের সংখ্যা প্রিন্ট করতে হবে।

যদি কোনটি 0 আসে তাহলে সেটি প্রিন্ট করতে হবে না। যদি কোনটি 1 এর চেয়ে বড় হয় তাহলে সেটিকে বহুবচনে প্রিন্ট করতে হবে, নাহলে একবচনে।

উদাহরণ স্বরূপ, যদি কোনটিতে 0 মিনিট আসে তাহলে তখন মিনিটের পার্টটি বাদ দিয়ে দিবে। যদি বছরের সংখ্যা 1 আসে তাহলে 1 year প্রিন্ট করবে, 1 years নয়।

যদি 10 মাস হয় তাহলে 10 months প্রিন্ট করবে, 10 month নয়। নমুনা আউটপুটে আরো বিস্তারিত দেখতে পারো।

উদাহরণঃ

ইনপুট:

```
6

1

59

60

395

3840305

31104000
```

আউটপুট:

```
Case 1: 1 second

Case 2: 59 seconds

Case 3: 1 minute

Case 4: 6 minutes 35 seconds

Case 5: 1 month 14 days 10 hours 45 minutes 5 seconds

Case 6: 1 year
```

প্রোগ্রামিং সমস্যা ৩২ - **দুটি আয়তক্ষেত্র**

বের করতে হবে দুটি অক্ষের সাপেক্ষে সমান্তরাল আয়তক্ষেত্র পরস্পরকে ছেদ করে কিনা।

একটি আয়তক্ষেত্র দুটি স্থানাংকের মাধ্যমে দেখানো হবে, নিচের বাম কোনার বিন্দু আর উপরে ডান কোনার বিন্দুর মাধ্যমে।

দুটি আয়তক্ষেত্র পরস্পরকে ছেদ করে যদি তাদের পরস্পরের মধ্যে অশুন্য ধনাত্মক সাধারণ এলাকা থাকে।

ইনপুটঃ

ইনপুট ফাইলের প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T ($T \leq 25$).

প্রতিটি টেস্ট কেসে দুটি লাইন থাকবে, যেখানে প্রথম লাইনে থাকবে চারটি পুর্ণ সংখ্যা x1, y1, x2, y2 এবং

দ্বিতীয় লাইনে থাকবে আরও চারটি পূর্ণ সংখ্যা x3, y3, x4, y4. এখানে (x1, y1) ও (x2, y2) হচ্ছে প্রথম আয়তক্ষেত্রের যথাক্রমে নিচের বাম কোনার ও উপরের ডান কোনার বিন্দুর স্থানাংক।

একই ভাবে (x3, y3) এবং (x4, y4) হচ্ছে দ্বিতীয় আয়তক্ষেত্রের যথাক্রমে নিচের বাম কোনার ও উপরের ডান কোনার বিন্দুর স্থানাংক।

তুমি নিশ্চিন্ত থাকতে পার যে, $x1 < x2$, $y1 < y2$, $x3 < x4$, $y3 < y4$ থাকবে এবং উভয় আয়তক্ষেত্র হচ্ছে অক্ষের সাপেক্ষে সমান্তরাল।

অর্থাৎ প্রতি আয়তক্ষেত্রের প্রতিটি বাহু হয় X অক্ষ অথবা Y অক্ষের সাপেক্ষে সমান্তরাল। স্থানাংকের পরম মান ≤100, অর্থাৎ -100 ≤ x1, y1, x2, y2, x3, y3, x4, y4 ≤ 100.

আউটপুটঃ

প্রতিটি কেসের জন্য একটি করে লাইন প্রিন্ট করতে হবে যেখানে "Case X: Yes" অথবা "Case X: No" লেখা থাকবে, কোন কোটেশন চিহ্ন ছাড়া।

এটি নির্ভর করবে প্রদত্ত আয়তক্ষেত্র গুলো পরস্পরকে ছেদ করে কিনা তার উপর যেখানে X হচ্ছে 1 থেকে শুরু করে টেস্ট কেসের সংখ্যা। নমুনা আউটপুটে আরো বিস্তারিত দেখতে পারো।

উদাহরণঃ

ইনপুট:

```
3

1  1  5  5

4  4  7  7

1  1  5  5

1  7  10  12
```

```
1 1 2 2
2 2 3 3
```

আউটপুট:

```
Case 1: Yes
Case 2: No
Case 3: No
```

প্রোগ্রামিং সমস্যা ৩৩ - অঙ্ক বদল

একটি অঋণাত্মক পূর্ণ সংখ্যা দেওয়া থাকবে। সংখ্যাটির যে অঙ্কগুলোর মান জোড় সেগুলোর সাথে 1 যোগ করে এবং যে অঙ্কগুলোর মান বিজোড়, সেগুলো থেকে 1 বিয়োগ করে একটি পরিবর্তিত সংখ্যা বানাতে হবে।

যেমন: 46709 এই সংখ্যাটি বদলে হবে 57618 ।

উল্লেখ্য, 0 একটি জোড় সংখ্যা।

ইনপুটঃ

প্রথম লাইনে একটি সংখ্যা থাকবে। ওই সংখ্যার মান যত, এর পরে ততগুলো লাইনে একটি করে ইনপুট সংখ্যা থাকবে।

প্রতিটি সংখ্যা 10000001-এর চেয়ে ছোট।

আউটপুটঃ

প্রতিলাইনের জন্য সেই লাইনে দেওয়া সংখ্যাটিকে নিয়ম অনুযায়ী পরিবর্তন করলে যে নতুন সংখ্যাটি পাওয়া যাবে সেটি প্রিন্ট করতে হবে।

উদাহরণঃ

ইনপুট:

5

231

117

0

8341

46709

আউটপুট:

320

6

1

9250

57618

প্রোগ্রামিং সমস্যা ৩৪ - শব্দ উল্টাও

প্রবলেমে তোমার কাজ হলো, একটা বাক্যের শব্দগুলোকে উল্টিয়ে দেওয়া।

শব্দ গুলো শুধু মাত্র ছোট হাতের বর্ণ দিয়ে গঠিত হবে এবং তারা একটি স্পেসের মাধ্যমে পৃথক থাকবে।

বাক্যের শুরুতে বা শেষে কোন স্পেস থাকবে না। উদাহরনস্বরূপ, বাক্যটি যদি হয় "sun rises in the east" তাহলে তোমার আউটপুট হবে "east the in rises sun"।

ইনপুটঃ

ইনপুট ফাইলের প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T (T≤20). এরপরে প্রতি লাইনে একটি করে বাক্য থাকবে।

বাক্যগুলোর দৈর্ঘ্য ১৫০ এর কম হবে।

আউটপুটঃ

প্রতি কেসের জন্য বাক্যটির শব্দগুলোকে উল্টিয়ে একটি লাইন প্রিন্ট করতে হবে। নমুনা আউটপুটে আরো বিস্তারিত দেখতে পারো।

উদাহরণঃ

ইনপুট:

2

```
sun rises in the east

hello world
```

আউটপুট:

```
east the in rises sun

world hello
```

প্রোগ্রামিং সমস্যা ৩৫ - বাইনারি

একটা পূর্ণসংখ্যা N দেওয়া থাকবে। তোমাকে N দৈর্ঘ্যের সব বাইনারি সংখ্যা প্রিন্ট করতে হবে।

ইনপুটঃ

ইনপুটের প্রথম লাইনে একটা সংখ্যা, T, দেয়া থাকবে।T নির্দেশ করে এরপরে কয়টি টেস্ট কেইস আছে।এর পরবর্তী T সংখ্যক লাইনের প্রত্যেকটিতে একটি করে সংখ্যা N (0<N≤10) দেয়া থাকবে।

আউটপুটঃ

প্রতি কেসের জন্য তোমাকে N দৈর্ঘ্যের সব বাইনারি সংখ্যা প্রিন্ট করতে হবে।

প্রিন্ট করার সময় ছোট থেকে বড় এই নীতি অনুসরন করতে হবে।

আরো বিস্তারিত উদাহরণের জন্যে নিচের নমুনা ইনপুট আউটপুট দেখতে পারো।

উদাহরণঃ

ইনপুট:

2

1

3

আউটপুট:

0

1

100

101

110

111

প্রোগ্রামিং সমস্যা ৩৬ - মৌলিক কী না

একটি সংখ্যা মৌলিক কিনা বের করতে হবে। মৌলিক সংখ্যা শুধুমাত্র ১ এবং নিজেকে দ্বারা বিভাজ্য।

ইনপুটঃ

ইনপুট ফাইলের প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T (T≤10), এরপরে T সংখ্যক লাইন থাকবে যাদের প্রতিটিতে একটি করে পূর্ণ সংখ্যা N(2 ≤ N ≤ 1000000000000) থাকবে।

আউটপুটঃ

প্রতিটি টেস্ট কেসের জন্য, যদি N মৌলিক হয়, প্রথমে প্রিন্ট করবে N, তারপরে " is a prime" স্ট্রিংটি কোনও কোটেশন ছাড়া প্রিন্ট করবে।

N মৌলিক না হলে প্রথমে প্রিন্ট করবে N, তারপরে " is not a prime" স্ট্রিংটি কোনও কোটেশন ছাড়া প্রিন্ট করবে। নমুনা আউটপুটে আরো বিস্তারিত দেখতে পারো।

উদাহরণঃ

ইনপুট:

```
3
2
6
11
```

আউটপুট:

```
2 is a prime
6 is not a prime
11 is a prime
```

প্রোগ্রামিং সমস্যা ৩৭ - খোঁজ দ্য সার্চ - ১

দুইটি স্ট্রিং দেওয়া থাকবে যার দ্বিতীয়টি প্রথমটির সাবস্ট্রিং।
খুঁজে বের করতে হবে প্রথমটিতে সাবস্ট্রিংটি সর্বপ্রথম কোথা থেকে শুরু হয়েছে।
উল্লেখ্য, কোনো স্ট্রিং এর একটানা কোনো অংশকে বলে তার সাবস্ট্রিং।
যেমন, banana এর একটা সাবস্ট্রিং ana এবং এটা সর্ব প্রথম শুরু হয়েছে 1 তম স্থান থেকে।
আরেকটি সাবস্ট্রিং ban যা শুরু হয়েছে 0 তম স্থান থেকে। ওদিকে anna প্রথম স্ট্রিংটির বৈধ সাবস্ট্রিং নয়।

ইনপুটঃ

প্রথম লাইনে একটি সংখ্যা থাকবে। ওই সংখ্যার মান যত, এর পরে ততগুলো লাইনে দুইটি করে স্ট্রিং থাকবে।
প্রতিটি স্ট্রিংএর দৈর্ঘ্য 128 এর কম এবং স্ট্রিং দুটি একটি স্পেস দিয়ে আলাদা।

আউটপুটঃ

প্রতিলাইনের জন্য সেই লাইনের দ্বিতীয় স্ট্রিংটি প্রথম স্ট্রিংতে সর্বপ্রথম কোথা থেকে শুরু হয়েছে তা বলতে হবে।

উদাহরণঃ

ইনপুট:

```
4

banana ana
```

```
banana ban
aquickbrownfoxjumpsoverthelazydog fox
foobar foobar
```

আউটপুট:

```
1
0
11
0
```

প্রোগ্রামিং সমস্যা ৩৮ - খোঁজ দ্য সার্চ - ২

দুইটি স্ট্রিং দেওয়া থাকবে যার দ্বিতীয়টি প্রথমটির সাবস্ট্রিং। খুঁজে বের করতে হবে প্রথমটিতে সাবস্ট্রিংটি কতবার আছে। উল্লেখ্য, কোনো স্ট্রিং এর একটানা কোনো অংশকে বলে তার সাবস্ট্রিং।
যেমন, banana এর একটা সাবস্ট্রিং ana এবং এটা সর্ব প্রথম শুরু হয়েছে 1-তম স্থান থেকে, আবার 3-তম স্থান থেকেও এটি আরেকবার আছে।

```
banana
 |||||
 ana||
   |||
   ana
```

অর্থাৎ, মোট 2 বার সাবস্ট্রিংটিকে পাওয়া যাচ্ছে। ওদিকে anna প্রথম শব্দটির কোনো বৈধ সাবস্ট্রিংই নয়।
তাই পাওয়া যাচ্ছে 0 বার।

ইনপুটঃ

প্রথম লাইনে একটি সংখ্যা থাকবে।
ওই সংখ্যার মান যত, এর পরে ততগুলো লাইনে দুইটি করে স্ট্রিং থাকবে।
প্রতিটি স্ট্রিংএর দৈর্ঘ্য ১২৮ এর কম এবং স্ট্রিং দুটি একটি স্পেস দিয়ে আলাদা।

আউটপুটঃ

প্রতিলাইনের জন্য সেই লাইনের দ্বিতীয় স্ট্রিংটি প্রথম স্ট্রিংটির মধ্যে কতবার আছে তা বলতে হবে।

উদাহরণঃ

ইনপুট:

```
5

banana ana

banana anna

fox  aquickbrownfoxjumpsoverthelazydog

ddddd ddd
```

```
foobar foobar
```

আউটপুট:

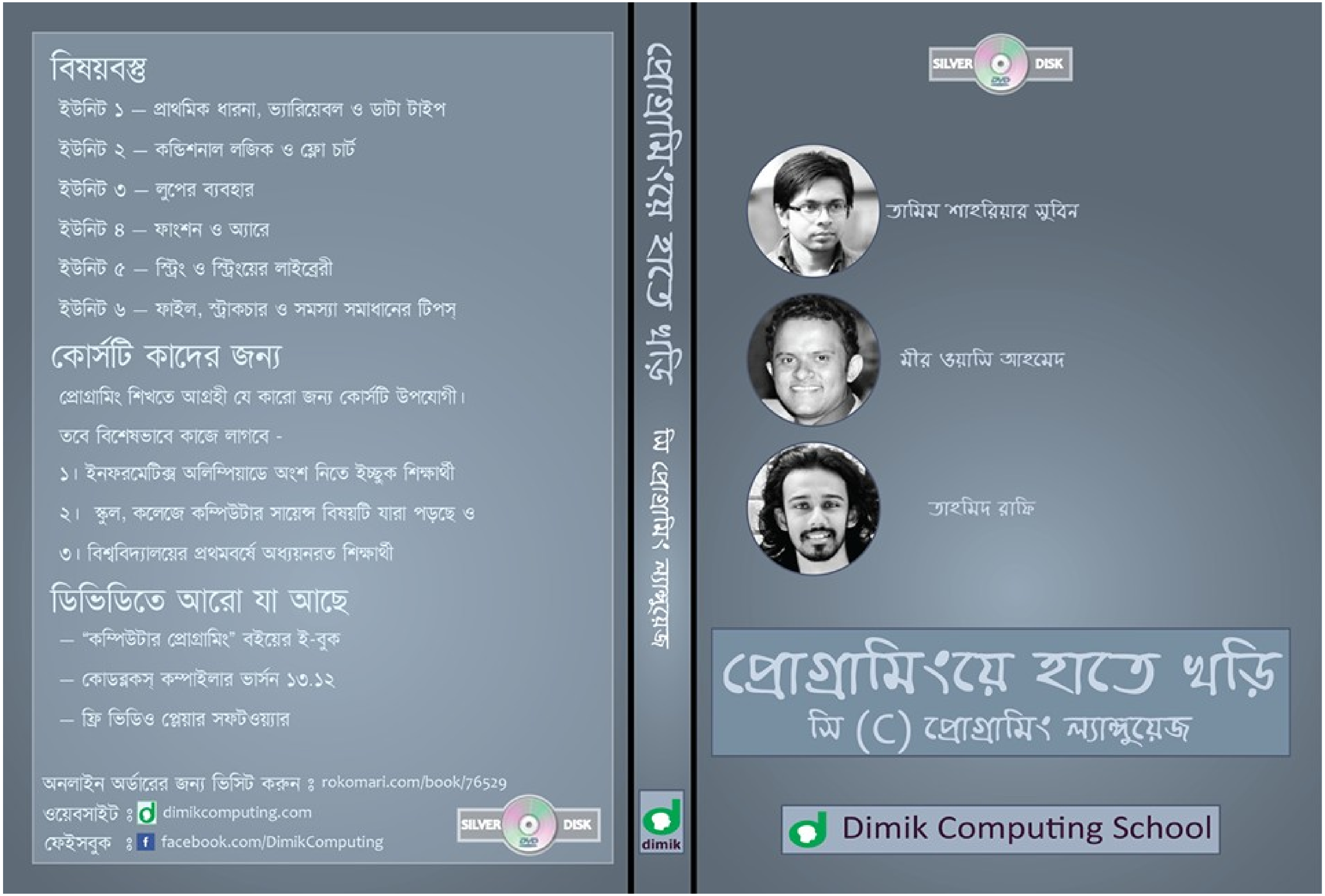
2

0

0

3

1



প্রোগ্রামিংয়ের উপর অনলাইন কোর্স-এর ডিভিডি, বিস্তারিত http://dimikcomputing.com, হোম ডেলিভারির জন্য ০১৫১৯৫২১৯৭১ নম্বরে কল করো।

প্রোগ্রামিং সমস্যা ৩৯ - রান রেট - ১

তুমি গেছ ক্রিকেট খেলা দেখতে। হঠাৎ খেয়াল করলে ইলেকট্রনিক স্কোরবোর্ডে আর সব সব ঠিক দেখালেও থাকলেও রানরেট ভুল দেখাচ্ছে। ব্যাপারটা ওদের বলতে গিয়ে জানলে যে ওরাও ভুলটা খেয়াল করেছে।
কিন্তু অস্ট্রেলিয়া থেকে আমদানি করা স্কোরবোর্ডের প্রোগ্রাম ঠিক করতে সেই দেশের ইঞ্জিনিয়ার আনতে হবে, তার জন্য মন্ত্রনালয় থেকে অনুমতি, অনুদান, সই, সিল, ছাপ্পর, এটা, সেটা অনেক কিছু লাগবে বলে কাজটা হয়ে উঠছে না।
কিন্তু তুমি বুঝতে পেরেছো ব্যাপারটা স্রেফ একটা প্রোগ্রামিং বাগ। দু লাইনের একটা কোড লিখেই ঠিক করে ফেলতে পারবে। সেটা বলতেই, কিছুটা ভেবে টেকনিশিয়ান লোকটা রাজি হয়ে গেল। শর্ত একটাই, তার বসকে কিছু বলা যাবে না।
তুমিও তোমার প্রোগ্রামিং স্কিল সত্যিকার একটা কাজে লাগানোর সুযোগ পেয়ে তুমুল উৎসাহে কোড করতে শুরু করে দিলে।
খেলাটা হচ্ছে 50 ওভারের ওয়ানডে ম্যাচ।
প্রতিবার যখন ডিস্প্লেতে ওভারপ্রতি বর্তমান রানের হার (current run rate) এবং জেতার জন্য কাঙ্ক্ষিত রানের হার (required run rate) দেখানো হয় তখন প্রতিপক্ষের করা মোট রান, ব্যাটসম্যানদের বর্তমান রান এবং খেলার আর কত বল বাকি আছে তা জানা থাকে।
উল্লেখ্য, ক্রিকেটে 6 বলে 1 ওভার হয় এবং জিততে হলে প্রতিপক্ষের মোট রানের চেয়ে অন্তত 1 রান বেশি করতে হয়।

ইনপুটঃ

প্রথম লাইনে একটি সংখ্যা থাকবে। ওই সংখ্যার মান যত, এর পরে ততগুলো লাইনে তিনটি করে সংখ্যা থাকবে।
প্রথম সংখ্যাটি প্রতিপক্ষের মোট রান, দ্বিতীয় সংখ্যাটি ব্যাটসম্যানদের বর্তমান রান, এবং তৃতীয় সংখ্যাটি খেলার আর কত বল বাকি আছে তা নির্দেশ করে।

আউটপুটঃ

প্রতিলাইনের জন্য সেই লাইনের দেওয়া তথ্য থেকে হিসাব করে প্রথমে ওভারপ্রতি বর্তমান রানের হার এবং এরপর একটি স্পেস দিয়ে কাংক্ষিত রানের হার প্রিন্ট করতে হবে।
প্রতিটি হার অবশ্যই দশমিকের পরে শুধু মাত্র দুই অঙ্ক পর্যন্ত দেখাতে হবে।

উদাহরণঃ

ইনপুট:

```
4

300 294 6

200 100 100

333 250 40

118 100 180
```

আউটপুট:

```
6.00 7.00

3.00 6.06

5.77 12.60

5.00 0.63
```

তুমি যদি এই পিডিএফ বইটি কারো কাছ থেকে কপি করে থাকো, কিংবা ইন্টারনেট থেকে ফ্রি ডাউনলোড করে থাকো, তুমি চাইলে লেখকের পরিশ্রমের মূল্য পরিশোধ করতে পারো। এই পিডিএফ বইটির দাম একেবারেই হাতের নাগালে। মাত্র **25** টাকা বিকাশ-এর মাধ্যমে পাঠিয়ে দাও এই নাম্বারেঃ **01622624182**, তোমরা যত বেশি বই কিনবে, সেটি হার্ডকপিই হোক, কিংবা ইবুক (যেমন এই পিডিএফ), লেখকরা তত বেশি উৎসাহ পাবেন এবং আরো বই লেখার ব্যাপারে আগ্রহী হবেন।

বিকাশের মাধ্যমে পেমেন্ট করার নিয়মঃ

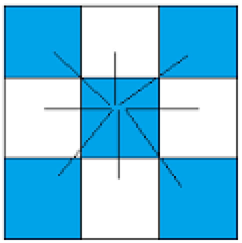
01. Go to your bKash Mobile Menu by dialing *247#
02. Choose “Payment”
03. Enter the Merchant bKash Account Number you want to pay to (01622624182)
04. Enter the amount you want to pay (25)
05. Enter a reference against your payment (cpbook pdf)
06. Enter the Counter Number (enter 0)
07. Now enter your bKash Mobile Menu PIN to confirm

১) প্রথমে *247# নাম্বারে ডায়াল করে বিকাশ মেনু আনতে হবে।
২) তারপরে 'পেমেন্ট' নির্বাচন করতে হবে।
৩) এবারে মার্চেন্ট বিকাশ একাউন্ট নাম্বার হিসেবে 01622624182 দিতে হবে।
৪) তারপরে টাকার পরিমাণ দিতে হবে (এই পিডিএফ বইয়ের জন্য ২৫ টাকা)
৫) রেফারেন্স হিসেবে cpbook pdf লিখে দিতে হবে।
৬) কাউন্টার নাম্বার ০ দিবে।
৭) এবারে তোমার বিকাশ মোবাইল মেনু পিন (PIN) দিতে হবে।

তোমার বিকাশ একাউন্ট না থাকলে একটি খুলে নিতে পারো (এটি খুলতে কোনো টাকা লাগে না) কিংবা অন্য কারো বিকাশ একাউন্ট ব্যবহার করতে পারো।

প্রোগ্রামিং সমস্যা ৪০ - দাবার বোর্ডে রাজার চাল

একটি অসীম আকারের দাবার বোর্ডে রাজার শুরুর অবস্থান (r1, c1) দেয়া আছে। বের করতে হবে, কত চালে দাবার রাজা (r2, c2) অবস্থানে যেতে পারবে।
যদি তোমার না জানা থাকে দাবার রাজার চাল কী, একটি দাবার রাজা এক চালে তার চারপাশের ৮ টি ঘরের যেকোনো একটি ঘরে যেতে পারে, নিচে দেখানো ছবির মত।



ইনপুটঃ

ইনপুট ফাইলের প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T (T≤50), এবং এর পরে T সংখ্যক লাইন থাকবে। প্রতি লাইনে 4 টি পূর্ণ সংখ্যা থাকবে: r1,c1,r2,c2, যেখানে, (r1,c1) হচ্ছে শুরুর ঘরের স্থানাংক আর (r2, c2) হচ্ছে শেষের ঘরের স্থানাংক।

(0 ≤ r1, c1, r2, c2 ≤ 1000).

আউটপুটঃ

প্রতিটি কেসের জন্য একটি করে লাইন প্রিন্ট করতে হবে যেখানে "Case X: Y" লেখা থাকবে, কোন কোটেশন চিহ্ন ছাড়া। এখানে X হচ্ছে 1 থেকে শুরু করে টেস্ট কেসের নম্বর এবং Y হচ্ছে (r1, c1) থেকে (r2, c2) তে পোঁছাতে দাবার রাজার প্রয়োজনীয় চালের সংখ্যা ।
নমুনা আউটপুটে আরো বিস্তারিত দেখতে পারো।

উদাহরণঃ

ইনপুট:

```
3
1 1 3 3
1 2 6 9
42 468 335 501
```

আউটপুট:

```
Case 1: 2
Case 2: 7
Case 3: 293
```