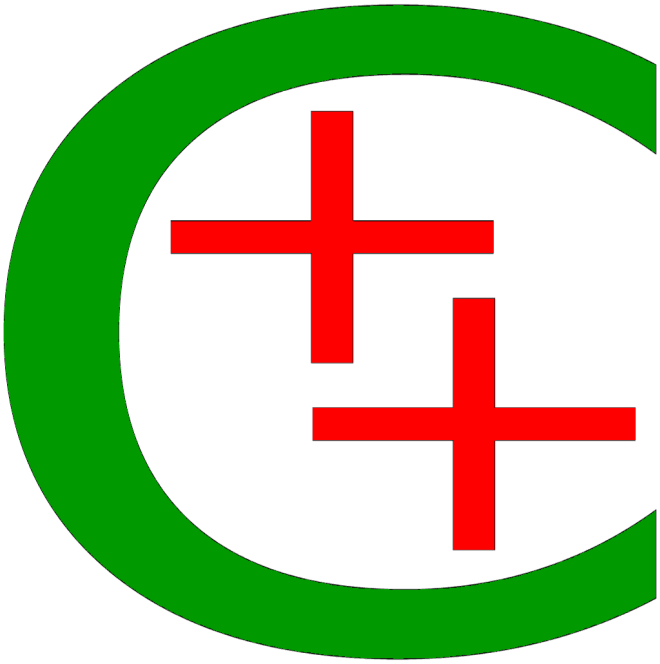
সিপিপি পরিগণনা

c++ programming

১ম খন্ড বাংলা (English)
২য় খন্ড English (বাংলা)



নিউটন মু. আ. হাকিম

সূচীপত্র

ফিরিস্তি তালিকা

খন্ড ১

বাংলা (English)

অধ্যায় ১

সিপিপিতে ক্রমলেখ রচনা (Writing c++ Programs)

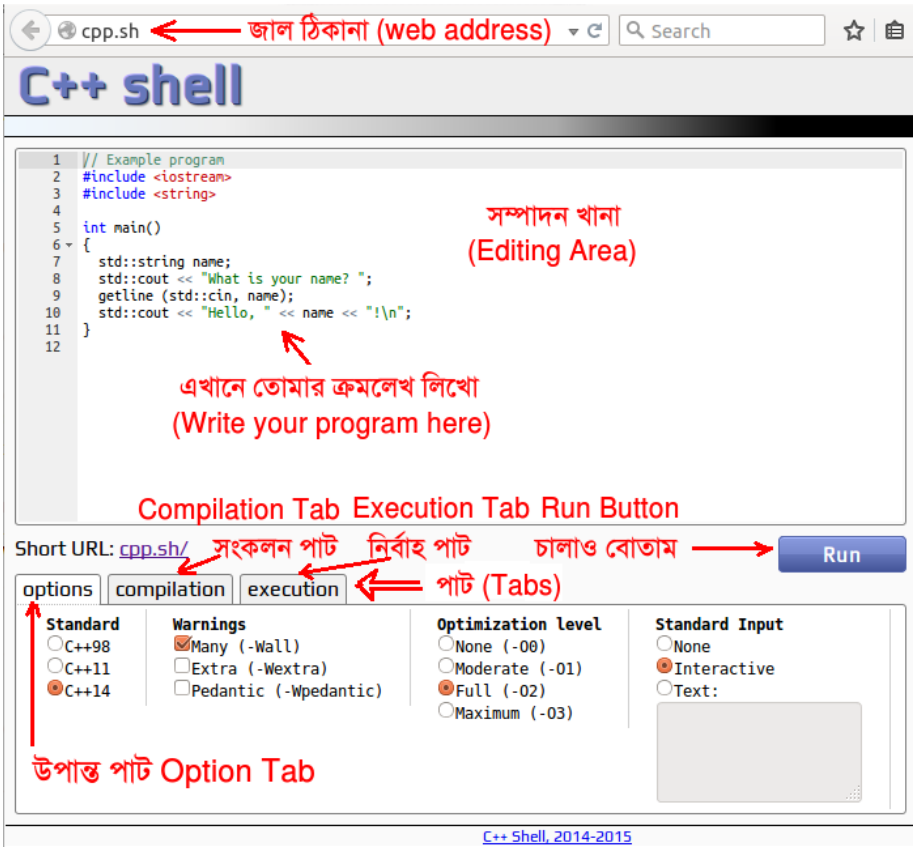
কোন পরিগণকের (programmer) কাছে নিজের লেখা ক্রমলেখ (program) একদম সন্তানের মতো। তিল তিল করে সময় নিয়ে পরিগণক একটি ক্রমলেখ গড়ে তোলে। যে সমস্যার জন্য ক্রমলেখ তৈরী করতে হবে, সেটা জানার পরে পরিগণক প্রথমে চিন্তা ভাবনা করে কী লিখবে, তারপর সেটা ক্রমলেখ রচনার যথাযথ নিয়ম মেনে লিখে ফেলে, তারপর সেটা চালিয়ে দেখে ঠিক ঠাক কাজ করে কি না। যদি ভুল কিছু থেকে থাকে, ভুলটা বের করে, সেটা ঠিক করে, তারপর আবার ক্রমলেখ চালিয়ে দেখে। এই চলতে থাকে যতক্ষণ না মন মতো সমস্যাটির সমাধান পাওয়া যাচ্ছে। আমরা সারা বইতে পড়বো ক্রমলেখতে কী লিখবো আমরা, আর যথাযথ ভাবে ক্রমলেখ রচনার নিয়ম কী। তবে এইখানে আলোচনা করবো, ক্রমলেখ লিখবো কোথায় আর সেটা চালাবো কী করে।

গণনিতে (computer) নির্বাহ (execution) করা জন্য আমরা যখন কোন একটি ক্রমলেখ (program) লিখতে চাই, তখন প্রথমে আমরা সেটা সম্পাদনা (edit) করি সাধারণত কোন একটা পরিগণনা ভাষায় (programming language)। এই পরিগণনা ভাষা ঠিক গণনিতে নির্বাহযোগ্য (executable) ভাষা নয়, আবার ঠিক মানুষের স্বাভাবিক ভাষাও (natural language) নয়, বরং এ দুটোর মাঝামাঝি কিছু একটা। পরিগণনা ভাষায় লিখিত আমাদের ক্রমলেখকে আমরা তাই এরপরে সংকলন (compile) করে যন্ত্রভাষায় (machine language) রূপান্তর করি যাতে গণনি সেটা বুঝতে পারে। তারপর রূপান্তরিত ক্রমলেখটিকে আমরা নির্বাহ (execute) করি।

১.১ হয়মান মন্ত্রপাতি (Online Software)

হয়মান (online) সম্পাদনা ও সংকলনের (editing and compilation) জন্য আমরা এখানে cpp.sh নামক জালপাতা (webpage) ব্যবহার করবো। তুমি খুঁজলে এরকম আরো অনেক জালপাতা পেতে পারো। যাই হোক তোমার আন্তর্জাল ব্রাজকে (internet browser) জাল ঠিকানা (web address) লিখবার জায়গায় cpp.sh লিখে তুমি উপরে উল্লেখিত ওই জালপাতায় যেতে পারো। তারপর ব্রাজকে (browser) ওই জালপাতা কেমন দেখা যাবে সেটা আমরা নীচের ছবিতে দেখতে পাবো। খেয়াল করো ওই ছবিতে বিভিন্ন অংশ তীর চিহ্ন দিয়ে চিহ্নিত করা হয়েছে। বড় সাদা অংশে সম্পাদন খানা (Editing Area) লেখা হয়েছে। সম্পাদন খানার নীচে বাম দিকে রয়েছে তিনটি পাট (tab): উপান্ত পাট (options tab), সংকলন পাট (compilation tab), নির্বাহ পাট (execution tab), আর ডান দিকে রয়েছে চালাও (run) বোতাম।

১.১. হয়মান মন্ত্রপাতি (Online Software)

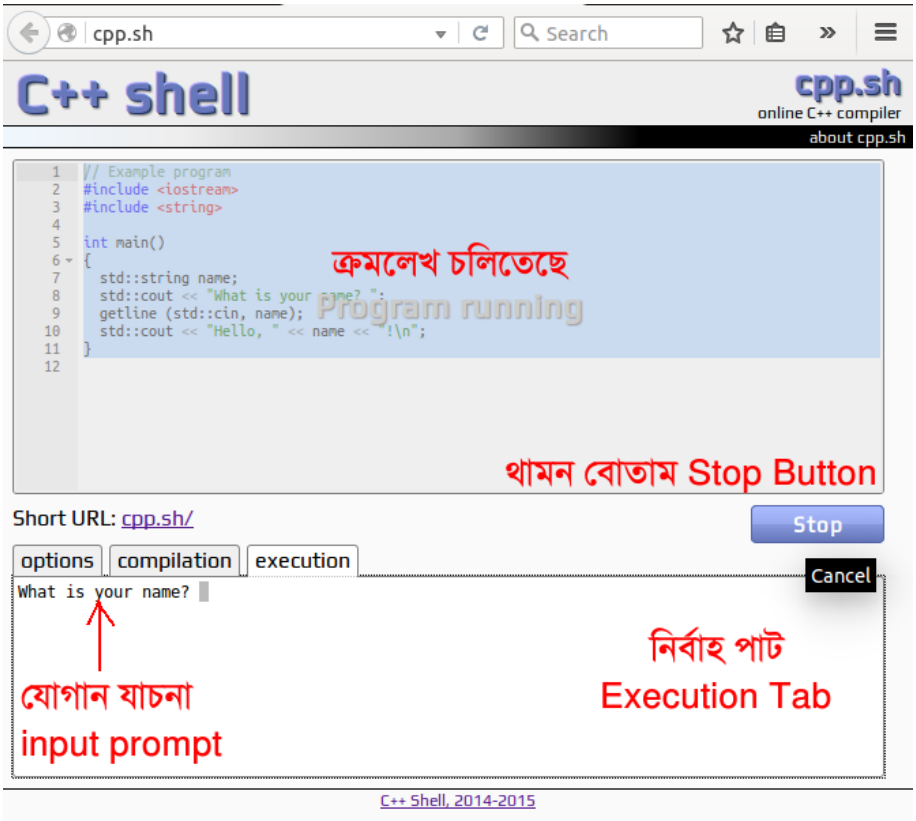


আমরা মুলত ক্রমলেখ রচনা ও সম্পাদনা করবো সম্পাদন খানায় (Editing Area)। নমুনা (sample) হিসাবে সম্পাদন খানায় আগে থেকে কিছু থাকতে পারে, তুমি সেগুলো মুছে দিতে পারো বা তোমার ক্রমলেখয়ের জন্য দরকার মতো বদলে নিতে পারো। cpp.sh জালপাতায় (webpage) গেলে সাধারণত নীচে দেখানো ক্রমলেখটিই (program) সেখানে থাকে। আমরা আপাতত উপান্ত পাটে (options tab) কোন পরিবর্তন না করে সরাসরি চালাও বোতামে (run button) টিপ দিয়ে নমুনা (sample) ক্রমলেখটিই চালাবো (run)।

```
// Example program
#include <iostream>
#include <string>

int main()
{
  std::string name;
  std::cout << "What is your name? ";
  getline (std::cin, name);
  std::cout << "Hello, " << name << "!\n";
}
```

ক্রমলেখ সম্পাদনা শেষ হলে অথবা মাঝামাঝি অবস্থাতেও পরীক্ষা করে দেখার জন্য আমরা সাধারণত চালাও বোতামে টিপ (click) দেবো। তাতে এক টিপেই প্রথমে ক্রমলেখ সংকলন

(compile) হবে তারপর নির্বাহ (execution) হবে। যখন ক্রমলেখ সংকলন হতে থাকবে তখন সম্পাদন খানার মাঝখানে দেখবে "অপেক্ষা করো সংকলন হচ্ছে" "Please Wait Compiling" লেখা আসবে। আর একই সাথে চালাও বোতামটি বদলে গিয়ে হয়ে যাবে বাতিল বোতাম (cancel button)। অনেক ক্ষেত্রে সংকলন হতে সময় লাগে, তুমি যদি কোন কারণে সংকলন বাতিল করতে চাও তাহলে বাতিল বোতামে টিপ দিলেই হবে। যখন সংকলন হতে থাকে তখন বাম দিকের পাটগুলো (tabs) খেয়াল করবে, উপান্ত পাটের (options tab) বদলে সংকলন পাট (compilation tab) সামনে চলে আসবে। সংকলনের সময় কোন ত্রুটি (error) পাওয়া গেলে সংকলন পাটে দেখা যাবে। আর কোন সংকলন ত্রুটি না থাকলে সংকলন সফল Compilation successful বার্তা দেখা যাবে সংকলন পাটে আর তারপর নির্বাহ পাট (execution tab) সামনে আসবে। নির্বাহ চলাকালীন সময়ে যোগান ও ফলন (input and output) নির্বাহ পাটে চলবে আর বাতিল বোতামটি (cancel button) বদলে হয়ে যাবে থামন বোতাম (stop button), যাতে যে কোন সময় নির্বাহ থামিয়ে দেয়া যায়। থামন বোতাম টিপলে অথবা নির্বাহ শেষ হয়ে গেলে আবার উপান্ত পাট (option tab) সামনে আসবে আর চালাও বোতাম (run button) ফিরে আসবে।

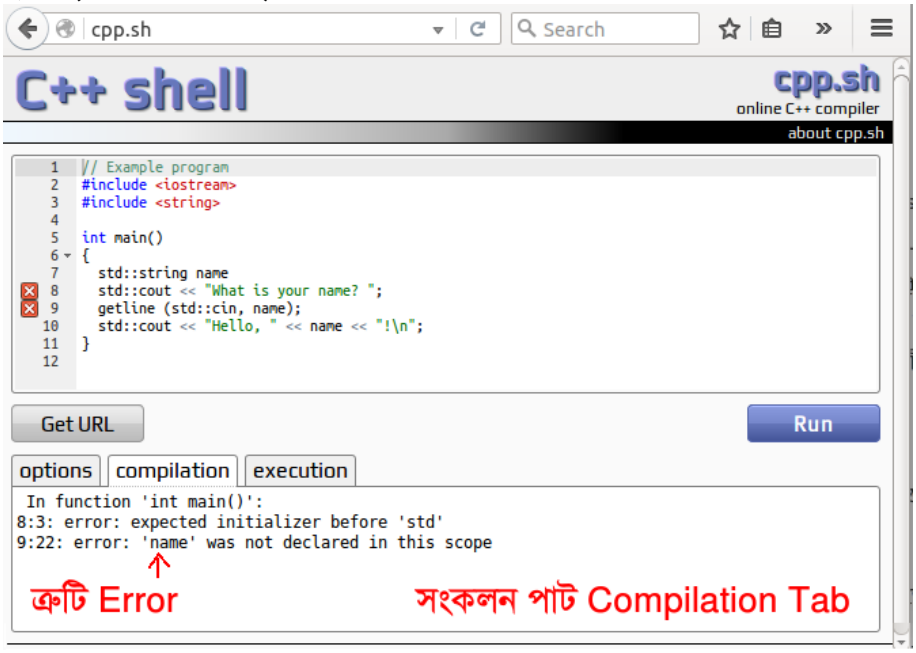


নমুনা ক্রমলেখটি আমাদের আপাতত বিস্তারিত বুঝার দরকার নাই, আমরা পরে ব্যাপারগুলো বিস্তারিত শিখবো। নমুনা ক্রমলেখটি চালালে উপরের ছবির মতো প্রথমে নির্বাহ পাটে (execution tab) দেখাবে What is your name? তখন তুমি যদি তোমার নাম লিখে দাও gonimia আর তারপর ভুক্তি (enter) চাপ দাও, তাহলে পরের সারিতে লেখা আসবে Hello, gonimia! নামটুকু নেওয়ার আগে What is your name? দেখানোকে আমরা বলি যোগান যাচনা (input prompt) আর নাম gonimia দেওয়াটাকে আমরা বলি যোগান (input) দেওয়া আর পরের সারিতে Hello, gonimia! দেখানোকে আমরা বলি ফলন (output) দেওয়া।

তো তুমি চালাও বোতামে (run button) টিপ দিয়ে দেখো কী হয়। প্রথমে সংকলন পাট হয়ে নির্বাহ পাটে (execution tab) গিয়ে উপরে যে ভাবে বলা হলো সে রকম হয় কী না দেখো। তোমার বোঝার সুবিধার্থে নির্বাহ পাটে শেষ পর্যন্ত কী থাকবে তা নীচে দেখানো হলো।

```
What is your name? gonimia
Hello, gonimia!
```

এবার আমরা একটু দেখি সংকলনে (compilation) ত্রুটি হলে কী ঘটে, আর আমাদের কী করতে হয়! এটার জন্য আমরা ইচ্ছে করে একটা ত্রুটি (error) তৈরী করে দেই। যেমন ধরো std::string name; লেখা রয়েছে যে সারিতে সেখানে একদম শেষ হতে দির্তি (semicolon) ; তুমি মুছে দাও। আর তারপর চালাও বোতামে (run button) টিপ দাও। দেখবে নীচের মতো করে ত্রুটি বার্তা দেখাবে সংকলন পাটে (compilation tab), আর সংকলন পাটই সামনে থাকবে, নির্বাহ পাট (execution tab) সামনে আসবে না।



সংকলন পাটে যে বার্তাগুলো আসবে তা নীচে দেখানো হলো। দ্বিতীয় সারিতে দেখো $8:3$ মানে বুঝাচ্ছে ৮ম সারিতে ত্রুটি আছে আর ৩য় অক্ষরে, আর ত্রুটিটা হলো ; থাকতে হবে। আসলে ; দরকার আমাদের ৭ম সারির শেষে। সাধারণত যে সারিতে ত্রুটি আছে বলা হয়, ত্রুটি সেই সারি বা আগের সারিতে থাকে। এখানে ; থাকায় সংকলক (compiler) আসলে ঠিক ৭ম আর ৮ম সারি নিয়ে কিঞ্চিত বিভ্রান্তিতে রয়েছে। ক্রমলেখ রচনার সময় আমরা নানান রকম ভুল ত্রুটি করি, তুমি ক্রমলেখ লেখার চর্চা করতে থাকলে এই ত্রুটিগুলোর সাথে পরিচিত হয়ে যাবে। তখন দেখা মাত্রই বুঝতে পারবে ভুলটুকু কী আর কী করে সেটা ঠিক করতে হবে। যাইহোক ত্রুটিটুকু বুঝতে পারলে আমরা সেটি ঠিক করে আবার চালাও বোতামে টিপ দিবো, আর তখন সফল ভাবেই নির্বাহিত হবে।

```
In function 'int main()':
8:3: error: expected initializer before 'std'
9:22: error: 'name' was not declared in this scope
```

সবশেষে আমরা উপান্ত পাট (options tab) সংক্ষেপে আলোচনা করবো। সেখানে থাকা নানা উপান্তগুলোর (option) কী কাজ মুলত সেটাই জানা আমাদের উদ্দেশ্য। তবে এগুলো নিয়ে আমরা

আপাতত পরীক্ষা-নিরীক্ষা করবো না, বরং যে রকম অবস্থায় আছে সে রকম অবস্থাতেই ক্রমলেখ (program) সম্পাদনা (editing), সংকলন (compile) ও নির্বাহ (execute) করবো।

১. সবচেয়ে বামের স্তম্ভে (column) দেখো প্রমিত (standard) উপাত্তগুলো রয়েছে। সিপিপি ভাষার নানান সংস্করণ (version) রয়েছে, তুমি চাইলে আগের সংস্করণ ব্যবহার করতে পারো, সাধারণত এখানে C++14 সংস্করণ নির্বাচন করা থাকে।

২. বাম থেকে দ্বিতীয় স্তম্ভে আছে সতর্কবার্তার উপাত্তগুলো, অর্থাৎ সংকলন (compile) করার সময় সংকলক (compiler) কতটা খুঁটি নাটি ত্রুটি ধরবে সেটা এখানে বলে দেয়া হয়। সাধারণত এখানে সব Many (-Wall) উপাত্ত নির্বাচিত থাকে।

৩. বামথেকে তৃতীয় স্তম্ভে আছে অনুকুলায়নের(optimisation) উপাত্তগুলো। একই ক্রমলেখ (program) সংকলক (compiler) চাইলে এমন ভাবে সংকলন (compile) করতে পারে যে ক্রমলেখটি অনেক দ্রুত নির্বাহ (execute) হবে, আবার এমন ভাবে সংকলন করতে পারে যে ক্রমলেখটি অনেক ধীরে নির্বাহ হবে। দ্রুত নির্বাহ হবে এমন সংকলন করতে স্বাভাবিক ভাবেই বেশী সময় লাগে, আর ধীরে নির্বাহ হবে সেরকম সংকলন করতে সময় কম লাগে। এখানে সাধারণত পূর্ণ Full (-O2) উপাত্ত নির্বাচিত থাকে।

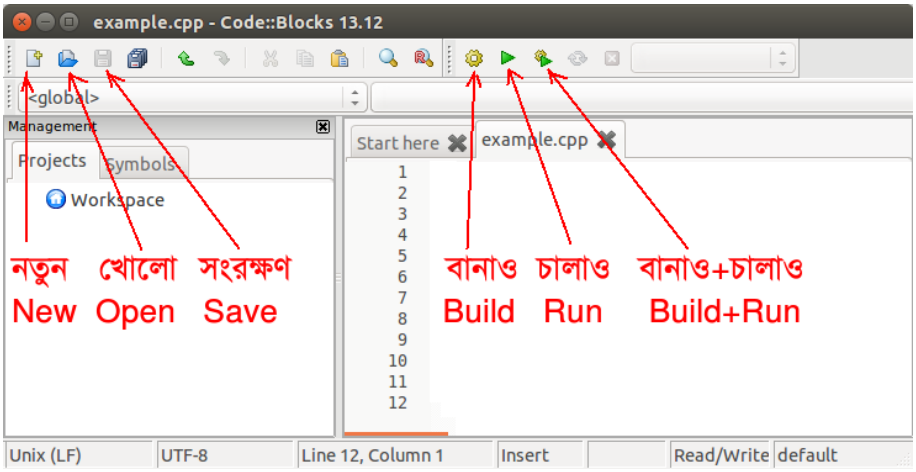
৪. সবচেয়ে ডানের স্তম্ভে আছে প্রমিত যোগান (standard input) উপাত্তসমুহ। সাধারণত এখানে মিথস্ক্রিয় (interactive) উপাত্ত নির্বাচিত থাকে যার অর্থ চাপনি (keyboard) ব্যবহার করে যোগান (input) দেওয়া যাবে। তোমার ক্রমলেখতে কোন যোগান না থাকলে তুমি কিছুনা (none) উপাত্ত নির্বাচন করতে পারো। অথবা তুমি যদি আগেই যোগান দিয়ে রাখতে চাও তাহলে পাঠনিক (text) উপাত্ত নির্বাচন করে ওইখানে থাকা বাক্সে আগে থেকে তোমার যোগানগুলো দিয়ে রাখতে পারো। তাতে ক্রমলেখ (program) চাপনি (keyboard) থেকে যোগান না নিয়ে ওইখান থেকে নিয়ে নিবে।

১.২ নয়মান মন্ত্রপাতি (Offline Software)

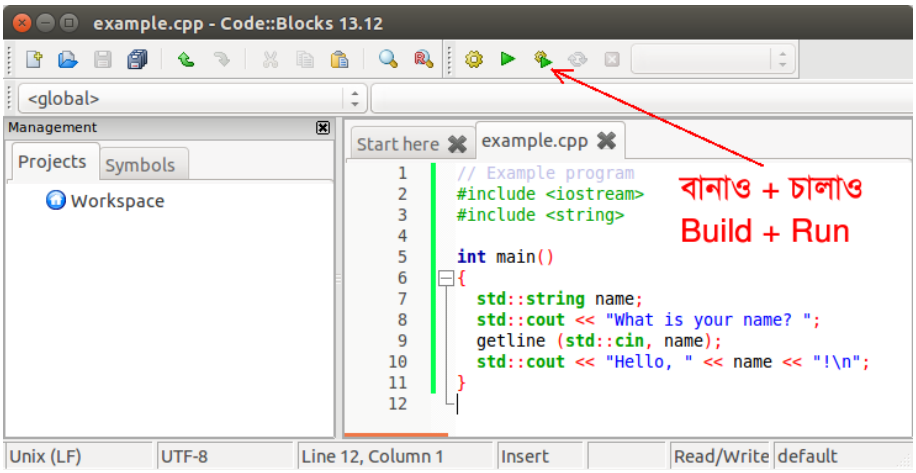
কোডব্লকস (Code::Blocks) একটি নয়মান মন্ত্রপাতি (offline software) যেটি লিনাক্স, উইন্ডোজ, ম্যাক ওএস, সব পরিচালনা তন্ত্রেই (operating system) ব্যবহার করা যায়। কোডব্লকসে তুমি c/c++ ক্রমলেখ সম্পাদনা ও সংকলন (editing and compilation) করতে পারবে। কোডব্লকস তোমার গণনিতে (computer) সংস্থাপন (install) করে নিলে তুমি আন্তর্জাল (internet) ব্যবহার করা ছাড়াই তোমার ক্রমলেখ সম্পাদনা ও সংকলন করে যেতে পারবে। কোডব্লকস পাওয়া যায় http://www.codeblocks.org/ জালপাতা (webpage) হতে, এর মন্ত্র তুমি http://www.codeblocks.org/downloads/binaries সূত্র হতে নামিয়ে নিতে পারো। তুমি উইন্ডোজ ব্যবহারকারী হলে codeblocks-13.12mingw-setup.exe সংস্করণ নামিয়ে সংস্থাপন (install) করবে। এই সংস্করণে GCC সংকলক (compiler) আর GDB আপদনাশক (debugger) আছে। তুমি লিনাক্স বা ম্যাক ব্যবহারকারী হলে দরকার মতো তোমার সংস্করণ নামিয়ে সংস্থাপন (install) করবে। গণনিতে (computer) কোডব্লকস সংস্থাপন (install) বিষয়ে সাহায্য পেতে চাইলে নীচের প্রথম দুটি সূত্র হতে ছবিও (video) দেখতে পারো আর ব্যবহার পুস্তিকা (user manual) পেতে চাইলে তা পেতে পারো তৃতীয় সূত্র হতে ।

উইন্ডোজ: https://www.youtube.com/watch?v=zOGU8fC3bvU
লিনাক্স: https://www.youtube.com/watch?v=3B4hPHZNtNw
ব্যবহার পুস্তিকা: http://www.codeblocks.org/user-manual

এরপরে তোমার গণনিতে (computer) পরিচালনা তন্ত্র (operating system) কী আছে তার জন্য দরকারী নির্দেশনা মেনে তুমি কোডব্লকস ঠিকঠাক মতো সংস্থাপন (install) করে নাও। সংস্থাপন হয়ে গেলে তারপর তুমি কোডব্লকস ক্রমলেখটি (program) চালাও (run)। দেখবে নীচের মতো জানালা (window) খুলে যাবে। তারপর তুমি প্রাপণ্য (menu) থেকে File এর অধীনে Newএর (নতুন) ভিতরে Empty Fileএ (ফাঁকা নথি) টিপ (click) দাও। নতুন নথিতে আপাতত কিছু থাকবে না। তারপর আবার প্রাপণ্য (menu) থেকে Save Fileএ টিপ দিয়ে দরকার মতো নথির নাম (File Name) যেমন example.cpp দিয়ে তোমার নতুন সৃষ্ট নথিটিকে সংরক্ষণ (save) করো। প্রাপণ্য (menu) থেকে এসব না করে নীচের ছবিতে দেখানো মূর্তি বোতামগুলোতে (icon button) টিপ দিয়েও তুমি তোমার নথি তৈরী ও তা নাম দিয়ে সংরক্ষণ করতে পারো।



এরপর example.cppতে নীচের মতো সংকেত (code) লিখো। এই ক্রমলেখটি আমাদের আপাতত বিস্তারিত বুঝার দরকার নাই, আমরা পরে ব্যাপারগুলো বিস্তারিত শিখবো। তবে সংক্ষেপে বলি এই ক্রমলেখ (program) তোমাকে জিজ্ঞেস করবে What is your name? আর তুমি তোমার নাম ধরো gonimia লিখে দিলে তখন বলবে Hello, gonimia!

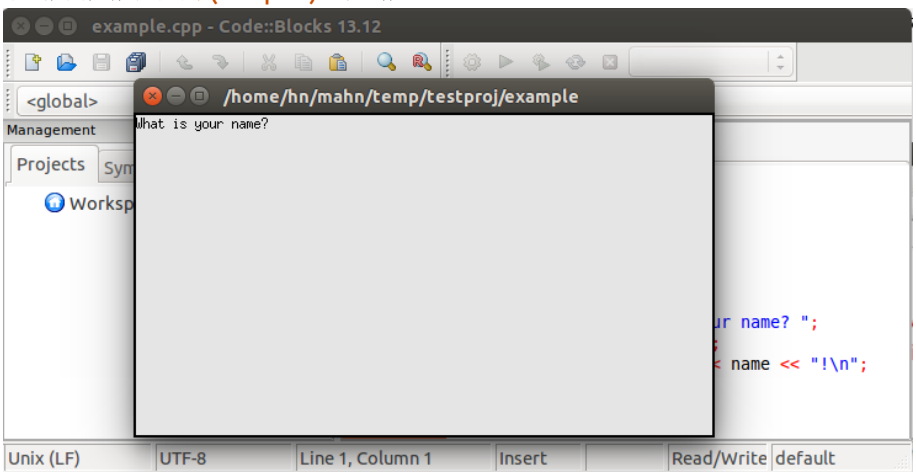


উপরের ছবিতে লক্ষ্য করো আমরা যে ক্রমলেখ লিখেছি সেটাতে আসলে নীচের সংকেত গুলোই লিখেছি। এটি ঠিক হয়মান মন্ত্রপাতি (online software) হিসাবে cpp.sh জালপাতা (webpage) ব্যবহার করে যে ক্রমলেখ লিখেছিলাম সেটিই। কোডব্লকসে নমুনা হিসাবে আগে থেকে এই রকম ক্রমলেখ থাকে না, তোমাকে নিজে এটা লিখে নিতে হবে। তারপর উপরের ছবিতে দে-

খানো বানাও+চালাও (Build+Run) বোতামে টিপ দিয়ে তুমি ক্রমলেখটি চালাবে। বানাও+চালাও বোতামে টিপ দিলে আসলে প্রথমে তোমার লেখা ক্রমলেখ সংকলন (compile) হয়ে নির্বাহযোগ্য (executable) ক্রমলেখ তৈরী হয়, আর তারপর সেই নির্বাহযোগ্য ক্রমলেখ আসলে চলে (run)।

```
// Example program
#include <iostream>
#include <string>

int main()
{
  std::string name;
  std::cout << "What is your name? ";
  getline (std::cin, name);
  std::cout << "Hello, " << name << "!\n";
}
```

তো বানাও+চালাও (Build+Run) বোতামে টিপ দিলে সাধারণত নীচের ছবির মতো করে একটা অতিরিক্ত জানালা (window) আসবে। আর তাতে লেখা থাকবে What is your name ? তখন তুমি যদি তোমার নাম লিখে দাও gonimia আর তারপর ভুক্তি (enter) চাপ দাও, তাহলে পরের সারিতে দেখবে লেখা আসবে Hello, gonimia!। নামটুকু নেওয়ার আগে What is your name? দেখানোকে আমরা বলি যোগান যাচনা (input prompt) আর নাম gonimia দেওয়াটাকে আমরা বলি যোগান (input) দেওয়া আর পরের সারিতে Hello, gonimia! দেখানোকে আমরা বলি ফলন (output) দেওয়া।

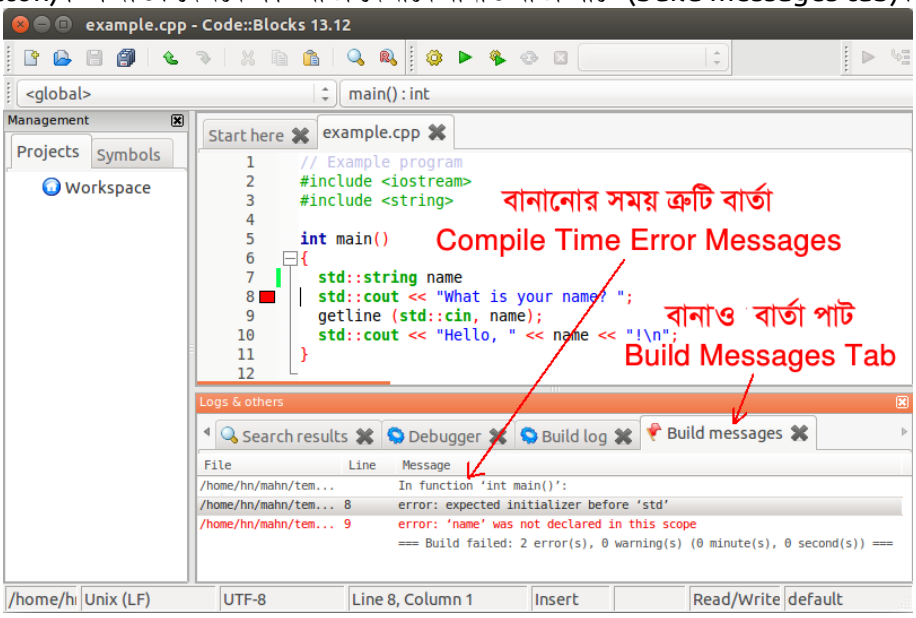


তো তুমি বানাও+চালাও বোতামে (Build+Run button) টিপ দিয়ে দেখো কী হয়। উপরে যে ভাবে বলা হলো সে রকম হয় কী না দেখো। তোমার বোঝার সুবিধার্থে নির্বাহ (execution) শেষে ওই অতিরিক্ত জানালাতে কী থাকবে তা নীচে দেখানো হলো।

```
What is your name? gonimia
Hello, gonimia!
```

এবার আমরা দেখবো কোডব্লকসে লেখা আমাদের ক্রমলেখতে যদি কোন ভুল থাকে তাহলে বানাও+চালাও (build+run) বোতামে টিপ দিলে কী ঘটবে? বানাও+চালাও বোতামে টিপ

দেওয়ার আগে তোমাকে প্রাপণ্য (menu) থেকে Viewএর (দৃষ্টি) অধীনে Logsএ (ঘটচা) টিপ দিতে বলবো, অথবা F2 চাপনি (key) চাপলেও একই কাজ হবে। এর ফলে নীচের ছবিতে দেখানো বানাও বার্তা পাটের (build messages tab) দেখা যাবে, যেখানে আসলে সংকলনে কোন ত্রুটি (error) থাকলে তা দেখানো হবে। এবার আমরা ইচ্ছে করে একটা ত্রুটি (error) তৈরী করে দেই। যেমন ধরো std::string name; লেখা রয়েছে যে সারিতে সেখানে একদম শেষ হতে দির্তি (semicolon) ; তুমি মুছে দাও। আর তারপর বানাও+চালাও বোতামে (build+run button) টিপ দাও। দেখবে ত্রুটি বার্তা দেখাবে বানাও বার্তা পাটে (build messages tab)।



বানাও বার্তা পাটে (build messages tab) যে বার্তাগুলো আসবে তা নীচে দেখানো হলো। দ্বিতীয় সারিতে দেখো 8 মানে বুঝাচ্ছে ৮ম সারিতে ত্রুটি আছে, আর ত্রুটিটা হলো ; থাকতে হবে। আসলে ; দরকার আমাদের ৭ম সারির শেষে। সাধারণত যে সারিতে ত্রুটি আছে বলা হয়, ত্রুটি সেই সারি বা আগের সারিতে থাকে। এখানে ; থাকায় সংকলক (compiler) আসলে ঠিক ৭ম আর ৮ম সারি নিয়ে কিঞ্চিত বিভ্রান্তিতে রয়েছে। ক্রমলেখ রচনার সময় আমরা নানান রকম ভুল ত্রুটি করি, তুমি ক্রমলেখ লেখার চর্চা করতে থাকলে এই ত্রুটিগুলোর সাথে পরিচিত হয়ে যাবে। তখন দেখা মাত্রই বুঝতে পারবে ভুলটুকু কী আর কী করে সেটা ঠিক করতে হবে। যাইহোক ত্রুটিটুকু বুঝতে পারলে আমরা সেটি ঠিক করে আবার বানাও+চালাও বোতামে (build+run button) টিপ দিবো, আর তখন ক্রমলেখ (program) সফল ভাবেই নির্বাহিত হবে।

```
In function 'int main()':
8: error: expected initializer before 'std'
9: error: 'name' was not declared in this scope
```

১.৩ গণনা পরিভাষা (Computing Terminologies)

- পরিগণক (programmer)
- পরিগণনা (programming)
- ভাষা (language)
- স্বাভাবিক (natural)

১.৩. গণনা পরিভাষা (Computing Terminologies)

- ক্রমলেখ (program)
- ক্রমলেখক (programmer)
- গণনি (computer)
- নির্বাহ (execution)
- নির্বাহযোগ্য (executable)
- সম্পাদনা (edit)
- সংকলন (compile)
- হয়মান (online)
- নয়মান (offline)
- মন্ত্র, মন্ত্রপাতি (software)
- জালপাতা (webpage)
- আন্তর্জাল (internet)
- ব্রাজক (browser)
- জাল ঠিকানা (web address)
- সম্পাদন খানা (editing area)
- পাট (tab)
- উপাত্ত (option)
- উপাত্ত পাট (options tab)
- সংকলন পাট (compilaiton tab)
- নির্বাহ পাট (execution tab)
- চালাও (run)
- বানাও (build)
- নমুনা (sample)
- টিপ (click)
- বোতাম (button)
- বাতিল (cancel)
- যোগান (input)
- ফলন (output)
- থামন (stop)
- ভুক্তি (enter)
- যাচনা (prompt)
- ত্রুটি (error)
- স্তম্ভ (column)
- প্রমিত (standard)
- অনুকুলায়ন (optimisation)
- মিথস্ক্রিয়া (interaction)
- মিথস্ক্রিয় (interactive)
- চাপনি (keyboard)
- কিছুনা (none)
- পাঠনিক (text)
- পরিচালনা তন্ত্র (operating system)
- সংস্থাপন (install)
- আপদনাশক (debugger)
- ছবিও (video)
- ব্যবহার পুস্তিকা (user manual)
- জানালা (window)
- প্রাপণ্য (menu)
- নথি (file)
- ফাঁকা নথি (empty file)
- নতুন (New)
- সংরক্ষণ (save),
- মূর্তি (icon)
- সংকেত (code)
- দির্তি (semicolon)
- দৃষ্টি (view)
- ঘটচা (log)

অধ্যায় ২

ক্রমলেখয়ের কাঠামো (Program Structure)

গণনিতে (computer) নির্বাহযোগ্য (executable) একগুচ্ছ নির্দেশের (instruction) ক্রমকে ক্রমলেখ (program) বলা হয়। আমরা সিপিপি (c++) ভাষায় ক্রমলেখ তৈরী করবো। ক্রমলেখ সাধারণত একটি সম্পাদনা (editor) মন্ত্র (software) ব্যবহার করে তৈরী করা হয়। আমরা একাজে আপাতত cpp.sh নামের একটি জালপাতা (webpage) ব্যবহার করবো। সিপিপি ভাষায় তৈরী ক্রমলেখকে প্রথমে একটি সংকলক (compiler) দিয়ে সংকলন (compile) করে গণনিতে নির্বাহযোগ্য সংকেত (code) তৈরী করা হয়। তারপর সেই সংকেত চালালে (run) বা নির্বাহ (execution) করলে আমরা সাধারণত যন্ত্রালয়ের (console) নজরিতে (monitor) ফলন (output) দেখতে পাই । ক্রমলেখ অনেক সময় আমাদের কাছ থেকে যন্ত্রালয়ের চাপনির (keyboard) বা টিপনির (mouse) মাধ্যমে যোগান (input) নিতে পারে। জেনে রেখো যন্ত্রালয় (console) বলতে যোগানের (input) জন্য চাপনি ও টিপনি (keyboard and mouse) আর ফলনের (output) জন্য নজরি (monitor) বুঝানো হয়। ক্রমলেখ লিখতে গেলে যন্ত্রালয় (console) থেকে যোগান (input) নেয়ার ও যন্ত্রালয়ে (console) ফলন (output) দেখানোর কথা তুমি প্রায়শই শুনতে পাবে। কাজেই এগুলো কী বুঝায় সেটা ভালো করে মনে রেখো।

২.১ শুভেচ্ছা বার্তার ক্রমলেখ (Wishing Program)

সিপিপি (c++) ভাষায় এমন একটি ক্রমলেখ (program) রচনা করো যেটি চালালে (run) তোমার ক্রমলেখ ব্যবহারকারীকে শুভেচ্ছা জানাবে। আসলে এটিই হবে সিপিপি ভাষায় তোমার লেখা প্রথম ক্রমলেখ। প্রত্যেক পরিগণনা ভাষায়ই এমন একটা করে ক্রমলেখ রচনা করা হয়।

নীচে শুভেচ্ছা বার্তা দেখানোর জন্য একটি ক্রমলেখ রচনা করা হয়েছে। আর ক্রমলেখটি সংকলন (compile) করে নির্বাহ (execution) করলে বা চালালে (run) যে ফলন (output) পাওয়া যাবে তাও দেখানো হয়েছে। ওই ক্রমলেখতে মুল যে বিবৃতিটি (statement) আমাদের shuvessa nin দেখাবে সেটি হল cout << "shuvessa nin" << endl; এখানে cout হল console out মানে যন্ত্রালয়ের ফলন যন্ত্র (output device)। আর endl হল end line অর্থাৎ যেখানে endl বলা আছে সেখানে ফলনে ওই সারি শেষ হবে। খেয়াল করো আমরা নজরিতে যা দেখাতে চাই তা হুবহু উদ্ধৃতি "" চিহ্নের ভিতরে লেখা হয়েছে। আর << দিয়ে আমরা "shuvessa nin" ও endl কথাগুলোকে cout এর কাছে পাঠাই দেখানোর জন্য।

স্মরণ রেখো cout এর বিবৃতিটি (statement) ছাড়া আমাদের ক্রমলেখতে আরো অন্যান্য বিবৃতি যেগুলি আছে সেগুলি আমাদের লেখা প্রায় সকল ক্রমলেখতেই থাকবে। আমরা তাই আপাতত ওগুলো একরকম জোর করে মনে রাখার চেষ্টা করবো। তারপরেও অবশ্য আমরা নীচের আলোচনা থেকে সংক্ষেপে জেনে নেব বাঁকী বিবৃতিগুলোর কোনটার কাজ মোটামুটি কী।

ফিরিস্তি ২.১: শুভেচ্ছা জানানোর ক্রমলেখ (Wishing Program)

```
#include <iostream>
using namespace std;
int main()
{
  cout << "shuvessa nin" << endl;

  return 0;
}
```

ফলন (output)

```
shuvessa nin
```

একদম শুরুতে আমরা #include <iostream> ব্যবহার করেছি কারণ iostream নামে একটা শির নথি (header file) আছে যেটা আমরা আমাদের ক্রমলেখতে অন্তর্ভুক্ত করতে চাই। ওই শির নথিতে নানান বিপাতক (function) আছে যেগুলো আমরা পরে জানব ও ব্যবহার করবো। আপাতত জেনে নেই, ওই নথিতে cout আর endl আছে। মুলত আমাদের ক্রমলেখতে cout আর endl ব্যবহার করার জন্যই আমরা iostream অন্তর্ভুক্ত করেছি। এরকম আরো শির নথির (header file) কথা আমরা পরে বিস্তারিত জানবো ও অবশ্যই ব্যবহার করবো।

using namespace std; আমরা ব্যবহার করেছি কারণ cout আর endl আসলে দুটো নাম, আর ওই নাম দুটো সিপিপিতে আগে থেকে বিদ্যমান std (standard বা প্রমিত) নামাধারের (namespace) অন্তর্গত। সিপিপিতে একই নাম ভিন্ন ভিন্ন নামাধারে অন্তর্গত হতে পারে। তো কোনো নাম বললে সেটি কোন নামাধার থেকে আসবে সেটি আমরা আগেই বলে দিচ্ছি, যেমন আমাদের সকল নাম আসলে std নামাধার থেকে আসবে। নামাধার কী তা আর একটু পরিস্কার করে বুঝতে হলে নীচের পরিচ্ছেদের (para) ঢাকার বনাম বগুড়ার গাবতলি নিয়ে আলোচনা পড়ো।

গাবতলি নামে ঢাকায় একটি জায়গা আছে আবার গাবতলি নামে বগুড়ায় আরেকটি জায়গা আছে। তো গাবতলি বলতে গেলে আমাদের বলতে হবে 'বগুড়ার গাবতলি' অথবা 'ঢাকার গাবতলি', কেবল গাবতলি বললে তো বুঝা যাবে না কোথাকার গাবতলি। বিকল্প হিসাবে আমরা আগেই বলে নিতে পারি যে আমরা এখন ঢাকার কথা আলোচনা করছি। তখন কেবল গাবতলি বললেই আমরা বুঝব এটি ঢাকার গাবতলি। আবার যদি আগেই বলে নেই যে এখন থেকে আমরা বগুড়ার কথা আলোচনা করবো তাহলে গাবতলি বললেই আমরা বগুড়ার গাবতলি বুঝব, ঢাকারটা নয়।

উপরের ক্রমলেখতে using namespace std; বলে আমরা আগেই বলে নিয়েছি যে এরপর থেকে আমরা std নামাধার (namespace) নিয়ে কাজ করবো। কাজেই পরে যখন cout আর endl ব্যবহার করেছি, তখন আর std এর কথা বলতে হয় নি। কিন্তু কেউ যদি তার ক্রমলেখতে using namespace std; না লেখে, তাহলে তাকে cout << "shuvessa nin" << endl ; এর বদলে লিখতে হবে std::cout << "shuvessa nin" << std::endl; অর্থাৎ cout আর endl দুটোর পূর্বেই std:: লাগিয়ে নিতে হবে, ঠিক যেমন গাবতলি বলার আগে ঢাকা লাগিয়ে

বলতে হবে ঢাকার গাবতলি। cout আর endl এর আগে std:: না লিখলে ক্রমলেখ সফল ভাবে সংকলন (compile) করা যাবে না, নানান ত্রুটি (error) বার্তা (message) দেখাবে। সংকলন সময়ে দেখানো ত্রুটিগুলোকে সংকলন কালীন (compile-time) ত্রুটি বলা হয়।

যে কোন সিপিপি ক্রমলেখতে একটি মুল বিপাতক (function) থাকে main যার নাম। এই main বিপাতকের কোন পরামিতি (parameter) থাকবে না, কাজেই main( ) এর পরে গোল বন্ধনী দুটোর মধ্যে কিছু বলা হয় নি। আর প্রতিটি বিপাতক চাইলে একটি মান ফেরত দেয়, main বিপাতক সাধারণত একটি পূর্ণক (integer) ফেরত দেয়, যা main লেখার আগে int হিসাবে উল্লেখ করা হয়েছে। বিপাতক নিয়ে বিস্তারিত আলোচনা আমরা পরে করবো। আপাতত সংক্ষেপে এইটুকুই জেনে রাখি। তো আমাদের ক্রমলেখতে return 0; বিবৃতিটি আসলে বলছে যে আমাদের main বিপাতকটি শুন্য ফেরত পাঠাবে। কার কাছে ফেরত পাঠাবে? যে আমাদের ক্রমলেখ চালাচ্ছে তার কাছে। main বিপাতক 0 পাঠানো মানে হলো, এটি সফল ভাবে শেষ হয়েছে, কোন ত্রুটি বিচ্যুতি ঘটে নি। 0 ছাড়া অন্যকিছু ফেরত পাঠানো নিয়েও আমরা পরে আলোচনা করবো।

সিপিপিতে দুটো বাঁকা বন্ধনির {} ভিতরে যা থাকে তাকে বলা হয় একটি মহল্লা (block)। প্রতিটি বিপাতকের একটি শরীর (body) থাকে যেটি মহল্লার ভিতরে থাকে। লক্ষ্য করে দেখো আমাদের main বিপাতকের cout আর return দিয়ে শুরু হওয়া বিবৃতি দুটি একটি মহল্লার ভিতরে রয়েছে। আর একটি বিষয় খেয়াল করো, আমাদের বিবৃতিগুলোর শেষে কিন্তু একটি করে দির্তি (semicolon) ; রয়েছে। সিপিপিতে বেশীরভাগ বিবৃতির পরেই আমরা এইরকম দির্তি ; দিয়ে বিবৃতি শেষ করি। ঠিক বাংলা ভাষায় প্রতিটি বাক্যের পরে দাঁড়ি । দেয়ার মতো ব্যাপার।

সব মিলিয়ে এই হল আমাদের প্রথম ক্রমলেখ, যেটা ব্যবহারকারীকে শুভেচ্ছা জানাবে।

২.২ নাম-ধাম-বৃত্তান্তের ক্রমলেখ (Detailing Program)

সিপিপিতে এমন একটি ক্রমলেখ (program) রচনা করো যেটি চালালে ব্যবহারকারীকে তোমার নাম-ধাম-বৃত্তান্ত কয়েক সারিতে মালা (string) আকারে বলে দেয়। সাথে সংখ্যা (number) হিসাবে তোমার বয়স ও তোমার ফলাফলের জিপিএও বলে দেয়।

ফিরিস্তি ২.২: নাম-ধাম-বৃত্তান্তের ক্রমলেখ (Detailing Program)

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  cout << "amar nam goni mia" << endl;
  cout << "amar bari bogra" << endl;
  cout << "ami thaki dhakai" << endl;
  cout << "amar boyos " << 20 << " bosor" << endl;
  cout << "amar folafol gpa " << 3.99 << endl;

  return EXIT_SUCCESS;
}
```

ফলন (output)

```
amar nam goni mia
amar bari bogra
ami thaki dhakai
amar boyos 20 bosor
amar result gpa 3.99
```

উপরের ক্রমলেখতে আমরা নাম-ধাম-বৃত্তান্ত কয়েক সারিতে দেখিয়েছি। এই ক্রমলেখয়ের প্রতিটি cout দিয়ে শুরু বিবৃতির সাথে পরে দেখানো ফলন মিলিয়ে নাও। লক্ষ্য করো cout দিয়ে " " উদ্ধৃতি অংশগুলোর ভিতরে আমরা যে মালাগুলো (string) দেখাতে বলেছি সেগুলোই ফলনে হুবহু সেভাবেই দেখানো হয়েছে। আর প্রতিবার endl অর্থাৎ end line পেলে ফলন পরের সারিতে চলে গেছে। শেষের দুটো cout বিবৃতিগুলো খেয়াল করো। এইদুটোতে বয়স ও জিপিএ আমরা সংখ্যা হিসাবে দেখিয়েছি। তুমি চাইলে কিন্তু সংখ্যা হিসাবে না দেখিয়ে মালার ভিতরেই দেখাতে পারতে যেমন নীচের মতো, সেক্ষেত্রে ফলন কিন্তু দেখতে একই রকম হতো।

```
cout << "amar boyos 20 bosor" << endl;
cout << "amar folafol gpa 3.99" << endl;
```

সবশেষে একটা বিষয় খেয়াল করো। আমরা এই ক্রমলেখতে return 0; এর বদলে লিখেছি return EXIT_SUCCESS; আর এই EXIT_SUCCESS আছে cstdlib শির নথিতে (header file)। আমরা তাই #include <cstdlib> লিখে cstdlib শির নথিটিও আমাদের ক্রমলেখতে অন্তর্ভুক্ত করেছি। মনে রাখবে EXIT_SUCCESS এর মান আসলে 0 কিন্তু 0 তো একটা সংখ্যা যেটা দেখে সরাসরি ঠিক অনুধাবন করা যায় না আমরা কী বুঝাতে চাইছি, মানে ক্রমলেখ সফল না বিফল হয়েছে। আমরা তাই স্পষ্ট করে EXIT_SUCCESS লিখবো যাতে চোখে দেখেই আমরা বুঝতে পারি ব্যাপারটা কী। বলে রাখি গণনির (computer) জন্য কিন্তু 0 আর EXIT_SUCCESS একই ব্যাপার কারণ EXIT_SUCCESS এর মান যে 0 ওইটা তো cstdlib নথিতে বলা আছে, সংকলন করার পরে EXIT_SUCCESS আসলে 0 হয়ে যাবে, গণনি ওইটা শুন্যই দেখতে পাবে। আমরা 0 এর বদলে EXIT_SUCCESS আসলে লিখছি কেবল মানুষের বুঝার সুবিধার জন্য, ক্রমলেখ পড়ে চোখে দেখেই যাতে সহজে বুঝা যায় ক্রমলেখটি সফল না বিফল ভাবে শেষ হচ্ছে, সেটাই আমাদের উদ্দেশ্য। তাহলে এখন থেকে ক্রমলেখয়ের main বিপাতকে return 0; না লিখে return EXIT_SUCCESS; লিখবে আর cstdlib শির নথিও অন্তর্ভুক্ত করে নেবে!

তো তোমরা এখন থেকে কয়েক সারিতে কিছু দেখানোর ক্রমলেখ রচনা করতে চাইলে এই ক্রমলেখয়ের মতো করে রচনা করবে। দরকার মতো সংখ্যা (number) ও মালা (string) মিশ্রণ করেও কিন্তু যা দেখাতে চাও তা দেখাতে পারবে। চেষ্টা করে দেখো কেমন?

২.৩ ক্রমলেখতে টীকা লিখন (Writing Program Comments)

এমন একটা ক্রমলেখ (program) রচনা করো যেটি বর্তমান সাল ২০১৫ থেকে তোমার বয়স ২০বছর বিয়োগ করে তোমার জন্ম বছর দেখায়। এই ক্রমলেখতে দরকার অনুযায়ী পর্যাপ্ত টীকা (comment) লিখো, যাতে অনেক দিন পরে তুমি যখন ক্রমলেখটি প্রায় ভুলে যাওয়ার মতো অবস্থায় যাবে তখন ক্রমলেখটি আবার দেখতে গিয়ে দ্রুত চোখ বুলিয়েই সহজে বুঝতে পারো যে এটি তোমার কীসের ক্রমলেখ ছিল। ক্রমলেখতে টীকা থাকলে তুমি ছাড়া অন্য কেউও তোমার লেখা ক্রমলেখ পড়ে সহজে বুঝতে পারবে। টীকা লেখা হয় মানুষ যে ভাষায় কথা বলে সেই ভাষায় যেমন

বাংলায় বা ইংরেজীতে, সিপিপি ভাষায়ও নয়, যন্ত্রের ভাষায়ও নয়, কাজেই টীকা লিখলে অনেক দিন পরেও আমাদের ক্রমলেখ বুঝতে সুবিধা হয়।

ফিরিস্তি ২.৩: ক্রমলেখতে টীকা লেখন (Commenting in Programs)

```
// list of header files needed for this program.

#include <iostream>
#include <cstdlib>

using namespace std;  // use the std namespace

int main()
{
  // Subtract 20years from 2015 to get birthyear

  cout << "amar jonmoshal " << 2015 - 20 << endl;

  return EXIT_SUCCESS;  /* return with success */
}
```

ফলন (output)

```
amar jonmoshal 1995
```

উপরের ক্রমলেখ খেয়াল করো। কঠিন কিছু নয়। আগের মতোই iostream আর cstdlib অন্তর্ভুক্ত (include) করা আছে। তারপর বলা হয়েছে using namespace std; তারপর মুল বিপাতক (function) হিসাবে int main() যেটির কোন পরামিতি (parameter) নাই কারণ () গোল বন্ধনীর ভিতরে কিছু নাই আর যেটি একটি পূর্ণক (integer) ফেরত দেয় কারণ int বলা আছে শুরুতে। তারপর মুল বিপাতকের শরীরে দুটো {} বাঁকাবন্ধনীর ভিতরের মহল্লায় (block) বলা আছে cout << "amar jonomoshal " << 2015 - 20 << endl; অর্থাৎ ফলনে amar jonomoshal দেখিয়ে তারপর 2015 থেকে 20 বিয়োগ করলে যে 1995 পাওয়া যায় তা দেখাবে। তারপর মহল্লার ভিতরে শেষ বিবৃতি (statement) আছে return EXIT_SUCCESS; যা আগের মতোই বলছে যে আমাদের ক্রমলেখ ওইখানে সফল ভাবে শেষে হয়ে বের হয়ে যাবে। EXIT_SUCCESS নিয়ে আমরা আগের পাঠে বিস্তারিত আলোচনা করেছি, ওই পাঠ থেকেই দেখে নিতে পারো, কাজেই সেটা আবার এখানে আলোচনা করছি না।

যাইহোক, খেয়াল করে দেখো ওপরে বর্ণিত বিষয়গুলো ছাড়াও উপরের ক্রমলেখতে আরো কিছু বাক্য ও সারি দেখা যাচ্ছে যেমন প্রথম সারিটিই হল // list of header files needed for this ।program এই বাক্যটি আসলে আমাদের ক্রমলেখয়ের অংশ নয়, অর্থ্যাৎ ক্রমলেখ যখন চালানো (run) হবে তখন এই বাক্যের কোন প্রভাব থাকবে না। ক্রমলেখ এমন ভাবে চলতে থাকবে যাতে মনে হবে ওই বাক্যটি যেন ওখানে নাই। এরকমের বাক্যগুলোকে বলা হয় টীকা (comment)। খেয়াল করো টীকার বাক্যটির একদম সামনে রয়েছে // অর্থাৎ সামনের দিকে হেলানো দুটো দাগ। ওই দুটো দাগ হতে শুরু করে ওই সারিতে তারপরে যাই থাকবে সব মিলিয়ে হবে একটি টীকা। এইরকম টীকা যেহেতু কেবল এক সারিতে সীমাবদ্ধ তাই একে বলা হয় সারি টীকা (line comment)। সিপিপি ভাষায় অধিকাংশ সময়ই সারি টীকা ব্যবহার করা হয়।

সারি টীকা যদি সারির একদম শুরুতে লেখা হয় তাহলে সাধারণত এটি টীকার ঠিক নীচে যে সংকেত (code) থাকে তার জন্য লেখা হয়। যেমন // list of header files needed for this program এই টীকাটি একদম সারির শুরু থেকে লেখা হয়েছে, এটি তাই পরের দুই সারিতে #include <iostream> আর #include <cstdlib> কেন লেখা হয়েছে সেটি ব্যাখ্যা করছে। সারি টীকা অনেক সময় সারির শেষ দিকেও লেখা হয়। যেমন // we will use the std namespace টীকাটি লেখা হয়েছে using namespace std; দিয়ে শুরু হওয়া সারির শেষে। সারির শেষ দিকে লেখা এইরকম সারি টীকা সাধারণত সারির প্রথমে যে সংকেত (code) লেখা হয়েছে তা ব্যাখ্যা করতে ব্যবহার করা হয়। অনেক সময় টীকা লিখা হয় শুরুতে /* আর শেষে */ চিহ্ন দিয়ে, যেমন return EXIT_SUCCESS; এর সারিতে শেষে লেখা হয়েছে। এইরকম টীকা একাধিক সারি মিলিয়ে হতে পারে, তাই এদেরকে সারি টীকা না বলে মহল্লা টীকা (block comment) বলা হয়। সিপিপিতে আমরা অধিকাংশ সময় আসলে সারি টীকাই ব্যবহার করি।

তুমি যখন তোমার ক্রমলেখতে টীকা লিখবে তখন হয়তো ইংরেজীতেই টীকা লিখবে। অথবা ইংরেজী অক্ষরে বাংলায়ও টীকা লিখতে পারো। আজকাল অনেক সংকলক (comiler) ও সম্পাদক (editor) ইউনিকোড (unicode) সংকেত বুঝতে পারে। কাজেই টীকা বাংলায়ও লেখা সম্ভব। আমরা এরপর থেকে সিপিপিতে লেখা সকল ক্রমলেখতে টীকা বাংলায় লিখবো, যাতে আমরা আমাদের নিজের ভাষায় সহজে বুঝতে পারি। এগুলো যেহেতু নির্বাহ execution) হবে না, কাজেই খামোকা কেন কষ্ট করে ইংরেজীতে লিখতে যাবো! আর বিদেশী কেউ তো আমাদের ক্রমলেখয়ের সংকেত দেখবে না, কাজেই আমরা আমাদের বাংলা ভাষাতেই টীকা লিখবো। তবে মনে রাখবে বিদেশী কারো পড়ার সম্ভাবনা থাকলে আমাদের টীকা সহ সবকিছু ইংরেজী ভাষাতেই লিখতে হবে। তাহলে সারি টীকা আর মহল্লা টীকা শেখা হলো। এখন থেকে ক্রমলেখ লেখার সময় যথেষ্ট পরিমানে টীকা দিবে কেমন? আমিও ক্রমলেখগুলোতে টীকা দেবো, যাতে তোমাদের বুঝতে সুবিধা হয়।

২.৪ ক্রমলেখতে ফাঁকা দেওয়া (Spacing and Indentation)

সিপিপি ক্রমলেখ (program) লিখতে কখন নতুন সারি শুরু করবে? কখন ফাঁকা ফাঁকা করে লিখবে? কখন সারিতে একটু ছাড়ন দিয়ে লিখবে। একটি ক্রমলেখ লিখে এই বিষয়গুলো আলোচনা করো। চলো আমরা আমাদের শুভেচ্ছা জানানোর ছোট ক্রমলেখটি দিয়েই আলোচনা করি।

ফিরিস্তি ২.৪: ক্রমলেখতে ফাঁকা দেওয়া (Spacing in Programs)

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  cout << "shuvessa nin" << endl;
  return EXIT_SUCCESS;
}
```

উপরের ক্রমলেখতে আমরা আপাতত টীকা (comment) লিখি নাই। এই ক্রমলেখতে # বা কাটাকাটি (octothorpe) চিহ্ন দিয়ে শুরু হওয়া সারিগুলো তোমাকে আলাদা আলাদা সারিতে লিখতে হবে। আর এটি সারির শুরু থেকে হলেই ভালো। তুমি অবশ্য সারির শেষের দিকে চাইলে

সারি টীকা (line comment) লিখতে পারো যাতে বুঝা যায় ওই সারির শুরুর দিকে তুমি আসলে কী করতে চেয়েছো। তোমার ক্রমলেখতে লেখা কাটাকাটি চিহ্ন # দিয়ে শুরু হওয়া সারিগুলো সাধারণত সংকলক (compiler) দিয়ে প্রক্রিয়া করা হয় না। আলাদা একটি মন্ত্র (software) যার নাম পুর্ব-প্রক্রিয়ক (preprocessor) সেটা দিয়ে সংকলন করারও আগে এইগুলো প্রক্রিয়া করা হয়, কাজটা বেশীর ভাগ সময়ে অবশ্য সংকলকই করিয়ে নেয়। পূর্ব-প্রক্রিয়ক (preprocessor) বিষয়ে বিস্তারিত আলোচনা আমরা পরে করবো।

```
#include <iostream> //  যোগান ফলন স্রোত শির নথি অন্তর্ভুক্ত হলো
```

কোন বিবৃতি পুর্বপ্রক্রিয়ক (preprocessor) না সংকলক (compiler) দিয়ে প্রক্রিয়াকরণ হবে এটা বুঝার আরেকটা সহজ উপায় আছে। এইরকম দির্তি ; (semicolon) আর মহল্লার (block) জন্য যে বাঁকা বন্ধনী } ব্যবহৃত হয় তা দিয়ে শেষ হওয়া বিবৃতিগুলো সাধারণত সংকলক দিয়ে প্রক্রিয়াকরন করা হবে, পূর্ব-প্রক্রিয়ক দিয়ে নয়। যাইহোক, সংকলক দিয়ে যে সংকেতগুলো (code) প্রক্রিয়া করা হয় সেগুলো যে ভিন্ন ভিন্ন সারিতেই লিখতে হবে, বা অনেক ফাঁকা ফাঁকা (space) করেই লিখতে হবে এ রকম কোন কথা নেই। তুমি চাইলে তোমার পুরো ক্রমলেখতে থাকা সকল সংকলনযোগ্য সংকেত এক সারিতে লিখতে পারো। যেমন উপরের ক্রমলেখয়ের সংকলনযোগ্য অংশটুকু আমরা চাইলে নীচের মতো করে টানা এক সারিতে লিখতে পারি।

```
#include <iostream>
#include <cstdlib>
using namespace std; int main() { cout << "shuvessa
 nin" << endl; return EXIT_SUCCESS; }
```

উপরে যদিও দুই সারিতে দেখা যাচ্ছে আমরা আসলে using থেকে শুরু করে } পর্যন্ত টানা একসাথে লিখেছি, কিন্তু এখানে পাশের দিকে স্থানের স্বল্পতার কারণে টানা সারিটি ভেঙে দুই সারি হয়ে গেছে। তোমার সম্পাদকে (editor) এ যদি পাশের দিকে অনেক জায়গা থাকে তুমি এক সারিতেই লিখতে পারবে। আসলে নুন্যতম একটি ফাঁকা (space) দেয়া বাধ্যতামুলক হয়ে যায় যখন পরপর দুটো শব্দ লেখা হয়। যেমন using, namespace, std, int, main এইরকম শব্দ পরপর দুটো থাকলে তোমাকে কমপক্ষে একটি ফাঁকা (space) দিতে হবে। দুটো চিহ্ন যেমন বন্ধনী () বা দির্তি ; বা আরো অনেক প্রতীক আছে, এইগুলো পরপর দুটো থাকলেও কোন সমস্যা নাই। অর্থ্যাৎ একাধিক প্রতীক কোন ফাঁকা না দিয়েও তুমি একসাথে লিখতে পারবে।

এখন প্রশ্ন করতে পারো ফাঁকা দেয়া যদি ব্যাপার না হয়, তাহলে ক্রমলেখ লিখতে কেন ফাঁকা দেবো। বেশী বেশী ফাঁকা আসলে গণনির (computer) জন্য দরকার নেই কিন্তু দরকার মানুষের জন্য। আগের পাঠের কথা মনে করো। আমরা কেন টীকা (comment) লিখেছিলাম? টীকা তো আর নির্বাহিত হয় না। আমরা যাতে অনেকদিন পরে ক্রমলেখয়ের সংকেত (code) দেখে সহজে বুঝতে পারি, আমরা তাই টীকা লিখেছিলাম। তো ক্রমলেখ যদি পুরোটা একটা লম্বা সারি হয়, আমাদের মানুষের পক্ষে সেটা দেখে বুঝে ওঠা খুবই কষ্টকর হবে। মুলত আমাদের মানুষের বুঝার সুবিধার্থে আমরা ক্রমলেখ সারিতে সারিতে ভেঙ্গে ভেঙ্গে লিখি বা দরকার মতো একসাথে লিখি।

ক্রমলেখতে ফাঁকা দেয়ার ব্যাপারটি বাংলায় বা ইংরেজীতে রচনা লেখার মতোই, কখন তুমি আলাদা বাক্য করবে, কখন তুমি আলাদা পরিচ্ছেদ (para) করবে, কখন তুমি আলাদা অনুচ্ছেদ (section) করবে, এই রকম। কোন বিষয়ের সাথে বেশী সম্পর্কিত বিবৃতিগুলো আমরা সাধারণত পরপর সারিতে কোন ফাঁকা (blank line) না দিয়ে লিখবো। আর দুটো বিষয়ের সারিগুলোর মাঝে হয়তো এক সারি ফাঁকা দিয়ে লিখবো, আর বিষয়গুলোর মধ্যে খুব বেশী যোগাযোগ না থাকলে হয়তো আমরা দুই বা আরো বেশী সারি ফাঁকা দিয়ে লিখবো। তাহলে এখন থেকে ক্রমলেখ লেখার সময় দরকার মতো ফাঁকা দিয়ে দিয়ে লিখবে যাতে তোমার ক্রমলেখ পড়া সহজ হয়।

সবচেয়ে উপরে যেভাবে আমরা ক্রমলেখ লিখেছি সেখানে আরো একটা ব্যাপার খেয়াল করো, আমরা cout বা return এর বিবৃতিগুলো লেখার আগে তাদের নিজ নিজ সারিতে বেশ কিছুটা ফাঁকা দিয়ে লিখেছি, একদম সারির শুরু থেকে লিখি নাই। এটি কেন করলাম? এটি করলাম এ কারণে যে ওই দুটো সারি আসলে আমাদের মহল্লার ভিতরে আছে। লক্ষ্য করো মহল্লার বাঁকা বন্ধনী দুটো কেমন দেখেই বুঝা যায় যে এরা দুজনে দুজনার। আর মহল্লার ভিতরের বিবৃতিদুটো কেমন একটু ভিতরের দিকে থাকায় পরিস্কার বুঝা যায় যে ওরা আসলেই ওই মহল্লার ভিতরে। তো দরকার মতো কোন বিবৃতি এরকম সারির একটু ভিতরের দিকে থেকে লেখার ব্যাপারটিকে বলা হয় ছাড়ন দেয়া (indentation)। ক্রমলেখ লেখার সময় এখন থেকে তোমরা অবশ্যই দরকার মতো ছাড়ন দিয়ে লিখবে, তাহলে দেখবে ক্রমলেখ পড়া ও বোঝা কত সহজ হয়ে যায়।

এই পর্যায়ে জিজ্ঞেস করতে পারো, প্রত্যেক সারিতে এভাবে অতগুলা করে ফাঁকা চাপবো কেমনে এইটা তো বিরক্তিকর। আসলে তোমার চাপনিমাঁচায় (keyboard) একটা লম্ফ (tab) চাপনি আছে, দেখো ওইটা চাপলে একসাথে ৪টা বা ৮টা ফাঁকা (space) এর সমপরিমান ফাঁকা একবারে আসে। তো দরকার মতো একবার বা দুবার লম্ফ চাপলেই হয়ে গেলো। কাজেই ক্রমলেখ লেখার সময় কখনোই এই আলসেমি টুকু করবে না। ছাড়ন দেয়া ক্রমলেখ লেখার জন্য গুরুত্বপূর্ণ ব্যাপার, সুন্দর দেখা যাওয়া আর তাড়াতাড়ি পড়ার জন্য দরকারী, ক্রমলেখতে কোন ভুল থাকলে আমরা যখন ভুল বের করতে চাই তখনও খুব খুব দরকারী, বড় বড় ক্রমলেখ যখন লিখবে তখন ব্যাপারটা খানিকটা ঠেকে ঠেকে শিখে অভিজ্ঞতা দিয়ে ভালো করে বুঝতে পারবে।

২.৫ অনুশীলনী সমস্যা (Exercise Problems)

ধারণাগত প্রশ্ন: নীচে কিছু ধারণাগত প্রশ্ন রয়েছে। প্রশ্নগুলোর উত্তর নিজে নিজে বের করবে।

১. ক্রমলেখ (program) বলতে কী বুঝো? ক্রমলেখ কি কেবল গণনিতেই নির্বাহযোগ্য?

২. সিপিপি ভাষায় ক্রমলেখ তৈরী হতে সেটার ফলাফল দেখা পর্যন্ত কী কী ঘটনা ঘটে?

৩. যন্ত্রালয় (console) কী? এর যোগান (input) ও ফলন (output) যন্ত্রগুলো কী কী?

৪. সিপিপিতে শিরনথি (header file) বলতে কী বুঝো? আমাদের ক্রমলেখগুলোতে শির নথি iostream ও cstdlib আমরা কেন ব্যবহার করেছি?

৫. নামাধার (namespace) কী? বাস্তব জীবনে ও পরিগণনায় উদাহরণ সহ ব্যাখ্যা করো।

৬. সিপিপিতে main বিপাতক হতে ফেরতের সময় return 0; না লিখে তার বদলে return EXIT_SUCCESS; লিখা কেন উত্তম? ব্যাখ্যা করো।

৭. ক্রমলেখতে ছাড়ন দেয়া (indentation) মানে কী? ছাড়ন দেয়ার পক্ষে-বিপক্ষে যুক্তি লিখ। ক্রমলেখ কেন বেশ ফাঁকা ফাঁকা করে লিখা উচিৎ?

৮. ক্রমলেখতে টীকা (comment) লেখা কী? ক্রমলেখতে টীকা (comment) লিখার কয়েকটি কারণ ব্যাখ্যা করো? সারি (line) টীকা ও মহল্লা (block) টীকা কী?

৯. একটি সিপিপি ক্রমলেখতে (program) নীচের কোন বিপাতকটি অবশ্যই থাকতে হবে?

ক) start() খ) system() গ) main() ঘ) program()

১০. ক্রমলেখ সফল ভাবে শেষ হলে main বিপাতক হতে সাধারণত কত ফেরত পাঠানো হয়?

ক) -1 খ) 0 গ) 1 ঘ) কিছুই না

১১. সিপিপিতে মহল্লা (block) বুঝানোর জন্য নীচের কোনগুলো ব্যবহার করা হয়?

ক) { } খ) < > গ) ( ) ঘ) begin end

১২. সিপিপিতে একটি বিবৃতির (statement) শেষে সাধারণত কোন চিহ্ন ব্যবহার করা হয়?

ক) . খ) ; গ) : ঘ) ,

১৩. সিপিপিতে নীচের কোনটি সঠিক টীকা (comment)?

ক) */ টীকা */ খ) ** টীকা ** গ) /* টীকা */ ঘ) { টীকা }

পরিগণনার সমস্যা: নীচে আমরা কিছু পরিগণনার সমস্যা দেখবো। এই সমস্যাগুলো আগে ধৈর্য্য ধরে নিজে নিজে সমাধান করতে চেষ্টা করবে। যখন একেবারেই পারছো না বলে মনে হয় তখনই কেবল সমাধান দেখে নিতে পারো। সমাধানগুলো পরিগণনার প্রশ্নগুলোর শেষে আছে।

১. নীচের কথাগুলো ফলনে (output) দেখানোর জন্য সিপিপিতে একটি ক্রমলেখ লিখো। দেখতে সুন্দর লাগার জন্য তোমার ক্রমলেখতে দরকার মতো ফাঁকা ফাঁকা দিবে আর সহজে বুঝার জন্য দরকার মতো টীকা (comment) লিখবে।

```
tomar boyosh noy bosor.
porigonona shikhte chao?
porigonona ki sohoj na!
```

২. সিপিপিতে একটি ক্রমলেখ রচনা করো যেটি নীচের নকশাটির মতো নকশা তৈরী করে। খেয়াল করে দেখো নকশাটি বাংলা অঙ্ক ৪ এর মতো। তুমি চাইলে আরো নানান নকশা, নানান বর্ণ বা অঙ্ক নিজের মতো করে ভেবে নিয়ে সেইমতো নকশা তৈরী করতে পারো। যাইহোক দেখতে সুন্দর লাগার জন্য তোমার ক্রমলেখতে দরকার মতো ফাঁকা ফাঁকা দিবে আর সহজে বুঝার জন্য দরকার মতো টীকা (comment) লিখবে।

```
*****
*   *
* * *
*   *
*****
```

২.৫. অনুশীলনী সমস্যা (Exercise Problems)

পরিগণনা সমাধান: এবার আমরা পরিগণনার সমস্যাগুলোর সমাধান দেখবো। মনে রাখবে সমাধানগুলো দেয়া হয়েছে তুমি যদি একান্তই নিজে নিজে করতে পারছোনা বলে মনে হয়, তখন যাতে একটু সাহায্য কেবল পেতে পারো তাই। কাজেই পারতপক্ষে সমাধান দেখবে না।

১. নীচের কথাগুলো ফলনে (output) দেখানোর জন্য সিপিপিতে একটি ক্রমলেখ লিখো। দেখতে সুন্দর লাগার জন্য তোমার ক্রমলেখতে দরকার মতো ফাঁকা ফাঁকা দিবে আর সহজে বুঝার জন্য দরকার মতো টীকা (comment) লিখবে।

```
tomar boyosh noy bosor.
porigonona shikhte chao?
porigonona ki sohoj na!
```

ফিরিস্তি ২.৫: অণুপ্রেরণার ক্রমলেখ (Inspiring Programming)

```
#include <iostream> //  cout ব্যবহার করার জন্য
#include <cstdlib>  //  EXIT_SUCCESS এর জন্য

using namespace std;// প্রমিত নামাধার ব্যবহারের জন্য

int main()
{
  //  দরকারী কথাগুলো ফলনে দেখাও
  cout << "tomar boyosh noy bosor." << endl;
  cout << "porigonona shikhte chao?" << endl;
  cout << "porigonona ki sohoj na!" << endl;

  return EXIT_SUCCESS;  //  সফল সমাপ্তি
}
```

২. সিপিপিতে একটি ক্রমলেখ রচনা করো যেটি নীচের নকশার মতো নকশা তৈরী করে। খেয়াল করে দেখো নকশাটি বাংলা অঙ্ক ৪ এর মতো। তুমি চাইলে আরো নানান নকশা, নানান বর্ণ বা অঙ্ক নিজের মতো করে ভেবে নিয়ে সেইমতো নকশা তৈরী করতে পারো। যাইহোক দেখতে সুন্দর লাগার জন্য তোমার ক্রমলেখতে দরকার মতো ফাঁকা ফাঁকা দিবে আর সহজে বুঝার জন্য দরকার মতো টীকা (comment) লিখবে।

```
*****
*   *
* * *
*   *
*****
```

এই ক্রমলেখটি কিন্তু অনেক মজার তাই না। তুমি কি বর্ণমালার প্রতিটা বর্ণ আর ০-৯ দশটা অঙ্কের জন্যেই এরকম নকশা তৈরী করতে পারবে? রাস্তাঘাটে বা বিয়ে বাড়িতে অনেক সময় ছোট ছোট বাতি দিয়ে নানান কিছু লেখা হয়, আসলে এই নকশাগুলোর মতো করে নকশা বানিয়েই সেগুলো করা হয়। গণনিতে (computer) এর নজরির (monitor) পর্দায়ও

অনেক কিছু এভাবে দেখানো হয়। আসলে যে কোন ছবিই এরকম অসংখ্য বিন্দুর সমন্বয়ে তৈরী, কিছু বিন্দু জ্বালানো, কিছু বিন্দু নেভানো। যে বিন্দুগুলো জ্বালানো সেগুলো হলো * আর যেগুলো নেভানো সেগুলো ফাঁকা। তো চলো আমরা ক্রমলেখটি দেখি।

ফিরিস্তি ২.৬: নকশা আঁকার ক্রমলেখ (Program Drawing Designs)

```
#include <iostream> //  cout ব্যবহার করার জন্য
#include <cstdlib>  //  EXIT_SUCCESS এর জন্য

using namespace std;// প্রমিত নামাধার ব্যবহারের জন্য

int main()
{
  // দরকার মতো * ও ফাঁকা দিয়ে নকশা
  cout << "*****" << endl;
  cout << "*   *" << endl;
  cout << "* * *" << endl;
  cout << "*   *" << endl;
  cout << "*****" << endl;

  return EXIT_SUCCESS;  //  সফল সমাপ্তি
}
```

২.৬ গণনা পরিভাষা (Computing Terminologies)

- কাটাকাটি (octothorpe) #
- ক্রমলেখ (program)
- গণনি (computer)
- চাপনি (keyboard)
- চালানো (run)
- ছাড়ন দেয়া (indentation)
- জালপাতা (webpage)
- টিপনি (mouse)
- টীকা (comment)
- ত্রুটি (error)
- দির্তি (semicolon) ;
- নজরি (monitor)
- নামাধার (namespace)
- নির্দেশ (instruction)
- নির্বাহ (execution)
- নির্বাহযোগ্য (executable)
- পরামিতি (parameter)
- পরিচ্ছেদ (para)
- পুর্ব-প্রক্রিয়ক (preprocessor)
- পূর্ণক (integer)
- প্রমিত (standard)
- ফলন (output)

২.৬. গণনা পরিভাষা (Computing Terminologies)

- ফলন যন্ত্র (output device)
- বার্তা (message)
- বিপাতক (function)
- বিবৃতি (statement)
- মন্ত্র (software)
- মহল্লা (block)
- মহল্লা টীকা (block comment)
- মালা (string)
- যন্ত্রালয় (console)
- যোগান (input)
- শরীর (body)
- শির নথি (header file)
- সংকলক (compiler)
- সংকলন (compile)
- সংকলন কালীন (compile-time)
- সংকেত (code)
- সংখ্যা (number)
- সম্পাদনা (editor)
- সারি টীকা (line comment)

অধ্যায় ৩

চলক ও ধ্রুবক
(Variables and Constants)

চলকের (variable) মান (value) বদলানো যায় কিন্তু ধ্রুবকের (constant) মান বদলানো যায় না। ক্রমলেখতে উপাত্ত (data) সরাসরি (directly) না লিখে চলক বা ধ্রুবকের মাধ্যমে ব্যবহার করলে একরকমের পরোক্ষতা (indirection) তৈরী হয়। ফলে উপাত্ত ঠিক কতো সেটা না ভেবে উপাত্তটি কীসের আর তার প্রক্রিয়াকরণ কেমন সেটা ভেবে ক্রমলেখ তৈরী সহজ হয়ে যায়।

৩.১ চলকের ব্যবহার (Using Variables)

একটি আয়তের দৈর্ঘ্য ৫ মিটার, প্রস্থ ৩ মিটার। সিপিপি ভাষায় এইরূপ আয়তের ক্ষেত্রফল ও পরিসীমা বের করার ক্রমলেখ (program) রচনা করো। এই ক্রমলেখতে তোমাকে চলক (variable) ব্যবহার করতে হবে, সরাসরি সূত্র থেকে ফলন (output) দেয়া যাবে না।

আমরা আগে এই সমস্যার জন্য সংক্ষিপ্ত ক্রমলেখটা দেখি যেটাতে চলক ব্যবহার না করে একদম সরাসরি সূত্র ব্যবহার করে ক্ষেত্রফল ফলনে (output) দেখানো হবে। আমরা জানি দৈর্ঘ্য আর প্রস্থের গুণফল হল ক্ষেত্রফল আর দৈর্ঘ্য ও প্রস্থের যোগফলের দ্বিগুণ হলো পরিসীমা।

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  cout << "khetrofol holo " << 5 * 3
       << " borgometer" << endl;
  cout << "porishima holo " << 2*(5+3)
       << " meter" << endl;

  return EXIT_SUCCESS;
}
```

উপরে আমরা যে ক্রমলেখ লিখলাম আমরা কিন্তু ওইটা চাই না। ওইখানে সংখ্যাগুলো সরাসরি সুত্রে বসিয়ে হিসাব করে ফলন (output) দেখানো হয়েছে। আমরা চাই ক্ষেত্রফল আর পরিসীমার সূত্রগুলো চলকের নাম দিয়ে লিখতে আর সূত্র লিখার আগে চলকগুলোর মান দিয়ে দিতে। চলক ব্যবহারের নানান সুবিধা আছে। যেমন একটি সুবিধা হলো সুত্রে চলকের নাম থাকায় সূত্র দেখেই সহজে বুঝা যায় কীসের সূত্র, যেমন নীচের ক্রমলেখ দেখো। আর একটি সুবিধা হলো কেউ যদি বলে ৫ না দৈর্ঘ্য হবে ৬, উপরের ক্রমলেখতে কিন্তু দুইখানে 5 বদলাইয়া 6 করতে হবে। ছোট একটা ক্রমলেখতেই যদি দুইখানে বদলাতে হয়, তাহলে বড় একটি ক্রমলেখয়ের কথা চিন্তা করো, সেটাতে আরো কত জায়গায় যে বদলাতে হবে ইয়ত্তা নাই। আমরা এ কারণে চলক ব্যবহার করবো।

ফিরিস্তি ৩.১: ক্রমলেখতে চলকের ব্যবহার (Variables in Programs)

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  int doirgho;  // এই চলকে থাকবে বর্গের দৈর্ঘ্য।
  int prostho;  // এই চলকে থাকবে বর্গের প্রস্থ।
  int khetrofol;  // এই চলকে থাকবে বর্গের ক্ষেত্রফল।
  int porishima;  // এই চলকে থাকবে বর্গের পরিসীমা।

  doirgho = 5;  // দৈর্ঘ্যের এই মান বলে দেয়া আছে
  prostho = 3;  // প্রস্থের এই মান বলে দেয়া আছে।

  // ক্ষেত্রফল বের করার সূত্র হল দৈর্ঘ্য আর প্রস্থের গুণফল।
   khetrofol = doirgho * prostho;

  // পরিসীমা বের করার সূত্র হল দৈর্ঘ্য ও প্রস্থের যোগফলের দ্বিগুন
  porishima = 2*(doirgho + prostho);

  // এবার ক্ষেত্রফল আর পরিসীমা ফলন দেয়া হবে
  cout<< "khetrofol holo " << khetrofol
      << " borgometer" << endl;
  cout << "porishima holo " << porishima
      << " meter" << endl;

  return EXIT_SUCCESS;
}
```

ফলন (output)

```
khetrofol holo 15 borgometer
porishima holo 16 meter
```

উপরের ক্রমলেখতে খেয়াল করো আমরা দৈর্ঘ্য, প্রস্থ, ক্ষেত্রফল, আর পরিসীমার জন্য চারটা চলক নিয়েছি যাদের নাম হলো doirgho, prostho, khetrofol, porishima। তুমি কিন্তু চাইলে এই নামগুলো ইংরেজী শব্দেও দিতে পারতে যেমন length, width, area, perimeter। তুমি চাইলে আবার শব্দগুলোর প্রথম অক্ষর নিয়ে এক অক্ষরের নামও দিতে পারতে যেমন l, w, a, p। তবে আমরা সবসময় চাই এমন নাম দিতে যাতে নামগুলো দেখলেই বুঝা যায় ওই চলকটা কী কাজে ব্যবহার হবে। এক অক্ষরের নাম দিলে অনেক সময় বুঝা যায় কিন্তু একই অক্ষর দিয়ে যদি একাধিক চলকের নাম শুরু হয়, তাহলে মুশকিল হয়ে যায়। অনেকে আবার খালি x, y, z, অথবা a, b, c এই রকম নাম দেয়। ওই রকম নাম দিলে পরে ক্রমলেখ বুঝতে তোমার নিজের বা অন্য কেউ যে পড়বে তার খুবই সমস্যা হবে। সময় নষ্ট করে বের করতে হবে কোন চলক আসলে কী কাজে ব্যবহার করা হয়েছে। কাজেই সবসময় অর্থবোধক আর যথেষ্ট বড় নাম দিতে চেষ্টা করবে, যাতে নাম দেখেই তার উদ্দেশ্য বুঝা যায়। সিপিপিতে চলকের অর্থবোধক (semantic) ও গঠনসিদ্ধ (syntax) নাম দেয়ার বিষয়ে আমরা পরের কোন পাঠে বিস্তারিত আলোচনা করব।

এখন একটা বিষয় খেয়াল করো আমরা এখানে চলকগুলোর নামের আগে লিখেছি int যেটা আসলে integer এর সংক্ষিপ্ত। integer হল পূর্ণক বা পূর্ণ সংখ্যা। আমরা চলকের নামের আগে এই রকম int লিখে বুঝিয়েছি যে আমাদের এই চলকগুলোর মান হবে পূর্ণক, আমরা কোন ভগ্নাংশ ব্যবহার করবো না। তুমি যদি ভগ্নাংশ ব্যবহার করতে চাও তাহলে তোমাকে int এর বদলে float লিখতে হবে। float হল একরকমের ভগ্নাংশ। আমরা সেই আলোচনা পরে আরো বিস্তারিত করবো। তবে int এর বদলে float লিখলে আমাদের ক্রমলেখতে কিন্তু আর কোথাও কোন কিছু বদলাতে হবে না, ঠিক কাজ করবে। আমরা আপাতত int রেখেই এই পাঠের আলোচনা চালাই।

তো উপরের ক্রমলেখতে আমরা যখন লিখলাম int doirgho; এর মানে হলো doirgho নামের আমাদের একটা চলক আছে আর তার মান হবে পূর্ণক। এইযে int doirgho; লিখে এই বিষয়গুলো বুঝাইলাম এটাকে বলা হয় চলক ঘোষনা (variable declaration)। চলক ঘোষনা করলে তারপর থেকেই চলকটি পরবর্তী যেকোন বিবৃতিতে (statement) ব্যবহার করা যায়, কিন্তু ঘোষনা করার সাথে সাথে ওইখানে চলকের মান কত সেইটা কিন্তু আমরা জানিনা, সাধারনত চলকে তখন একটা উল্টাপাল্টা মান থাকে। এইটা নিয়ে আমরা পরে আরো আলোচনা করবো। এই ক্রমলেখতে আমরা দেখছি এর পরে doirgho = 5; লিখে অর্থাৎ = চিহ্ন ব্যবহার করে আমরা doirgho চলকের মান আরোপ (value assign) করেছি 5। সুতরাং এরপর থেকে doirgho চলকের মান হবে 5। একই ভাবে prostho চলকের মানও আমরা 3 আরোপ করেছি।

এবার খেয়াল করো, চলকের মান আরোপণ শেষ হলে আমরা ক্ষেত্রফল আর পরিসীমার সূত্রগুলো লিখেছি, সেখানে কিন্তু এবার মানগুলো সরাসরি লিখি নাই, তার বদলে চলকগুলো ব্যবহার করেছি। এইখানে হিসাব করার সময় চলকের যে মান থাকবে সেইটাই আসলে ব্যবহার হবে। উপরে যদি doirgho চলকের মান থাকে 5 তাহলে 5 ধরে হিসাব হবে, আর যদি পরে doirgho এর মান 5 এর বদলে 6 করে দেয়া হয়, তাহলে 6 ব্যবহার হবে। এই পরিবর্তন কেবল মান আরোপণের ওইখানে করলেই কাজ হয়ে যাবে, সারা ক্রমলেখতে করতে হবে না। তবে একটা গুরুত্বপূর্ণ বিষয় বলি এখানে doirgho আর prostho চলক দুটিতে মান আরোপণ কিন্তু ক্ষেত্রফল আর পরিসীমার সুত্রের ব্যবহারের আগেই করতে হবে। না করলে সংকলন করার সময় সতর্ক বার্তা (warning message) আসতে পারে, আর ক্রমলেখ চালানোর সময় উল্টোপাল্টা ফলও আসতে পারে।

সবশেষে খেয়াল করো ফলন (output) দেওয়া হয়েছে যেখানে সেখানে উদ্ধৃতি চিহ্ন "" এর ভিতরে যা আছে তা কিন্তু মালা (string)। কাজেই ওইটা কিন্তু ওইভাবেই ফলনে এসেছে এমনকি khetrofol কথাটাও হুবহু এসেছে যেটা কিনা চলকের নামের হুবহু একই রকম। কিন্তু "" উদ্ধৃতির বাইরে যখন khetrofol লেখা হয়েছে একই সারিতে পরের দিকে সেখানে কিন্তু আর khetrofol ফলনে আসে নি, এসেছে সেটাকে চলক ধরলে যে মান হওয়ার কথা সেই 15। কাজেই এটা মনে রাখবে যে চলকের নাম "" উদ্ধৃতির ভিতরে মালা আকারে থাকলে ওইটা আসলে চলকটাকে বুঝায়

না। নামটা যখন উদ্ধৃতির বাইরে থাকে তখন ওইটা একটা নাম হয়, এইক্ষেত্রে একটা চলকের নাম হয় আর ওইটার মান নিয়ে কাজ হয়। একই অবস্থা porishima এর ক্ষেত্রেও। উদ্ধৃতি চিহ্নের ভিতরে থাকা porishima কথাটি হুবহু ফলনে এসেছে কিন্তু উদ্ধৃতির বাইরে থাকা porishima কথাটির বদলে ওটিকে চলক ধরলে যে মান পাওয়া যাবে তা ফলনে এসেছে।

৩.২ ধ্রুবকের ব্যবহার (Using Constants)

একটি বৃত্তের ব্যাসার্ধ দেয়া আছে ৫ সেমি, বৃত্তটির ক্ষেত্রফল নির্ণয়ের জন্য সিপিপিতে একটি ক্রম-লেখ (program) রচনা করো। তোমার ক্রমলেখতে তুমি ব্যাসার্ধের জন্য একটি পূর্ণক (integer) ব্যবহার করবে। আর ক্ষেত্রফলের জন্য প্রথমে পূর্ণক ব্যবহার করে দেখবে কী হয়, তারপর ভগ্নক (fraction) অর্থাৎ সচলবিন্দু সংখ্যা (floating-point number) বা float ব্যবহার করবে। তুমি তো জানো বৃত্তের ক্ষেত্রফল হিসাব করার জন্য আমাদের পাইয়ের মান লাগবে। আমরা ওইটা সরাসরি সংখ্যায় না দিয়ে একটা ধ্রুবক (constant) হিসাবে ব্যবহার করবো, কারণ পাইয়ের মান তো কখনো বদলাবে না, সব সময় ধ্রুবক থাকবে। পাইয়ের মান যেহেতু ভগ্নক আমাদের ধ্রুবকটি তাই হবে float ধ্রুবক। চলো আমরা এবার তাহলে ক্রমলেখটি দেখি।

ফিরিস্তি ৩.২: ক্রমলেখতে ধ্রুবকের ব্যবহার (Constants in Programs)

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  int bashardho = 5;  // এটি বৃত্তের ব্যাসার্ধের জন্য চলক

  float const pai = 3.1415; // পাইয়ের মানের জন্য ধ্রুবক

  // নীচে আমরা বৃত্তের ক্ষেত্রফলের সূত্র লিখছি
  int khetrofol = pai * bashardho * bashardho;

  // এবার আমরা পর্দায় ফলন দেখাবো।
  cout << "britter khetrofol " << khetrofol
       << " borgo cm" << endl;

  return EXIT_SUCCESS;
}
```

ফলন (output)

```
britter khetrofol 78 borgo cm
```

তো হয়ে গেল আমাদের বৃত্তের ক্ষেত্রফল নির্ণয়ের ক্রমলেখ। এই রকম চলক (variable) আর ধ্রুবক (constant) ব্যবহার না করেই তুমি কিন্তু ক্রমলেখ লিখতে পারতে, তাই না! আমরা কিন্তু সেটা আগের পাঠে এটা আলোচনা করেছি। সেক্ষেত্রে main() বিপাতকের {} বন্ধনী দুটোর মধ্যে return EXIT_SUCCESS; এর আগে মাত্র এক সারিতে cout << "britter khetrofol " << 3.1415 * 5 * 5 << " borgo cm" << endl; লিখলেই আমাদের কাজ হয়ে যেতো। কিন্তু আমরা সেটা না করে কেন চলক ব্যবহার করছি সেটাও ওই একই পাঠে আলোচনা করেছি।

এবার আসি আমরা যে ক্রমলেখটি লিখলাম সেটার বিস্তারিত আলোচনায়। আমরা #include , using namespace, int main(), return ইত্যাদি সম্পর্কে ইত্যমধ্যে জেনেছি আগের পাঠ-গুলো থেকে, কাজেই আমাদের আলোচনা সীমাবদ্ধ থাকবে main() বিপাতকে আর যা যা লিখেছি সেই বিষয়গুলোতে। তো চলো আমরা এবার সারির পরে সারি ধরে আলোচনা করি।

উপরে যেমন বলা হয়েছে, সেই অনুযায়ী আমরা প্রথমে ব্যাসার্ধের জন্য একটা চলক নিয়েছি bashardho নামে যেটি হবে int ধরনের অর্থাৎ পূর্ণক বা পূর্ণ সংখ্যা। বৃত্তের ব্যাসার্ধ যদি তোমার ভগ্নক হয়, তুমি চাইলে int ব্যবহার না করে float ব্যবহার করতে পারো। আগের পাঠের সাথে এই পাঠে একটা বিষয় খেয়াল করো, আমরা কিন্তু ব্যাসার্ধ bashardho চলকের মান আলাদা সারিতে না দিয়ে যেখানে চলক ঘোষনা (variable declare) করেছি সেখানেই = চিহ্ন দিয়ে মান আরোপ করেছি অর্থাৎ bashardho এর মান সরাসরি 5 হয়ে গেছে। এটাকে বলা হয় চলকের আদি মান আরোপণ (initial value assignment)। এটা করার দুটো সুবিধা: একটা হলো আমাদের দুইটা আলাদা সারিতে দুইবার লিখতে হলো না, আরেকটা হলো চলকে উল্টাপাল্টা মান থাকার কারণে ক্রমলেখয়ে ভুল হিসাব করার সম্ভাবনা কমে গেল। জেনে রাখো চলক ঘোষনার সাথে সাথে কোন মান না দিয়ে না দিলেও ওখানে উল্টা পাল্টা একটা মান থাকে, কী মান থাকবে আমরা কিন্তু কোন ভাবেই আগে থেকে সেটা জানিনা, পুরাই উল্টাপাল্টা একটা মান হতে পারে সেটা। আর ভুলক্রমে ওই চলকে যদি পরে আর মান আরোপ (assign) করা না হয়, অথবা যদি আরোপ করার আগেই অন্য কোন হিসাবে চলকটি ব্যবহার করা হয়, তাহলে সঙ্গত কারণেই উল্টাপাল্টা মানটি কাজে লাগিয়ে একটা উল্টাপাল্টা ফলাফল আসবে, যেটা আমরা কখনোই চাই না।

ব্যাসার্ধের জন্য চলক নেয়ার পরে আমরা পাইয়ের মান রাখার জন্য একটি float const ধরনের ধ্রুবক নেবো যার নাম pai। পাইয়ের মান যেহেতু ভগ্ন সংখ্যা আমাদের তাই float নিতে হবে, আর পাইয়ের মান যেহেতু সব সময় ধ্রুবক তাই আমরা float এর পরে const লিখে দিতে চাই। তুমি যদি const না লিখো তাহলে কিন্তু এটা একটা চলকের মতো কাজ করবে।

```
int cholok = 15;  // একটা চলক ঘোষনা করে যার মান দিলাম 15
int const dhrubok = 20; // একটা ধ্রুবক ঘোষনা করলাম মান 20

// এখন পর্যন্ত চলক cholok এর মান 15, নীচে নতুন মান দেবো 23
// আবার মান আরোপ না করা পর্যন্ত cholok এর মান থাকবে 23

cholok = 23;     // এটা করা যাবে

// এখন পর্যন্ত ধ্রুবক dhrubok এর মান 20, নীচে নতুন মান দেবো 25
// কিন্তু ক্রমলেখ সংকলন (compile) করার সময় আমরা ত্রুটিবার্তা পাবো।
// cpp.sh দিয়ে সংকলন করলে ত্রুটিবার্তাটি নিম্নরূপ হতে পারে
// error: assignment of read-only variable 'dhrubok'

dhrubok = 25;    // এটা করা যাবে না, ত্রুটি বার্তা আসবে
```

উপরের ক্রমলেখ লক্ষ্য করো। চলক আর ধ্রুবকের মধ্যে তফাৎ হলো চলকের (variable) মান ঘোষনার সময় একবার আরোপ করা যায়, আর তারপরেও যতবার ইচ্ছা ততবার নতুন নতুন মান আরোপ (assign) করা যায়। কিন্তু ধ্রুবকে (constant) একটা মান কেবল ঘোষনা করার সময় বলে দেওয়া যায়, ক্রমলেখতে পরে আর কোথাও ওই ধ্রুবকের মান বদলে নতুন মান আরোপ (assign) করা যায় না। যদি করো তাহলে সংকলক (compiler) ত্রুটি বার্তা (error message) দেখাবে। তো আমরা যেহেতু জানি যে পাইয়ের মান সবসময় ধ্রুবক, এটার মান আমাদের কখনো বদল হবে না, আমরা তাই এটাকে চলক হিসাবে ঘোষনা না করে ধ্রুবক হিসাবে ঘোষনা করবো।

আশা করা যায় চলক আর ধ্রুবকের পার্থক্য পরিস্কার হয়েছে। এবার দেখো আমাদের বৃত্তের ক্ষেত্রফলের ক্রমলেখতে আমরা ক্ষেত্রফলের জন্য khetrofol নামে একটা চলক নিয়েছি, যার প্রকরণ হল int বা পূর্ণক. যদিও আমরা জানি পাইয়ের মান ভগ্নক হওয়ার কারণে আমাদের ফলাফল আসলে একটি ভগ্নক হবে। এইটা আমরা মুলত পরীক্ষামুলক করছি। তো int নেয়ার কারণে আমরা আমাদের ক্রমলেখয়ের ফলন দেখতে পাবো 78 আসলে হওয়ার কথা 78.5375। এইটা কেন হলো কারণ হলো প্রথমে 78.5375 ঠিক মতো ভিতরে ভিতরে হিসাব হয়ে যাবে, কিন্তু যখন khetrofol চলকের মধ্যে মানটা আরোপ (assign) হবে তখন যেহেতু পূর্ণ সংখ্যা বলে ভগ্নাংশটুকু ঢুকানো যাবে না, তাই ওইটা বাদ পরে যাবে (truncation)। আর মান যেটা আরোপ হবে সেটা হলো বাঁকী পূর্ণাংশটুকু বা 78। তো ভগ্নাংশ সহ সঠিক ক্ষেত্রফল পাওয়ার জন্য khetrofol এর সামনে int না লিখে float লিখে দাও তাহলে দেখবে ঠিক ঠিক 78.5375 ই ফলন হিসাবে চলে আসবে।

উপরের আলোচনায় আমরা তিনটা ব্যাপার শিখলাম: ১) আমরা চলক (variable) না ধ্রুবক (constant) ব্যবহার করবো সেটা; তারপর ২) ঘোষনা করার সাথে সাথে একটা আদি মান দিয়ে দেয়া যাকে বলা হয় আদি মান আরোপণ (initial assignment), আর ৩) কোন চলক বা ধ্রুবকের প্রকরণ কেমন হবে, int না float হবে, পূর্ণক না ভগ্নক হবে সেটা আগে থেকে ধারনা করতে পারতে হবে, আর সেই অনুযায়ী চলক বা ধ্রুবকের প্রকার বলে দিতে হবে, না হলে সঠিক ফলাফল নাও পাওয়া যেতে পারে, যেমন 78.5375 এর বদলে 78 পাওয়া যেতে পারে।

৩.৩ চলক ঘোষনা (Variable Declarations)

এই পাঠে সিপিপিতে একাধিক চলক (variable) আমরা কী ভাবে সহজে ঘোষনা (declaration) করতে পারি তা আলোচনা করবো। আমরা আগে দেখেছি প্রতিটি চলক আলাদা আলাদা করে, এমনকি আলাদা আলাদা সারিতে ঘোষনা করতে। তো সুবিধার জন্য আমরা চাইলে একাধিক চলক এক সারিতেই একটা বিবৃতিতেই ঘোষনা করতে পারি, যদি তাদের সকলের উপাত্ত প্রকরণ (data type) একই হয়, যেমন ওই চলকগুলোর সবই যদি int ধরনের হয় অথবা float ধরনের হয়। উদাহরণ দিয়ে ব্যাপারগুলো পরিস্কার করা যাক। ধরো doirgho, prostho, porishima নামে আমরা তিনটি চলক নিলাম, তিনটা চলকের প্রকরণই int অর্থাৎ পূর্ণক বা পূর্ণসংখ্যা।

```
int doirgho;  // দৈর্ঘ্যের জন্য চলক যা int ধরনের অর্থাৎ পূর্ণক
int prostho;  // প্রস্থের জন্য চলক যা int ধরনের অর্থাৎ পূর্ণক
int porishima;  // পরিসীমার জন্য চলক যা int ধরনের অর্থাৎ পূর্ণক
```

উপরের তিনটি চলকই যেহেতু int ধরনের, কাজেই আমরা ওই তিনটি চলককে চাইলে একটি বিবৃতিতেই (statement) ঘোষনা করতে পারি। সেক্ষেত্রে আমাদের int একবার লিখতে হবে, আর চলকগুলোর নাম একটার পর একটা বির্তি , (comma) দিয়ে লিখতে হবে।

```
int doirgho, prostho, porishima;// সবগুলোই int ধরনের
```

এবার আর একটি উদাহরণ দেখি, যেখানে বৃত্তের ব্যাসার্ধ আর ক্ষেত্রফল বের করতে হবে। তো ব্যাসার্ধ যদি int ধরনের বা পূর্ণক হয় আর ক্ষেত্রফল তো float ধরনের বা ভগ্নক হবেই। কাজেই আমরা এ দুটোকে একটা বিবৃতি (statement) দিয়ে ঘোষনা করতে পারবো না।

```
int bashardho;     // ব্যাসার্ধের জন্য চলক int ধরনের।
float khetrofol;   // ক্ষেত্রফলের জন্য চলক float ধরনের
```

কিন্তু যদি porishima এর মতো bashardho টাও float বা ভগ্নক ধরনের হতো তাহলে আমরা এক বিবৃতি দিয়েই দুটোকে এক সাথে ঘোষনা করতে পারতাম।

```
float bashardho, khetrofol; // ব্যাসার্ধ ও ক্ষেত্রফলের চলক
```

তাহলে একটা ক্রমলেখতেই (program) যদি আমরা আয়তের পরিসীমা আর বৃত্তের ক্ষেত্রফল বের করতে চাই, আমরা দরকারী সবগুলো চলক নীচের মতো করে ঘোষনা করতে পারি, যেখানে int চলকগুলো একটা বিবৃতিতে (statement) থাকবে আর float চলকগুলো আলাদা আরেকটা বিবৃতিতে থাকবে। মনে রেখো আমরা কিন্তু এই পাঁচটি চলকের প্রত্যেককে আলাদা আলাদা বিবৃতিতে লিখতেই পারতাম। এখানে আমরা বরং সেটা না করা নিয়েই আলোচনা করছি।

```
int doirgho, prostho, porishima;  // দৈর্ঘ্য, প্রস্থ, পরিসীমা
float bashardho, khetrofol; // ব্যাসার্ধ ও ক্ষেত্রফল
```

৩.৪ আদিমান আরোপণ (Initial Assignment)

আগের পাঠে আমরা একাধিক চলক (variable) ঘোষনা (declaration) নিয়ে আলোচনা করেছি। এখন আমরা এদের আদি মান (initial value) আরোপ (assign) করার দিকে নজর দেই। আদিমান হল প্রথমবারের মতো যে মান দিয়ে দেওয়া হয় সেই মানটি। ঘোষনা দেয়ার পরে চলকে আলাদা করে আদি মান আরোপ করতে চাইলে আমরা নীচের মতো করে করবো।

```
doirgho = 6;
prostho = 3;
bashardho = 5;
```

অথবা চাইলে এক বিবৃতিতে এক সাথেও করা সম্ভব, বির্তি , (comma) দিয়ে।

```
doirgho = 6, prostho = 3, bashardho = 5;
```

আমরা কিন্তু চাইলে আদিমানগুলো নীচের মতো ঘোষনার সাথে সাথেই দিতে পারতাম।

```
int doirgho = 6, prostho = 3, porishima;
float bashardho = 5, khetrofol;
```

ঘোষনার সাথে সাথে চলকের আদিমান দিলে ক্রমলেখয়ের (program) দক্ষতা অল্প একটু বাড়তে পারে। কারণ ঘোষনার সাথে সাথে আদিমান না দিলেও একটা উল্টাপাল্টা মান তো ভিতরে ভিতরে দেয়াই হয়, পরে যখন আমরা আবার মান দেই, তখন আরেকবার দেওয়া হলো, প্রথমবারেই দেওয়া হলো না। আর ঘোষনার সাথে সাথে আদিমান দিলে, একদম প্রথমবারেই মানটি চলকে দেওয়া হয়ে গেলো। ধ্রুবকের ক্ষেত্রে কিন্তু আদি ও একমাত্র মান ঘোষনার সাথে সাথেই দিতে হবে, পরে দেয়ার কোন সুযোগ নাই, সংকলক (compiler) ত্রুটি বার্তা দেখাবে।

কোন চলক ঘোষনার সাথে সাথে তাতে কোন আদিমান না দিলেও যে উল্টাপাল্টা মান থাকে সেটা কত তা যদি জানতো চাও তবে পরীক্ষা করে দেখতে পারো। ধরো তোমার চলক doirgho। এখন ঘোষনার পরেই cout << "doirgho holo " << doirgho << endl; লিখে ক্রম-লেখ সংকলন (compile) করে চালিয়ে (run) দেখতে পারো। কিন্তু মনে রাখবে প্রতিবার চালালে যে একই মান আসবে তার কোন নিশ্চয়তা নাই, যদি আসে সেটা নেহায়েত কাকতাল।

আমরা আগেই জানি বৃত্তের ক্ষেত্রফল নির্ণয়ের জন্য আমাদের পাইয়ের মান দরকার হবে, যেটি একটি ধ্রুবক (constant) আর পাইয়ের মান আসলেই ভগ্নক বা float। কিন্তু float হওয়া সত্ত্বেও আমরা কিন্তু পাইয়ের জন্য pai নামক চলকটিকে bashardho আর khetrofol এর সাথে একই বিবৃতিতে ঘোষনা করতে পারবো না। কারণ bashardho ও khetrofol হল চলক (variable) যাদের মান পরে যতবার ইচ্ছা বদলানো যাবে আর pai হল ধ্রুবক (constant) যার মান একবার দেওয়ার পরে আর বদলানো যাবে না। পাইয়ের মান তাই আলাদা করে ঘোষনা করতে হবে।

```
int doirgho = 6, prostho = 3, porishima;
float bashardho = 5, khetrofol;
float const pai = 3.1415; // পাইয়ের মানের জন্য ধ্রুবক
```

আমাদের যদি একাধিক float constant থাকে সেগুলোকে আবার এক বিবৃতিতেই ঘোষনা করতে পারবো, যেমন পাই আর g এর মান ঘোষনা করছি নীচে। তোমরা জানো জি হল মাধ্যাকর্ষনের ত্বরণের মান, যা নির্দিষ্ট স্থানে মোটামুটি একটা ধ্রুবক।

```
float const pai = 3.1415, g = 9.81;
```

পরিসীমা আর ক্ষেত্রফলের জন্য আমাদের সূত্র লিখতে হবে, সেগুলোকে বির্তি , (comma) দিয়েই এক বিবৃতিতে লেখা সম্ভব, যেমন নীচে লিখলাম।

```
int doirgho = 6, prostho = 3
int porishima = doirgho * prostho;
float bashardho = 5;
float khetrofol = pai * bashardho * bashardho;
float const pai = 3.1415;
```

উপরে যা লিখলাম তাতে কিন্তু একটা ত্রুটি আছে, সংকলন (compile) করতে গেলেই ত্রুটি ধরা পড়বে। ত্রুটিটি হল আমরা pai ঘোষনা করেছি পঞ্চম বিবৃতিতে, কিন্তু pai ব্যবহার করেছি চতুর্থ বিবৃতিতে khetrofol এর সূত্র লিখতে গিয়েই। কোন চলক ঘোষনা করার আগে সেটা ব্যবহার করা যাবে না, সংকলক যখন চলে (run) তখন সে একে একে বিবৃতিগুলো উপর থেকে নীচে আর বামে থেকে ডানে পড়তে থাকে। তো কোন চলক বা ধ্রুবক ঘোষনার আগেই যদি সংকলক তাদের ব্যবহারটা পড়ে ফেলে যেমন pai, তাহলে সে বুঝতে পারবে না pai টা কী জিনিস, এইটা কি চলক নাকি ধ্রুবক, এটা কি int ধরনের নাকি float ধরনের। আমাদের তাই ঘোষনা অবশ্যই আগে করতে হবে, ব্যবহার করতে হবে পরে। তো চলো নীচে আমরা পাইয়ের ঘোষণা আগে লিখি।

```
int doirgho = 6, prostho = 3;
int porishima = doirgho * prostho;
float const pai = 3.1415; // ঘোষনা আগে করা হলো
float bashardho = 5;
float khetrofol = pai * bashardho * bashardho;
```

লক্ষ্য করো doirgho, prostho, bashardho এর জন্য কিন্তু উপরের ওই ত্রুটি ঘটে নি, কারন সুত্রে ব্যবহারের আগেই তো ওগুলো ঘোষনা হয়েছে, যদিও একই সারিতে কিন্তু বামের বিষয়গুলো যেহেতু ডানেরগুলোর থেকে আগে, তাই ঘোষনা আগেই হয়েছে। আমরা অবশ্য উপরের মতো করে সূত্রও একই বিবৃতিতে না দিতে বলবো। তাতে পড়ারও সুবিধা হয়, আবার আগে লিখবো না পরে লিখবো সেই সমস্যাও দুর হয়। তাহলে পুরো ব্যাপারটি দাঁড়াচ্ছে নীচের মতো:

```
int doirgho = 6, prostho = 3, porishima;
float const pai = 3.1415;
float bashardho = 5, khetrofol;

porishima = doirgho * prostho;
khetrofol = pai * bashardho * bashardho;
```

৩.৫ অনুশীলনী সমস্যা (Exercise Problems)

ধারণাগত প্রশ্ন: নীচে কিছু ধারণাগত প্রশ্ন রয়েছে। প্রশ্নগুলোর উত্তর নিজে নিজে বের করবে।

১. ক্রমলেখতে (program) চলক ও ধ্রুবক কেন ব্যবহার করা হয়? কারণগুলো উল্লেখ করো।

২. ক্রমলেখতে (program) চলক ঘোষনা বলতে কী বুঝ, যথাযথ উদাহরণ দিয়ে দেখাও।

৩. চলকে (variable) মান আরোপণ (value assign) বলতে কী বুঝ? ব্যাখ্যা করো।

৪. কখন তুমি চলক (variable) ব্যবহার না করে ধ্রুবক (constant) ব্যবহার করবে?

৫. সিপিপিতে কী ভাবে চলক ও ধ্রুবক ঘোষনা করতে হয়। যথাযথ উদাহরণ দিয়ে দেখাও।

৬. সিপিপিতে কী ভাবে পূর্ণক ও ভগ্নক ধরনের চলক ঘোষনা করতে হয় উদাহরণ দিয়ে দেখাও।

৭. সিপিপিতে এক সারিতে কখন একাধিক চলক ঘোষনা করা যায়? উদাহরণ দিয়ে দেখাও।

৮. চলকে (variable) আদিমান আরোপণ (initial value assignment) কী?

৯. চলকে (variable) আদিমান আরোপণ না করলে সম্ভাব্য কী ফলাফল ঘটতে পারে?

১০. ধ্রুবকে (constant) কেন আদিমান আরোপ করতে হয়, কিন্তু পরে আরোপ করা যায় না?

১১. ফলাফল ভগ্নক (float) কিন্তু int ধরনের পূর্ণক চলকে আরোপ করলে কী ঘটে?

পরিগণনার সমস্যা: নীচে আমরা কিছু পরিগণনার সমস্যা দেখবো। এই সমস্যাগুলো আগে ধৈর্য্য ধরে নিজে নিজে সমাধান করতে চেষ্টা করবে। যখন একেবারেই পারছো না বলে মনে হয় তখনই কেবল সমাধান দেখে নিতে পারো। সমাধানগুলো পরিগণনার প্রশ্নগুলোর শেষে আছে।

১. এমন একটি ক্রমলেখ রচনা করো যেটি দুটি পূর্ণক int ধরনের, আর একটি ভগ্নক float ধরনের চলক ঘোষনা করে। ক্রমলেখটি তারপর চলক তিনটির মান যথাক্রমে $10, 15, 12.6$ আরোপণ করে। পরিশেষে ক্রমলেখটি চলকগুলোর মান পর্দায় দেখায়।

২. ধরো দুটো পূর্ণ সংখ্যা 89 আর 56। এই দুটিকে তুমি দুটো চলকে নিবে, আর তারপর দুইটি চলকে তাদের যোগফল, বিয়োগফল নির্ণয় করবে। সবশেষে সবগুলো চলকের মান ফলনে দেখাবে। সব মিলিয়ে এই রকম একটি ক্রমলেখ রচনা করো।

৩. যদি তাপমাত্রা সেলসিয়াসে c ডিগ্রী হয় আর ফারেনহাইটে হয় f ডিগ্রী, তাহলে আমরা সূত্র লিখতে পারি $f = 9c/5+32$। ধরো তাপমাত্রা 30 ডিগ্রী সেলসিয়াস, তাহলে ফারেনহাইটে এটি কত হবে? তোমার ক্রমলেখতে তুমি ভগ্নক float ধরনের চলক ব্যবহার করবে।

৪. যদি তাপমাত্রা ফারেনহাইটে হয় f ডিগ্রী আর সেলসিয়াসে হয় c ডিগ্রী, তাহলে আমরা সূত্র লিখতে পারি $c = 5(f - 32)/9$। ধরো তাপমাত্রা 76 ডিগ্রী ফারেনহাইট, তাহলে সেলসিয়াসে এটি কত হবে? তুমি এখানে ভগ্নক float ধরনের চলক ব্যবহার করবে।

৫. ধরো একটা কাজ করতে তোমার 7 ঘন্টা 15 মিনিট 39 সেকেন্ড লেগেছে। এই সময়কে সেকেন্ডে রূপান্তর করো। তোমার ক্রমলেখতে তুমি 60 সেকেন্ডে এক মিনিট আর 60 মিনিটে এক ঘন্টা এই দুটি বিষয় বুঝানোর জন্য দুটো ধ্রুবক ব্যবহার করবে।

পরিগণনা সমাধান: এবার আমরা পরিগণনার সমস্যাগুলোর সমাধান দেখবো। মনে রাখবে সমাধানগুলো দেয়া হয়েছে তুমি যদি একান্তই নিজে নিজে করতে পারছোনা বলে মনে হয়, তখন কেবল একটু সাহায্য যাতে পেতে পারো তাই। কাজেই পারতপক্ষে সমাধান দেখবে না।

১. এমন একটি ক্রমলেখ রচনা করো যেটি দুটি পূর্ণক int ধরনের, আর একটি ভগ্নক float ধরনের চলক ঘোষনা করে। ক্রমলেখটি তারপর চলক তিনটির মান যথাক্রমে $10, 15, 12.6$ আরোপণ করে। পরিশেষে ক্রমলেখটি চলকগুলোর মান পর্দায় দেখায়।

ফিরিস্তি ৩.৩: চলক ঘোষনার ক্রমলেখ (Program Declaring Variables)

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  int purnok1, purnok2; //  পূর্ণক দুটি এক সাথে ঘোষনা
  float vognok;     //  ভগ্নকটি আলাদা ঘোষনা

  purnok1 = 10, purnok2 = 15; //  পূর্ণকে মান আরোপণ
  vognok = 12.6;          //  ভগ্নকে মান আরোপণ

  cout << "purnok duti ";   // এখানে endl দেই নাই
  cout << purnok1 << " " << purnok2 << endl;
  cout << "vognok holo " << vognok << endl;

  return EXIT_SUCCESS;  //  সফল সমাপ্তি
}
```

ফলন (output)

```
purnok duti 10 15
vognok holo 12.6
```

২. ধরো দুটো পূর্ণ সংখ্যা 89 আর 56। এই দুটিকে তুমি দুটো চলকে নিবে, আর তারপর দুইটি চলকে তাদের যোগফল, বিয়োগফল নির্ণয় করবে। সবশেষে সবগুলো চলকের মান ফলনে দেখাবে। সব মিলিয়ে একটি ক্রমলেখ রচনা করো।

আমরা এই ক্রমলেখতে কেবল দরকারী অংশটুকু দেখাচ্ছি। ধরে নিচ্ছি যে তুমি দরকারী শির নথি (header) অন্তর্ভুক্ত করা, নামাধার std ব্যবহার, main বিপাতক লেখা ও মান ফেরত দেয়া ইত্যমধ্যে ভালো করে শিখে ফেলেছো। তো তুমি যদি সত্যি নীচের লেখা ক্রম-লেখ সংকলন করে চালাতে চাও, তোমাকে কিন্তু আগে include, namespace, main, return ওইগুলো লিখে নিতে হবে, তারপর main বিপাতকের ভিতরে return এর আগে তুমি আমাদের নীচের অংশগুলো লিখে নিবে। তারপর সংকলন করে ক্রমলেখ চালাবে।

ফিরিস্তি ৩.৪: পাটিগণিতের অণুক্রিয়ার ক্রমলেখ (Arithmetic Program)

```
int prothom = 89, ditiyo = 56;

int jogfol = prothom + ditiyo;
int biyogfol = prothom - ditiyo;

cout << "prothom holo " << prothom;
cout << " ditiyo holo " << ditiyo;
cout << endl;

cout << "jogfol holo " << jogfol;
cout << " biyogfol holo " << biyogfol;
cout << endl;
```

৩. যদি তাপমাত্রা সেলসিয়াসে c ডিগ্রী হয় আর ফারেনহাইটে হয় f ডিগ্রী, তাহলে আমরা সূত্র লিখতে পারি $f = 9c/5+32$। ধরো তাপমাত্রা 30 ডিগ্রী সেলসিয়াস, তাহলে ফারেনহাইটে এটি কত হবে? তোমার ক্রমলেখতে তুমি ভগ্নক float ধরনের চলক ব্যবহার করবে।

ধরে নিচ্ছি প্রথম সমস্যার সমাধান দেখে তুমি ক্রমলেখয়ের কাঠামো দাঁড় করতে পারবে।

ফিরিস্তি ৩.৫: সেলসিয়াস থেকে ফারেনহাইটে রূপান্তর (Celcius to Fahrenheit)

```
float c = 30, f = 9 * c / 5 + 32;
```

৪. যদি তাপমাত্রা ফারেনহাইটে হয় f ডিগ্রী আর সেলসিয়াসে হয় c ডিগ্রী, তাহলে আমরা সূত্র লিখতে পারি $c = 5(f - 32)/9$। ধরো তাপমাত্রা 76 ডিগ্রী ফারেনহাইট, তাহলে সেলসিয়াসে এটি কত হবে? তুমি এখানে ভগ্নক $float$ ধরনের চলক ব্যবহার করবে।

ধরে নিচ্ছি প্রথম সমস্যার সমাধান দেখে তুমি ক্রমলেখয়ের কাঠামো দাঁড় করতে পারবে।

ফিরিস্তি ৩.৬: ফারেনহাইট থেকে সেলসিয়াসে রূপান্তর (Fahrenheit to Celcius)

```
float f = 76, c = 5*(f - 32) / 9;
```

৫. ধরো একটা কাজ করতে তোমার ৭ ঘন্টা ১৫ মিনিট ৩৯ সেকেন্ড লেগেছে। এই সময়কে সেকেন্ডে রূপান্তর করো। তোমার ক্রমলেখতে তুমি ৬০ সেকেন্ডে এক মিনিট আর ৬০ মিনিটে এক ঘন্টা এই দুটি বিষয় বুঝানোর জন্য দুটো ধ্রুবক ব্যবহার করবে।

ধরে নিচ্ছি প্রথম সমস্যার সমাধান দেখে তুমি ক্রমলেখয়ের কাঠামো দাঁড় করতে পারবে।

ফিরিস্তি ৩.৭: সময়কে সেকেন্ডে রূপান্তর (Convert Time to Seconds)

```
int ghonta = 7, minit = 15, sekend = 39;
int const ghontaiMinit = 60, miniteSekend = 60;

int motMin = ghonta * ghontaiMinit + minit;
int motSek = motMin * miniteSekend + sekend;
```

৩.৬ গণনা পরিভাষা (Computing Terminologies)

- আদি মান (initial value)
- আরোপণ (assign)
- ঘোষনা (declaration)
- চলক (variable)
- ধ্রুবক (constant)
- ভগ্নক (fraction)
- মান (value)
- মান আরোপণ (value assign)
- সচলবিন্দু (floating-point)
- চলক ঘোষণা (var declaration)

অধ্যায় ৪

শনাক্তকের নামকরণ (Naming Identifiers)

নামে কী আসে যায় কর্মে পরিচয়। আপনার কাজই নির্ধারণ করে দেবে আপনার পরিচয়। আপনার নাম পরিচয় হবে আপনার কাজের কারণেই। ক্রমলেখ (program) লিখতে গিয়ে আমরা তাই চলক (variable), ধ্রুবক (constant), বিপাতক (function) সহ যে কোন কিছুর নাম দেই তাদের কী কাজে লাগানো হবে সেটা মাথায় রেখে।

৪.১ সুগঠিত নাম (Well-formed Names)

সিপিপিতে চলক ও ধ্রুবকের ব্যবহার তুমি ইত্যমধ্যে শিখে ফেলেছো। আর চলকের নাম কী রকম দিতে হবে সেটাও আগে একটু জেনেছো। এখন আমরা বিস্তারিত ভাবে শিখব সিপিপিতে কী ভাবে চলক বা ধ্রুবকের নাম দিতে হয়, বিশেষ করে নামের গঠনরীতি (syntax) কেমন অর্থাৎ নামে কী রকম অক্ষর থাকতে পারবে অথবা পারবে না। আমরা আপাতত কেবল main বিপাতক (function) নিয়ে কাজ করছি। কিন্তু ভবিষ্যতে আমরা যখন নিজেদের জন্য নানান বিপাতক তৈরী করবো, তখন বিপাতকের নামকরণের জন্যেও চলক বা ধ্রুবকের নাম তৈরীর নিয়মগুলোই কাজে লাগবে। চলক বা ধ্রুবক বা বিপাতক যাইহোক নাম কে বলা হয় শনাক্তক (identifier)।

সিপিপিতে কোন শনাক্তকের (identifier) নামে কেবল ১) ইংরেজী বর্ণমালার বড় হাতের অক্ষর A-Z, ২) ইংরেজী বর্ণমালার ছোট হাতের অক্ষর a-z, ৩) ইংরেজী অংক 0-9 আর ৪) নিম্নদাগ (underscore) _ থাকতে পারবে। তবে শনাক্তকের নামের প্রথম অক্ষর আবার অংক 0-9 হতে পারবে না, প্রথম অক্ষর ছাড়া অন্য যে কোন অক্ষর হিসাবে অংকগুলো ব্যবহার করা যাবে। সুতরাং বোঝাই যাচ্ছে প্রথম অক্ষর যে কোন বর্ণ A-Z বা a-z অথবা নিম্নদাগ (underscore) _ হতে পারবে। আর তারপরের যে কোন অক্ষর বর্ণ বা অংক বা নিম্নদাগ হতে পারবে। সিপিপিতে শনাক্তকের নামের দৈর্ঘ্য নিয়ে কোন বিধিনিষেধ নেই তবে ক্রমলেখ (program) সংকলনে (compile) কী সংকলক ব্যবহার করা হচ্ছে তার ওপর এটা নির্ভর করতে পারে। cpp.sh দিয়ে সংকলন করলে কোন বিধি নিষেধ নেই, মাইক্রোসফট c++ দিয়ে সংকলন করলে ২০৪৮ অক্ষর পর্যন্ত হতে পারে। যাইহোক আমরা এখানে গঠনরীতি অনুযায়ী বৈধ ও অবৈধ কিছু নাম দেখবো।

অবৈধনাম	কারণ
12	নামের সবগুলোর অক্ষর অংক হতে পারবে না
12cholok	নামের প্রথম অক্ষর অংক হতে পারবে না
amar cholok	নামের মাঝখানে কোন ফাঁকা (space) থাকতে পারবে না
ama;cho+k	বর্ণ, অংক, নিম্নদাগ ছাড়া অন্য কোন প্রতীক থাকতে পারবে না

ক্রমলেখতে (program) অবৈধনাম ব্যবহার করলে কী হয়? করে দেখো কী হয়! সংকলক (compiler) ত্রুটিবার্তা (error message) দিবে, আর তোমাকে নামটি ঠিক করতে হবে। তাহলে এখন থেকে তোমার ক্রমলেখতে নাম দেওয়ার সময় নামের এই গঠনরীতি গুলো মেনে চলবে।

বৈধনাম	কারণ
p	একটাই অক্ষর সেটি ছোট হাতের বর্ণ
P	একটাই অক্ষর সেট বড় হাতের বর্ণ
abc	তিনটা অক্ষর সব ছোট হাতের বর্ণ
ABC	তিনটা অক্ষর সব বড় হাতের বর্ণ
Abc	তিনটা অক্ষর ছোটহাতের বড়হাতের মিশানো
bAc	তিনটা অক্ষর ছোটহাতের বড়হাতের মিশানো
a1bc	তিনটা ছোটহাতের অক্ষর ও একটা অংক, অংকটি শুরুতে নয়
a1Bc	তিনটা ছোটবড় হাতের অক্ষর ও একটা অংক যেটি শুরুতে নয়
a_bc	তিনটা ছোটহাতের অক্ষর ও একটি নিম্নদাগ
_abc	তিনটা ছোট হাতের অক্ষর ও তিনটি নিম্নদাগ
_A_b_c	তিনটা ছোটবড় হাতের অক্ষর ও তিনটি নিম্নদাগ
amar_cholok	ছোটহাতের অক্ষর ও নিম্নদাগ, নামটি অধিক বোধগম্য
_amar_Cholok	ছোটবড় হাতের অক্ষর ও নিম্নদাগ, অধিক বোধগম্য
_amarCholok123	ছোটবড় হাতের অক্ষর, নিম্নদাগ, ও অংক যেটি শুরুতে নয়
amar125cholok	ছোটহাতের অক্ষর ও অংক, অংকটি শুরুতে নয়।

৪.২ অর্থবোধক নাম (Meaningful Names)

সিপিপিতে শনাক্তকের (identifier) নাম কেমন হতে পারে আর কেমন হতে পারে না, আমরা তা আগের পাঠে দেখেছি। এই পাঠে আমরা দেখবো নামের অর্থবোধকতা (semantic)। আমরা যখন কোন নাম দেবো, তখন নামটি অবশ্যই অর্থবহ হওয়া চাই। আমরা আগের একটি পাঠে অল্প একটু আলোচনা করেছি নামের অর্থবোধকতা নিয়ে। এখন আরো বিস্তারিত আলোচনা করছি নামগুলো কেমন হলে ভালো হয় সে সম্পর্কে। চলক (variable) বা ধ্রুবক (constant) বা বিপাতকের (function) নাম সবসময় তার কাজ ও ব্যবহারের দিকে খেয়াল রেখে অর্থবোধক হওয়া উচিৎ। অর্থবোধক না হলে ক্রমলেখ (program) বোঝা আমাদের জন্য কঠিন হয়ে যায়।

অনেকে অতিরিক্ত আগ্রহে যত্রতত্র নিজের নামে বা প্রিয় কারো নামে শনাক্তকের নামকরণ করে থাকে যেমন gonimia1, gonimia2, ইত্যাদি। তো এই চলক দুটোর একটা যদি ব্যাসার্ধের জন্য আরেকটা যদি ক্ষেত্রফলের জন্য ব্যবহার করা হয়, তাহলে চলকের নাম থেকে মোটেও বুঝা যাবে না কোন নামটি কী কাজে ব্যবহৃত হচ্ছে। ব্যাসার্ধের জন্য বরং radius বা bashardho অথবা নিদেনপক্ষে r বা b ব্যবহার করা যেতে পারে। এক অক্ষরের নাম দেয়া অনেকে পছন্দ করে, কারণ তাড়াতাড়ি লেখা যায়, কিন্তু একই আদ্যাক্ষর যুক্ত একাধিক চলক থাকলে তখন মুশকিল হয়ে যায়। সেক্ষেত্রে ওই অক্ষরের সাথে আরো অক্ষর লাগিয়ে অথবা সংখ্যা লাগিয়ে প্রতিটি নামকে আলাদা করতে হবে, যাতে অন্তত বুঝা যায় কোন চলকটি কী উদ্দেশ্যে ব্যবহার করা হয়েছে।

আমরা যদি দুটো বৃত্ত নিয়ে কাজ করি তাহলে তাদের ব্যাসার্ধের জন্য দুটি চলক হতেই পারে bashardho1 আর bashardho2 তাতে কোন সমস্যা নাই। ব্যাপারটা দীপু নম্বর ২ চলচ্চিত্রের মতো, একজনের নাম দীপু নম্বর ১ আর একজন দীপু নম্বর ২। অথবা কেউ চাইলে নাম দিতে পারে bashardhoA আর bashardhoB। এভাবে একই ধরনের কাজে ব্যবহার হবে এরকম অনেকগুলো চলক লাগলে আমরা সংখ্যা লাগিয়ে বা বর্ণ লাগিয়ে আলাদা আলাদা নাম তৈরী করে নিবো। এর জন্য অবশ্য সাজন (array) নামে আলাদা একটা ধারণা আছে, যেটা আমরা পরে জানবো। সাজন ব্যবহার করে আমরা সংখ্যা লাগিয়ে যত ইচ্ছা ততগুলো একই ধরনের নাম পাই। অনেকে অতিরিক্ত অলস হয়ে অথবা যে কোন কারণে শনাক্তকের নাম করণ করতে থাকে a, b, c , p, q, r, i, j, k, x, y, z ইত্যাদি একের পর এক অক্ষর দিয়ে। এটা একটা খুবই বাজে অভ্যাস। এইরকম শনাক্তক মোটেও অর্থবোধক নয়। এগুলো থেকে বুঝার কোন উপায় নেই কোন চলকটি ঠিক কী কাজে ব্যবহার করা হচ্ছে। সবসময় এরকম নামকরণ থেকে দুরে থাকবে।

এখানে প্রশ্ন করতে পারো: নামকরণে কি সবসময় একটা মাত্র শব্দই ব্যবহার করবো? একের অধিক শব্দ ব্যবহার করবো না? উত্তর হচ্ছে অর্থবোধক করার জন্য তুমি দরকার মতো একাধিক শব্দ অবশ্যই ব্যবহার করবে, এইটা খুবই ভালো অভ্যাস। আর সেক্ষেত্রে যাতে প্রতিটি শব্দ খুব সহজে বোঝা যায় সে জন্য তোমার কিছু কৌশল অবলম্বন করতে হবে। একটা কৌশল হলো দুটি শব্দের মাঝে একটি নিম্নদাগ _ দেওয়া যেমন amar_cholok। আরেকটি কৌশল হল প্রতিটি শব্দের প্রথম অক্ষরটি বড়হাতের দেওয়া আর অন্যগুলো ছোট হাতের, যেমন AmarCholok তবে চাইলে একদম প্রথম শব্দের প্রথম অক্ষরটি ছোটহাতেরও রাখতে পারো যেমন amarCholok। নীচের সারণীতে আমরা কিছু অর্থবোধক নামের উদাহরণ দেখবো।

নাম	যথোপযুক্ততার কারণ
sum	যোগফলের জন্য sum চলকের ইংরেজী নাম
jogfol	যোগফলের জন্য jogfol চলকের বাংলা নাম
bijor_songkhar_jogfol	নিম্নদাগ _ দিয়ে অর্থবোধক শব্দ আলাদা হয়েছে
odd_number_sum	নিম্নদাগ _ দিয়ে অর্থবোধক শব্দ আলাদা হয়েছে
Bijor_Shongkhar_Jogfol	নিম্নদাগ _ দিয়ে আলাদা, বড়হাতের আদ্যাক্ষর
BijorShongkharJogfol	বড়হাতের প্রথম অক্ষর দিয়ে আলাদা আলাদা
bijorShongkharJogfol	এটি অনেক প্রচলিত ও অনেকেরই পছন্দের

৪.৩ লিপি সংবেদনশীলতা (Case Sensitivity)

সিপিপি ভাষা একটি লিপি সংবেদনশীল (case sensitive) ভাষা। এই কথার অর্থ কী? সিপিপিতে বড়হাতের ছোটহাতের অক্ষর কি ভিন্নভিন্ন ধরা হয়, নাকি ইংরেজীর মতো একই ধরা হয়?

```
barek is going home
BAREK IS GOING HOME
Barek Is Going Home
```

আগের কয়েকটি পাঠে চলক (variable) বাধ্রুবক (constant) বা বিপাতকের (function) নাম, এককথায় শনাক্তকের (identifier) নামকরণ নিয়ে আমরা আলোচনা করেছি। নামকরণের নিয়মগুলো আলোচনা করার সময় দেখেছি যে কোন শনাক্তকের নামকরণে আমরা চাইলে বড়হাতের বর্ণ A-Z, ছোটহাতের বর্ণ a-z, অংক 0-9, আর নিম্নদাগ _ ব্যবহার করতে পারবো। একই নামে বড়হাতের ছোটহাতের অক্ষর মিশিয়েও নামকরণ করতে পারবো। এমতাবস্থায় প্রশ্ন হচ্ছে কোন নাম ইচ্ছামতো একবার বড়হাতের অক্ষরে অথবা ছোট হাতের অক্ষরে অথবা আরেকবার কিছু অক্ষর

ছোটহাতের কিছু অক্ষর বড় হাতের এইভাবে লিখতে পারবো কিনা। বিশেষ করে আমরা জানি ইংরেজীতে আমি ছোট হাতেরই লিখি আর বড় হাতেরই লিখি শব্দটা আসলে একই থাকে, সিপিপিতেও কি তাই? আমরা বরং উদাহরণ দিয়ে ব্যাপারটা দেখি। ইংরেজীতে ছোট হাতের বড় হাতের অক্ষর আলাদা হলেও ওগুলো কেবলই সৌন্দর্যবর্ধন মুলক। উপরের তিনটে ইংরেজী বাক্য তাই একই।

এবার আমরা সিপিপি ভাষায় ছোট হাতের বড় হাতের অক্ষরের ব্যবহার দেখি। নীচের নামগুলোর প্রত্যকেটি সিপিপি ভাষায় আলাদা আলাদা নাম হিসাবে ধরা হবে।

```
amarcholok, amarCholok, AmarCholok, amar_cholok,
Amar_Cholok, amarChoLok, AmarChOlOk
```

সিপিপিতে উপরের একটা নাম দিয়ে যে চলক বা ধ্রুবক বা বিপাতককে বুঝানো হবে অন্য নাম দিয়ে ওইটাকে বুঝানো যাবে না, বরং অন্য একটা বুঝানো হয়ে যাবে। মোট কথা দুটো নামের একটা অক্ষরেও যদি এদিক সেদিক থাকে তাহলে নামদুটো আসলে আলাদা। দুটোকে একই জিনিসের নাম হিসাবে ধরে নেয়া যাবে না। সুতরাং ক্রমলেখ (program) লেখার সময় খেয়াল রাখবে যাতে একটা চলককে বুঝাতে গিয়ে কেবল বড়হাতের ছোটহাতের বর্ণের ভিন্নতার কারণে আরেকটাকে বুঝিয়ে না ফেলো, তাতে সব ভজঘট লেগে যাবে। তোমার ক্রমলেখও উল্টাপাল্টা ফলাফল দিবে। আবার ধরো তোমার একটাই চলক যার নাম amarcholok, কিন্তু পরে তুমি লিখেছো amarCholok। এই অবস্থায় সংকলন (compile) করলে তোমাকে "amarCholok is not declared" এইরকম ত্রুটিবার্তা (error message) দিবে। তোমাকে তখন amarCholok এর বদলে amarcholok লিখে ঠিক করতে হবে। ক্রমলেখ তৈরীর সময় আমরা প্রায়শই এইরকম ভুল করে থাকি।

উপরের এই নিয়ম জানার পরে তুমি হয়তো মনে করবে এইটা তো ভালোই। আমার যদি দুইটা বৃত্তের ব্যাসার্ধের জন্য চলক লাগে একটার নাম দিবো bashardho আর একটার নাম দিবো Bashardho। হ্যাঁ, তুমি সেটা দিতেই পারো। সিপিপি যেহেতু দুইটাকে আলাদা আলাদা চলক হিসাবে ধরে নিবে, তাই এই দুটো হলো দুটো বৈধ আলাদা নাম। তবে অর্থবোধকতার দিক ভেবে তুমি হয়তো এরকম নাম করণ থেকে দূরে থাকার চেষ্টা করবে। একটা অক্ষর বড় বা ছোটহাতের কেবল এই অল্প একটুখানি ভিন্নতা দিয়ে আসলে তেমন বেশী অর্থবোধক পার্থক্য তৈরী করা যায় না, ফলে ক্রমলেখ (program) পড়া কঠিন হয়। আর একটা ব্যাপার: চলকের নামকরণে বড়হাতের ছোটহাতের অক্ষর মিশাতে তো পারোই যেমন AmarCholok, কিন্তু এমন ভাবে মিশিও না যে পড়াটা খুব কঠিন হয়ে যায়, যেমন AmArChOlOk, এই রকম নাম চট করে পড়া আসলে সম্ভব না, বরং এইরকম নাম যন্ত্রনাদায়ক। কাজেই সবমিলিয়ে সহজ ও সুন্দর নাম দিবে, কেমন!

৪.৪ সংরক্ষিত ও চাবি শব্দ (Reserved & Key Words)

সংরক্ষিত শব্দ (reserved word) বা চাবি শব্দ (key word) কী? আমি কি চলক (variable), ধ্রুবক (constant) বা বিপাতকের (function) এর শনাক্তক (identifier) হিসাবে সংরক্ষিত শব্দ বা চাবি শব্দ ব্যবহার করতে পারবো? সিপিপিতে সংরক্ষিত শব্দ বা চাবি শব্দ কোনগুলো?

সংরক্ষিত শব্দ বিষয়ে আলোচনার আগে আমরা একটা গল্প বলে নেই। এক বাড়িতে থাকে জামাই-বউ আর তাদের সাথে থাকে বড় কুটুম অর্থাৎ বউয়ের ভাই বা জামাইয়ের শ্যালক। তো সেই শ্যালকের নাম হল দুলাল। একদিন জামাই বেচারা তার বউয়ের কষ্ট লাঘব করার জন্য একজন কাজের ছেলে নিয়ে আসে। বউ জিজ্ঞেস করে "এই ছেলে তোমার নাম কী?" কাজের ছেলে বলে তার নাম দুলাল। বউ তখন জামাইকে বলে ছেলেটির নাম বদলাতে হবে। জামাই অবাক, অবাক কাজের ছেলেটিও। তার নাম দুলাল, ভালোই তো নামটি, সেটা বদলাতে হবে কেন। বউ জামাইকে বকতে থাকে "তুমি জানো না আমার ভাই অর্থাৎ তোমার শ্যালকের নাম দুলাল"। যে বাসায় শ্যালকের

নাম দুলাল, সেই বাসার কাজের ছেলের নাম দুলাল হয় কেমনে, শ্যালক হলো বড় কুটুম, তার কী এত বড় অসম্মান করা যায়! আর জামাইয়ের নাম হলো কাদের। তো বউ আরো এক কাঠি বাড়িয়ে বলতে থাকে ঠিক আছে কাজের ছেলের নাম বদলে কাদের রাখা হউক, দেখি জামাইয়ের কেমন লাগে। তারপর জামাইয়ের সামনেই কাজের ছেলেকে বলে "এই এখন থেকে তোর নাম দিলাম কাদের।" তারপর হেঁড়ে গলায় ডাকতে থাকে "কাদের, এই কাদের, এই দিকে আয়।" কেমন একটা বেড়াছেড়া অবস্থা। শেষ পর্যন্ত ঠিক হয় এক বাসায় দুইটা দুলাল তো হতে পারেনা, একজনের নাম বদলাতে হবে। আর বাসার বড় কুটুমের নাম তো আর বদলানো যাবে না কোন ভাবেই, ওটা সংরক্ষিত নাম, কাজেই বদলাতে হবে কাজের ছেলের নাম। সুতরাং কাজের ছেলের নাম দেয়া হয় দুলাল্যা। তাহলে শ্যালকের নাম দুলাল, আর কাজের ছেলের নাম দুলাল্যা।

সিপিপি ভাষায় গঠন কাঠামো ঠিক রাখার জন্য কিছু সুনির্দিষ্ট শব্দ আছে। আমরা ইত্যমধ্যে এরকম কিছু শব্দ ব্যবহার করেছি। যেমন return, int, float। এই শব্দগুলোর অর্থ সিপিপি ভাষাতে আগে থেকে সুনির্দিষ্ট, যেমন return মানে যখন বিপাতক (function) শেষ হয়, int আর float হল চলকের মান কেমন পূর্ণক বা পূর্ণ সংখ্যা না ভগ্নক বা ভগ্ন সংখ্যা এইরকম। এই তিনটি ছাড়াও আরো অনেকগুলো এই রকম শব্দ আছে। এই শব্দগুলো চাইলে আমরা নিজেরা আমাদের চলক বা ধ্রুবক বা বিপাতকের নাম হিসাবে ব্যবহার করতে পারবো না। এইগুলো হচ্ছে সংরক্ষিত শব্দ (reserved word)। এই শব্দগুলোকে অন্য কথায় চাবি শব্দও (key word) বলা হয়। তাহলে তোমার ক্রমলেখতে তুমি এইরূপ সংরক্ষিত শব্দ বা চাবি শব্দ শনাক্তকের (identifier) নাম হিসাবে ব্যবহার করবে না। কারণ ওগুলো বড়কুটুম দুলালের নামের মতো। যদি একান্তই দরকার হয় তাহলে দুলাল কে দুলাল্যা বানানোর মতো কিছু যোগ-বিয়োগ করে ভিন্ন শব্দ বানিয়ে ব্যবহার করবে। যেমন return না ব্যবহার করে returnValue ব্যবহার করলে, এইরকম। নীচে আমরা সিপিপির সংরক্ষিত শব্দগুলোর তালিকা দিচ্ছি।

- সংগঠিত পরিগণনায় (structured programming) ব্যবহৃত শব্দ:
  break, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, int, long, register, short, signed, sizeof, static, struct, switch, typedef, union, unsigned, return, void, volatile, while
- বস্তুমুখী (object-oriented) পরিগণনায় ব্যবহৃত শব্দ:
  class, explicit, delete, friend, inline, mutable, namespace, new, opeator, private, protected, public, this, using, virtual
- ত্রুটি সামলানোর (error handling) জন্য শব্দ:
  catch, noexcept, throw, try
- যুক্তি ও বিটপ্রতি অণুক্রিয়ার (logical and bit-wise operators) শব্দ:
  bool, and, and_eq, bitand, bitor, compl, false, not, not_eq, or, or_eq, true, xor, xor_eq
- উপাত্ত প্রকরণ (data type) সংক্রান্ত শব্দ:
  auto, const_cast, decltype, nullptr, dynamic_cast, reinterpret_cast, static_cast, typeid
- ছাঁচ (template) সংক্রান্ত শব্দ:
  export, template, typename

- সংকলন সময়ে (compile-time) ব্যবহৃত হওয়া শব্দ:
  static_assert, constexpr
- পূর্ব প্রক্রিয়কের (preprocessor) জন্য শব্দ:
  if, elif, else, endif, defined, ifdef, ifndef, define, undef, include, line, error, pragma
- বিভিন্ন আকারের অক্ষরের জন্য শব্দ: char16_t, char32_t, wchar_t
- বিবিধ শব্দ: alignas, alignof, asm, concept, requires, thread_local

৪.৫ অনুশীলনী সমস্যা (Exercise Problems)

ধারণাগত প্রশ্ন: নীচে কিছু ধারণাগত প্রশ্ন রয়েছে। প্রশ্নগুলোর উত্তর নিজে নিজে বের করবে।

১. শনাক্তক (identifier) কী? ক্রমলেখতে শনাক্তকের ভুমিকা কী?

২. সিপিপিতে শনাক্তকের (identifier) নাম করণের নিয়মাবলী বর্ণনা করো।

৩. ক্রমলেখতে (program) গঠনগত ভাবে অবৈধ নাম ব্যবহার করলে কী ঘটে?

৪. অর্থবোধক নাম কী? ক্রমলেখতে অর্থবোধক নাম ব্যবহার করা উচিৎ কেন?

৫. সিপিপি একটি লিপি সংবেদনশীল (case sensitive) ভাষা, এর মানে কী ?

৬. সংরক্ষিত ও চাবি শব্দ কী? এগুলো কেন শনাক্তক হিসাবে ব্যবহার করা যায় না?

চর্চামুলক প্রশ্ন: নীচের শব্দগুলো গঠনগত (syntactically) ভাবে শনাক্তকের (identifier) নাম হিসাবে বৈধ নাকি অবৈধ? যদি বৈধ হয় তাহলে অর্থবোধক (meaningful) নাকি অর্থবোধক নয়? অথবা কোন শব্দ কি সংরক্ষিত বা চাবি শব্দ (reserved or key word)? প্রথমে নিজে নিজে উত্তর বের করার চেষ্টা করবে, একান্ত না পারলে নীচের সমাধান দেখবে।

১. void
২. MAX_ENTRIES
৩. double
৪. time
৫. G
৬. Sue's
৭. return
৮. cout
৯. xyz123
১০. part#2
১১. "char"
১২. #include
১৩. a_long_one
১৪. _xyz
১৫. 9xyz
১৬. main
১৭. mutable
১৮. max?out
১৯. Name
২০. name
২১. name_1
২২. Int
২৩. INT
২৪. _SUM

২৫. sum_of_numbers	২৮. printf	৩১. $sum
২৬. firstName	২৯. int	৩২. num^2
২৭. Identifier	৩০. pow	৩৩. num 1

চর্চামুলক উত্তর: উপরের প্রশ্নগুলোর উত্তর এখানে দেয়া হচ্ছে। প্রথমে নিজে নিজে উত্তর বের করার চেষ্টা করবে, একান্ত না পারলে এই সমাধান দেখবে।

১. void : সংরক্ষিত শব্দ, কোন প্রকারেরই না এমন বুঝানো হয়

২. MAX_ENTRIES : বৈধ শনাক্তক, অর্থবোধক

৩. double : সংরক্ষিত শব্দ, বড় আকারের ভগ্নকের জন্য

৪. time : বৈধ শনাক্তক, কিন্তু ভাষালয়ে (library) বিদ্যমান, তাই পরিত্যাজ্য

৫. G : বৈধ শনাক্তক, কিন্তু অর্থ বুঝা যাচ্ছে না, যদি না পারিপার্শ্বিকতা পরিস্কার থাকে

৬. Sue's : অবৈধ শনাক্তক কারণ নামে ' ব্যবহার করা যায় না

৭. return : সংরক্ষিত শব্দ, বিপাতক থেকে ফেরত গমন

৮. cout : বৈধ শনাক্তক, কিন্তু ভাষালয়ে (library) বিদ্যমান, তাই পরিত্যাজ্য

৯. xyz123 : বৈধ শনাক্তক, কিন্তু অর্থবোধক কিনা পরিস্কার না

১০. part#2 : অবৈধ শনাক্তক কারণ নামে # ব্যবহার করা যায় না

১১. "char" : অবৈধ শনাক্তক কারণ নামে " ব্যবহার করা যায় না

১২. #include : পূর্ব-প্রক্রিয়কের (preprocessor) জন্য সংরক্ষিত শব্দ

১৩. a_long_one : বৈধ শনাক্তক, কিন্তু অর্থ সেই ভাবে পরিস্কার নয়।

১৪. _xyz : বৈধ শনাক্তক, কিন্তু অর্থ সেই ভাবে পরিস্কার নয়

১৫. 9xyz : অবৈধ শনাক্তক, নামের শুরুতে অঙ্ক থাকতে পারে না, পরে থাকতে পারে

১৬. main : সংরক্ষিত শব্দ নয়, কিন্তু পরিত্যাজ্য কারণ এটি প্রত্যেক ক্রমলেখতেই থাকে

১৭. mutable : সংরক্ষিত শব্দ, কোন ধ্রুবকও বিশেষ অবস্থায় পরিবর্তন যোগ্য হলে

১৮. max?out : অবৈধ শনাক্তক, নামে ? চিহ্ন থাকতে পারবে না

১৯. Name : বৈধ শনাক্তক, অর্থবোধক, কীসের নাম সেটা পরিস্কার নয়

২০. name : বৈধ শনাক্তক, অর্থবোধক, কীসের নাম সেটা পরিস্কার নয়

২১. name_1 : বৈধ শনাক্তক, অর্থবোধক, কীসের নাম সেটা পরিস্কার নয়

২২. Int : বৈধ শনাক্তক, তবে সংরক্ষিত শব্দ int এর সাথে বিভ্রান্তি দেখা দিতে পারে

২৩. INT : বৈধ শনাক্তক, তবে সংরক্ষিত শব্দ int এর সাথে বিভ্রান্তি দেখা দিতে পারে

২৪. _SUM : বৈধ শনাক্তক, অর্থবোধক, যোগফলের জন্য

২৫. sum_of_numbers : বৈধ শনাক্তক, অর্থবোধক

২৬. firstName : বৈধ শনাক্তক, অর্থবোধক, অনেকের পছন্দের

২৭. Identifier : বৈধ শনাক্তক, অর্থবোধক, কীসের শনাক্তক পরিস্কার নয়

২৮. printf : বৈধ শনাক্তক, অর্থবোধক, কিন্তু ভাষালয়ে (library) বিদ্যমান, তাই পরিত্যাজ্য

২৯. int : সংরক্ষিত শব্দ, পূর্ণক উপাত্ত ধারনের জন্য উপাত্ত প্রকরণ

৩০. pow : বৈধ শনাক্তক, অর্থবোধক, কিন্তু ভাষালয়ে (library) বিদ্যমান, তাই পরিত্যাজ্য

৩১. $sum : অবৈধ শনাক্তক, নামে $ চিহ্ন ব্যবহার করা যায় না

৩২. num^2 : অবৈধ শনাক্তক, নামে ^ চিহ্ন ব্যবহার করা যায় না

৩৩. num 1 : অবৈধ শনাক্তক, নামে ফাঁকা ব্যবহার করা যায় না

৪.৬ গণনা পরিভাষা (Computing Terminologies)

- শনাক্তক (identifier)
- নিম্নদাগ (underscore)
- সাজন (array)
- সংগঠিত (structured)
- পরিগণনা (programming)
- সংগঠিত পরিগণনা (structured programming)
- বস্তুমুখী (object-oriented)
- বস্তুমুখী পরিগণন (object oriented programming)
- ত্রুটি সামলানো (error handling)
- যুক্তি অণুক্রিয়া (logical operators)
- বিটপ্রতি অণুক্রিয় (bit-wise operators)
- উপাত্ত প্রকরণ (data type)
- ছাঁচ (template)
- সংকলন সময় (compile-time)

অধ্যায় ৫

যোগান ও আরোপণ (Input and Assignment)

ক্রমলেখতে (program) উপাত্ত (data) কোথা থেকে আসে? হয় আমরা ক্রমলেখয়ের ভিতরে সরাসরি লিখে দেই, যেমনটি আগের পাঠগুলোতে করেছি, আর না হয় আমরা উপাত্ত ব্যবহারকারী-দের কাছে থেকে যোগান (input) নেই। উপাত্ত যোগান নিয়ে সেটিকে ধারন করার উদ্দেশ্যে আমরা চলকে (variable) আরোপণ (assign) করি যাতে ওই উপাত্ত পরে কাজে লাগানো যায়।

৫.১ উপাত্ত যোগান (Data Input)

সিপিপিতে এমন একটি ক্রমলেখ (program) লিখো যেটি যে কোন আয়তের ক্ষেত্রফল ও পরিসী-মা নির্ণয় করতে পারে। তোমার ক্রমলেখ তুমি মাত্র একবারই সংকলন (compile) করতে পারবে, আর প্রত্যেক আলাদা আয়তের জন্য তুমি ক্রমলেখটি বারবার কেবল চালাতে পারবে, কিন্তু ক্রম-লেখয়ের ভিতরে দৈর্ঘ্য ও প্রস্থ বদলে দিয়ে বারবার সংকলন করতে পারবে না। তারমানে তোমাকে দৈর্ঘ্য ও প্রস্থ যোগান (input) হিসাবে তোমার ক্রমলেখ ব্যবহারকারীর কাছে থেকে নিতে হবে।

উক্ত ক্রমলেখ লেখার আগে চলো আমরা কিছু দরকারী আলোচনা সারি। আমরা যখন কোন গণনা সমস্যার (computing problem) সমাধান করতে চাই, যেমন আলোচ্য ক্ষেত্রে আমরা আয়তের দৈর্ঘ্য ও প্রস্থ জেনে তার ক্ষেত্রফল ও পরিসীমা নির্ণয় করতে চাই, তখন আমরা মুলত একটি ক্রমলেখ (program) ব্যবহার করবো, মানে আমরা ক্রমলেখটি চালাবো (run)। এখন এই ক্রমলেখ হয়তো আমরা নিজেরা তৈরী করবো অথবা অন্য কেউ আমাদের তৈরী করে দিবে। বেশীর ভাগ ক্ষেত্রে ক্রমলেখটি অন্যের তৈরী করা দেয়া, আমরা কেবল ব্যবহারকারী।

ভেবে দেখো ক্রমলেখ তৈরী করা (write) আর চালানো (run) আসলে দুটো ভিন্ন ঘটনা। এই দুটো ঘটনা পরপর একসাথে ঘটবে এরকম সবসময় হয় না। বরং বেশীর ভাগ সময়ে এই ঘটনা দুটো আসলে ভিন্ন দুটি স্থানে ভিন্ন দুটি সময়ে ভিন্ন দুই ব্যক্তির দ্বারা সংঘটিত হয়। তাছাড়া ক্রমলেখ যে চালাবে সে হয়তো কেবল একটা আয়তের ক্ষেত্রফল ও পরিসীমা নির্ণয় করতে চায় না বরং তার হাতে হয়তো অনেক অনেক আয়ত আছে আর সে সবগুলো আয়তের জন্যই ক্ষেত্রফল ও পরিসীমা নির্ণয় করতে চায়। সুতরাং প্রতিটা আয়তের জন্য তার একটা করে আলাদা ক্রমলেখ লাগবে যদি ক্রমলেখয়ের ভিতরে আয়তের দৈর্ঘ্য ও প্রস্থ দিয়ে দেয়া হয়। অথবা তার এমন একটা ক্রমলেখ লাগবে যেটা কোন না কোন ভাবে সবগুলো আয়তের জন্যই কাজ করবে, আর সঠিক ভাবেই করবে অর্থাৎ ক্রমলেখটি মুলত সুত্রের (formula) ওপর নজর দেবে, উপাত্তের (data) ওপর নয়।

৫.১. উপাত্ত যোগান (Data Input)

আমরা উপরে যেসব অবস্থা আলোচনা করলাম সেই সব অবস্থায় ক্রমলেখক (programmer) ক্রমলেখ তৈরী করার সময় জানবেন না আয়তের দৈর্ঘ্য ও প্রস্থ কী হবে, সেটি জানা সম্ভব হবে পরে ব্যবহারকারী যখন ক্রমলেখটি চালাবেন কেবল তখন। প্রশ্ন হচ্ছে এমতাবস্থায় ক্রমলেখক উপাত্ত (data) ছাড়া কী ভাবে ক্রমলেখ তৈরী করবেন। সত্যি বলতে উত্তর তো গণিতেই আছে: চলক (variable) ব্যবহার করে। আর আমরা তো ইত্যমধ্যে ক্রমলেখতে চলক ব্যবহার করেছিই। আমাদের কেবল যেটা করা দরকার তা হলো ক্রমলেখয়ের ভিতরে দৈর্ঘ্য বা প্রস্থ সরাসরি লিখে না দিয়ে ওইটা যাতে ব্যবহারকারী ক্রমলেখ চালানোর সময় দিয়ে দিতে পারে সেই ব্যবস্থা করা। নীচের ক্রমলেখতে আমরা তাই করেছি। আমরা ব্যবহারকারীর কাছে থেকে চলকের মান উপাত্ত হিসাবে যোগান (input) নিয়েছি। এবার আমরা ওই ক্রমলেখটির সংশ্লিষ্ট অংশটুকু বিশ্লেষণ করি।

ফিরিস্তি ৫.১: উপাত্ত যোগানের ক্রমলেখ (Programs with Data Input)

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  int doirgho;  // আয়তের দৈর্ঘ্যের জন্য চলক
  cin >> doirgho; // দৈর্ঘ্য যোগান হিসাবে নেওয়া হবে

  int prostho;  // আয়তের প্রস্থের জন্য চলক
  cin >> prostho; // প্রস্থ যোগান হিসাবে নেওয়া হবে

  // ক্ষেত্রফল ও পরিসীমার সূত্র
  int khetrofol = doirgho * prostho;
  int porishima = 2*(doirgho + prostho);

  //  ক্ষেত্রফল ও পরিসীমা ফলন
  cout <<"khetrofol holo "<< khetrofol << endl;
  cout <<"porishima holo "<< porishima << endl;

  return EXIT_SUCCESS;  // সফল ভাবে ফেরত
}
```

যোগান-ফলন (input-output)

```
13
12
khetrofol holo 156
porishima holo 50
```

উপরের ক্রমলেখতে খেয়াল করো আমরা দৈর্ঘ্যের জন্য একটি চলক doirgho ঘোষনা করেছি, কিন্তু সাথে সাথে তার কোন আদিমান আরোপ (initial value assign) করি নাই। কারণ আগেই যেমন আলোচনা করলাম, আমরা যখন ক্রমলেখ লিখছি তখন আমরা আসলে জানিনা যে doirgho এর মান কতো। আমরা বরং ওইটা ব্যবহারকারীর কাছে থেকে নেবো। আর সে কারণে আমরা cin >> doirgho; লিখেছি। এখানে cin হল console in। সাধারণত যোগান যন্ত্র (input device) চাপনি (keyboard) ও টিপনি (mouse) আর ফলন যন্ত্র (output device) নজরি (monitor) মিলিয়ে হল আমাদের console বা যন্ত্রালয়। তো console in বলতে আমরা এখানে যোগান যন্ত্র বিশেষ করে চাপনি (keyboard) থেকে যোগান (input) নেয়া বুঝাচ্ছি। তাহলে cin ব্যবহারকারীর কাছে থেকে চাপনির মাধ্যমে সংখ্যাটা নিয়ে সেটা doirgho চলকের ভিতরে দিয়ে দিবে। এতে ওই চলকে একরকমের মান আরোপণ (value assign) হয়ে যাবে।

ব্যবহারকারীর কাছে থেকে দৈর্ঘ্য নেবার পরে আমাদের প্রস্থও নিতে হবে। উপরের ক্রমলেখতে খেয়াল করো আমরা দৈর্ঘ্যের মতো করেই প্রস্থের জন্যও prostho নামে একটা int ধরনের চলক ঘোষনা করেছি আর তার পরের সারিতে cin ব্যবহার করে prostho এর মান ব্যবহারকারীর কাছে থেকে নেয়ার কথা লিখেছি। উপরের ক্রমলেখয়ের বাঁকী অংশটুকু তো আগের পাঠের ক্রমলেখগুলোতে যেমন দৈর্ঘ্য ও প্রস্থ ব্যবহার করে ক্ষেত্রফল ও পরিসীমার সূত্র লিখা হয়েছে আর তারপরে ফলন (output) দেখানো হয়েছে ঠিক তেমনই। আমরা সেগুলো আর আলোচনা করছি না।

এবার আমরা আর একটু আলোচনা করি উপরের ক্রমলেখটি সংকলন (compile) করে চালালে কী ঘটবে তা নিয়ে। উপরের ক্রমলেখটি চালালে আমরা দেখব পর্দায় (screen) কিছু আসছে না, চটুলটা (cursor) কেবল লাফালাফি করছে। আমরা এই অবস্থায় দৈর্ঘ্যের মান, ধরা যাক 13 চেপে ভুক্তি (enter) চাপবো। ভিতরে ভিতরে cin ওই মান নিয়ে doirgho চলকের মধ্যে রেখে দিবে। চটুলটা (cursor) তারপরও লাফালাফি করবে। আমরা তখন 12 দিয়ে ভুক্তি (enter) চাপবো, cin ওইটা prostho চলকে রেখে দিবে। তারপর পর্দায় আমরা ফলন দেখতে পাবো। প্রথম সারিতে থাকবে khetrofol holo 156 আর পরের সারিতে porishima holo 50।

উপরে ক্রমলেখতে আমরা চাইলে কিছু সংক্ষিপ্তকরণ করতে পারি। যেমন দৈর্ঘ্য ও প্রস্থ ঘোষনা (declaration) ও যোগান (input) নেয়া চার সারিতে না করে আমরা ওগুলোকে মাত্র দুই সারিতে সারতে পারি। প্রথম সারিতে আমরা চলক দুটো ঘোষনা করবো। আর পরের সারিতে আমরা চলক দুটোর যোগান নিবো। নীচের ক্রমলেখতে (program) এইগুলো দেখানো হলো।

```
int doirgho, prostho;    // আয়তের দৈর্ঘ্যে ও প্রস্থের জন্য চলক
cin >> doirgho >> prostho;  // দৈর্ঘ্য ও প্রস্থ যোগান নেওয়া হবে
```

আর সেক্ষেত্রে ক্রমলেখটি চালানোর সময় যোগান নেয়ার অংশ নিম্নরূপ হবে। লক্ষ্য করবে চটুল (cursor) যখন যোগান নেবার জন্য লাফাতে থাকবে, আমরা তখন 13 ও 12 সংখ্যা দুটি ফাঁকা দিয়ে এক সাথে দিয়েই ভুক্তি (enter) চাপতে পারবো, অথবা চাইলে 13 লিখে ভুক্তি চেপে তারপর 12 লিখে আবার ভুক্তি চাপতে পারবো। আর ফলনের অংশ আগের মতোই হবে।

```
13 12
```

কেউ যদি চায় তাহলে কিন্তু ফলন অংশেও এরকম সংক্ষিপ্তকরণ করতে পারে। যেমন ক্ষেত্রফল ও পরিসীমা চাইলে এক সারিতেই ফলন দিতে পারে।

```
cout << "khetrofol ar porishima holo " << khetrofol
 << " " << porishima << endl; // cout হতে এই পর্যন্ত
 পুরোটা আসলে এক সারিতে
```

তবে সবকিছু একবার cout দিয়ে দেওয়ার চেয়ে আমরা হয়তো দুইবারে দিতে চাইবো।

```
cout << "khetrofol ar porishima holo ";
cout << khetrofol << " " << porishima << endl;
```

উপরের উভয় ক্ষেত্রে পর্দায় ফলন কিন্তু একসারিতেই আসবে।

```
khetrofol ar porishima holo 156 50
```

৫.২ যোগান যাচনা (Input Prompt)

সিপিপিতে এমন একটি ক্রমলেখ (program) রচনা করো যেটি যে কোন আয়তের ক্ষেত্রফল ও পরিসীমা নির্ণয় করতে পারে। তোমার ক্রমলেখ আয়তের দৈর্ঘ্য ও প্রস্থ ব্যবহারকারীর কাছে থেকে যোগান (input) নিবে। আর দৈর্ঘ্য ও প্রস্থ যোগান নেবার আগে তোমার ক্রমলেখ অবশ্যই ব্যবহারকারীকে দৈর্ঘ্য ও প্রস্থের মান জিজ্ঞেস করবে অর্থাৎ যাচনা (prompt) করবে।

ফিরিস্তি ৫.২: যোগান যাচনার ক্রমলেখ (Program with Input Prompt)

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  int doirgho;          // আয়তের দৈর্ঘ্যের জন্য চলক
  cout << "doirgho koto? ";    // মান যাচনা করা হচ্ছে
  cin >> doirgho;         // দৈর্ঘ্য যোগান হিসাবে নেওয়া হবে

  int prostho;          // আয়তের প্রস্থের জন্য চলক
  cout << "prostho koto? "; // মান যাচনা করা হচ্ছে
  cin >> prostho;    // প্রস্থ যোগান হিসাবে নেওয়া হবে

  // ক্ষেত্রফল ও পরিসীমার সূত্র
  int khetrofol = doirgho * prostho;
  int porishima = 2*(doirgho + prostho);

  //  ক্ষেত্রফল ও পরিসীমা ফলন দেয়া হবে
  cout <<"khetrofol holo "<< porishima << endl;
  cout <<"porishima holo "<< porishima << endl;

  return EXIT_SUCCESS;  // সফল ভাবে ফেরত
}
```

যোগান-ফলন (input-output)

```
doirgho koto? 13
prostho koto? 12
khetrofol holo 156
porishima holo 50
```

আগের পাঠের ক্রমলেখতে আমরা চলকের মান ব্যবহারকারীর কাছে থেকে নেয়ার জন্য cin ব্যবহার করেছি। ওই ক্রমলেখটি যখন আমরা চালাই তখন দেখি পর্দায় (screen) কিছু নাই আর চটুলটা (cursor) কেনো যেনো লাফালাফি করছে। সেই অবস্থায় আমরা প্রথমে দৈর্ঘ্যের মান 13 দিয়ে ভুক্তি (enter) চেপেছি। চটুলটা তারপরও লাফালাফি করছিল। আমরা তখন 12 দিয়ে ভুক্তি চেপেছি। তারপর পর্দায় ফলন এসেছিল প্রথম সারিতে khetrofol holo 156 আর পরের সারিতে porishima holo 50। তো এই যে চটুলটা (cursor) লাফালাফি করছিল দৈর্ঘ্য ও প্রস্থের মান নেয়ার জন্য এইটা আমরা বুঝতে পারি কারণ আমরা নিজেরাই এক্ষেত্রে ক্রমলেখটি তৈরী write) করেছি আর নিজেরাই সেটা সংকলন (compile) করে চালাচ্ছি (run)। আমরা এক্ষেত্রে জানি যে আমাদের ক্রমলেখটি প্রথমে দৈর্ঘ্য চাচ্ছে আর সেটা দেবার পর প্রস্থ চাচ্ছে। এবার ভেবে দেখো আমাদের লেখা ক্রমলেখ যদি আমরা ছাড়া অন্য কেউ চালায় (run) তাহলে সে কী ভাবে জানবে চটুলটি (cursor) ওই অবস্থায় কেন লাফাচ্ছে। সে কি আসলেই দৈর্ঘ্য বা প্রস্থ নেয়ার জন্য অপেক্ষা করছে নাকি ভিতরে ভিতরে ঘটনা অন্য কিছু, সে হয়তো অন্য কিছু করছে।

তো ওপরের সমস্যা সমাধানের জন্য আমরা যেটি করবো সেটি হলো আমাদের ক্রমলেখতে cin >> doirgho; লেখার আগে আমরা একটা বার্তা দেখাবো যে আমরা দৈর্ঘ্যের মান চাই। উপরের ক্রমলেখ খেয়াল করো cin >> doirgho; লেখার আগে আমরা cout << "doirgho koto? "; লিখে আসলে সেটাই করতে চাইছি। এই ক্রমলেখ যখন চালানো হবে তখন প্রথমে পর্দায় doirgho koto? দেখা যাবে। আর cout এর শেষে আমরা যেহেতু endl অর্থাৎ end line দেই নাই, চটুলটা (cursor) সেহেতু ওই একই সারিতে লাফাইতে থাকবে, লাফাইতে থাকবে মুলত cin >> doirgho; এর কারণে doirgho এর মান নেয়ার জন্য। আমরা তখন doirgho এর মান দিয়ে ভুক্তি (enter) চাপবো। তাহলে "চটুল কেন লাফায়?" আমরা এই সমস্যার সমাধান করে ফেললাম কেমন! এই যে যোগান (input) নেবার আগে একটা বার্তা দিয়ে ব্যবহারকারীকে জানানো যে আমরা কী যোগান চাই, এই ব্যাপারটিকে বলা হয় যোগান যাচনা (input prompt)। উপরের ক্রমলেখতে খেয়াল করো আমরা প্রস্থের জন্যেও একই ভাবে যোগান (input) নেবার আগে "prostho koto? " বার্তা দিয়ে যোগান যাচনা (input prompt) করেছি। তাহলে এখন থেকে তোমার ক্রমলেখতে যোগান নেবার আগে অবশ্যই যোগান যাচনা করবে, কেমন?

উপরে ক্রমলেখতে আমরা চাইলে কিছু সংক্ষিপ্তকরণ করতে পারি। যেমন দৈর্ঘ্য ও প্রস্থ ঘোষনা (declaration), যোগান যাচনা (input prompt) করা, ও যোগান (input) নেয়া ছয় সারিতে না করে আমরা ওগুলোক মাত্র তিন সারিতে সারতে পারি। প্রথম সারিতে আমরা চলক দুটো ঘোষনা করবো। আর পরের সারিতে আমরা যোগান যাচনা করবো তারপরে সারিতে চলক দুটোর মান যোগান নিবো। নীচে ক্রমলেখতে (program) এইগুলো দেখানো হলো।

```
int doirgho, prostho;      // দৈর্ঘ্যে ও প্রস্থের জন্য চলক
cout << "doirgho o prostho koto? ";// একসাথে যাচনা
cin >> doirgho >> prostho;       // দৈর্ঘ্য ও প্রস্থ যোগান
```

আর সেক্ষেত্রে ক্রমলেখটি চালানোর সময় যোগান নেয়ার অংশ নিম্নরূপ হবে। অর্থাৎ ক্রমলেখ চালালে doirgho o prostho koto? দেখানোর পরে চটুলটা (cursor) যোগান নেবার জন্য লাফাতে থাকবে। আমরা 13 ও 12 সংখ্যা দুটি ফাঁকা দিয়ে এক সাথে দিয়েই ভুক্তি (enter) চাপতে

পারবো, অথবা চাইলে 13 লিখে ভুক্তি চেপে তারপর 12 লিখে আবার ভুক্তি চাপতে পারবো। আর ফলনের অংশ আগের মতোই হবে, কাজেই আমরা সেটা আর দেখাচ্ছি না।

```
doirgho o prostho koto? 13 12
```

৫.৩ মান আরোপণ (Value Assignment)

ক্রমলেখতে (program) চলক নিয়ে তাতে মান আরোপণ (value assign) করলে আসলে কী ঘটে? চলকে একটা মান আগে থেকে আছেই, এমতাবস্থায় আরেকটা মান আরোপ করলে কী ঘটে? একটা চলক থেকে আরেকটা চলকে মান আরোপ করলেই বা কী ঘটে?

```
int amar;
int tomar = 5;
```

উপরে আমরা দুটো চলক ঘোষনা (variable declare) করলাম: একটার নাম amar আর আরেকটার নাম tomar, দুটোই int ধরনের অর্থাৎ পূর্ণক, একটাতে আদিমান (initial value) দিয়ে দিলাম আর একটাতে দিলাম না। আমরা যখন চলক ঘোষনা করি তখন আসলে আমরা গণনির (computer) স্মরনিতে (memory) কিছু জায়গা দখল করি। ধরে নিতে পারো স্মরনি হল একটা রাস্তার পাশে অনেকগুলো একই রকম বাড়ী। কোন চলক ঘোষনা করার সময় আমরা আসলে ওই বাড়ীগুলোর একটা দখল করে সেই বাড়ীটার নাম দিয়ে দেই আমাদের চলকের নামে। তোমরা নিশ্চয় দেখেছো অনেকেরই বাড়ীর নাম থাকে যেমন "শান্তি নীড়"। আমাদের চলক বাড়ীগুলোর নাম amar ও tomar। তো আমরা যখন উপরের দুটো চলক ঘোষনা করলাম তখন স্মরনিতে ওই রকম দুটো জায়গা নিয়ে তাদের নাম দিয়ে দিলাম amar আর tomar। এখন কথা হচ্ছে স্মরণিতে (memory) ওই জায়গায় আমরা আসলে রাখবো কী? উত্তরটাতো সহজ আমরা রাখবো চলকটির মান। যখন আমরা আদিমান দিয়ে দিলাম তখন ওই জায়গাতে আমাদের দেয়া মানটা থাকবে, আর যখন আদিমান দিবো না, তখনও ওই জায়গাটিতে আগে থেকে যাই ছিল তাই থাকবে।

```
amar = tomar;
```

এবার আমরা যদি উপরের মতো করে tomar এর মান amar এ আরোপ করি তাহলে কী ঘটবে? আসলে উপরের এই বিবৃতি (statement) চালানোর পরে amar এর আগের মান মুছে গিয়ে সেটার নতুন মান হয়ে যাবে tomar এর মানের সমান অর্থাৎ amar এর মানও হবে 5। এখানে একটা গুরুত্বপূর্ণ বিষয় বলে রাখতে হবে যে এই যে tomar থেকে amar এ মান আরোপ করা হলো এতে কিন্তু tomar এর মানে কোন পরিবর্তন হবে না। অর্থাৎ tomar এর মান আগের মতো 5-ই থাকবে। আরোপণে (assignment) সমান চিহ্নের বামে যা থাকে সেটাকে লক্ষ্য (target) আর ডানে যেটা থাকে সেটাকে উৎস (source) বলা হয়, কারণ উৎস থেকে মান নিয়ে লক্ষ্যে আরোপ করা হয়। উপরের আরোপণে amar = চিহ্নের বামে তাই এটি লক্ষ্য আর tomar ডানপাশে তাই এটি উৎস। আরোপণের ফলে লক্ষ্যের মান বদলে কিন্তু উৎসের মান বদলে না, একই থাকে।

উপরের ক্রমলেখ (program) আর ফলন (output) লক্ষ্য করো। আমরা প্রথমে চলক x ঘোষনা করে তার আদি মান (initial value) দিয়েছি 3, তারপর চলক y ঘোষনা করে তার আদিমান দিয়েছি x+5 অর্থাৎ 3 + 5 = 8। এই পর্যায়ে ফলন দেখানো হয়েছে x আর y দুটোর মানেরই। ফলনে আমরা দেখতে পাচ্ছি x 3 y 8। তারপর ক্রমলেখতে আমরা লিখেছি x = y * 3; ফলে x এর মান হবে এখন 8 * 3 = 24, আর y এর মান কিন্তু একই থাকবে, কারণ y এর মান আমরা কেবল ব্যবহার করেছি, y এ তো কোন মান আরোপ করি নাই। ক্রমলেখতে

পরের বিবৃতিতে (statement) আমরা x ও y এর তখনকার মান দেখিয়েছি, ফলনে সেটা ঠিকই x 24 y 8 দেখাচ্ছে। ক্রমলেখতে এরপরের বাক্যে আমরা আবার x এ মান আরোপ করেছি x = y + 3 * 4; তো এর ফলে আগের মতোই y এর মান বদল হবে না, কিন্তু x এর নতুন মান হয়ে যাবে 8 + 3 * 4 = 20, যা পরের cout এর মাধ্যমে ফলনে ঠিকই দেখানো হয়েছে x 20 y 8। ক্রমলেখতে এরপর আমরা দেখি y = x * 2; এর ফলে y এর নতুন মান হবে y = 20 * 2 = 40, আর x এর মান এবার আগে যা ছিলো তাই থাকবে, কারণ x এর মান কেবল ব্যবহার করা হয়েছে, x এ কোন মান আরোপ করা হয় নি।

```
int x = 3;                  // আদি মান আরোপ
int y = x + 5;              // আদি মান আরোপ
cout << "x " << x << " y " << y << endl; // মান দেখাও

x = y * 3;                  // পুনরায় মান আরোপ
cout << "x " << x << " y " << y << endl; // মান দেখাও

x = y + 3 * 4;              // পুনরায় মান আরোপ
cout << "x " << x << " y " << y << endl; // মান দেখাও

y = x * 2;                  // পুনরায় মান আরোপ
cout << "x " << x << " y " << y << endl; // মান দেখাও
```

ফলন (output)

```
x 3 y 8
x 24 y 8
x 20 y 8
x 20 y 40
```

সবমিলিয়ে একটা বিষয় দেখা যাচ্ছে আরোপণে = চিহ্নের বামে থাকা লক্ষ্য চলকের (target variable) মান কেবল পরিবর্তন হয়, আর = চিহ্নের ডানে থাকা চলক (variable) বা রাশির (expression) এর মান কোন পরিবর্তন হয় না। আরেকটি ব্যাপার হলো কোন চলকে পরে কোন নতুন মান আরোপ না হওয়া পর্যন্ত আগেরবার যে মান আরোপ করা হয়েছিল সেটাই থাকে।

৫.৪ মান অদল-বদল (Value Swapping)

ধরো তোমার দুটো চলক (variable) আছে x আর y আর তাদের মান যথাক্রমে 12 ও 13। তো তোমাকে এমন কিছু বিবৃতি (statement) লিখতে হবে যাতে ওই বিবৃতিগুলো চালানোর (run) পরে আমরা x আর y এর মান যথাক্রমে 13 আর 12 পাই অর্থাৎ মানদুটো অদল-বদল হয়ে যায়।

```
int x = 12; // x এর মান আরোপ করা হলো
int y = 13; // y এর মান আরোপ করা হলো
```

উপরে আমরা কেবল চলক x আর y ঘোষনা করে তাদের আদিমান হিসাবে 12 ও 13 দিয়ে দিলাম। এখন আমরা এমন কিছু করবো যাতে x আর y মান অদল-বদল হয়ে যায়। প্রথমেই আমরা একটা চটুল সমাধান করি। তোমাদের মধ্যে যারা দুষ্টু ধরনের আর চটপটে তারা সাধারণত এই

সমাধানটি করতে চাইবে। নীচের বিবৃতি দুটো লক্ষ্য করো: আমরা স্রেফ x এর মধ্যে সরাসরি 13 আরোপ করেছি আর y এর মধ্যে 12 আরোপ করেছি। ব্যস হয়ে গেল x আর y এর মান অদল-বদল! আসলে আমরা কী এইটে চেয়েছিলাম? এখানে তো চলক দুটোর মধ্যে একটা থেকে আরেকটাতে মান নেয়ার মতো কোন ঘটনা ঘটে নি, কাজেই কোন অদল বদলের কিছু ঘটে নি!

```
x = 13; // x এর মান আরোপ করা হলো
y = 12; // y এর মান আরোপ করা হলো
```

অদল-বদল বুঝার জন্য চিন্তা করো তোমার দুটি পেয়ালা আছে: কাঁচের পেয়ালা আর কাঁসার পেয়ালা। কাঁচের পেয়ালায় আছে আঙুরের রস আর কাঁসার পেয়ালায় কমলার রস। এখন তুমি এই পেয়ালা দুটোতে থাকা ফলের রস অদল-বদল করতে চাও যাতে কাঁচের পেয়ালায় থাকে কমলার রস আর কাঁসার পেয়ালায় থাকে আঙুরের রস। তো এখন তুমি কী করবে। তুমি তো আর সরাসরি এক-টার ফলের রস আরেকটাতে ঢেলে দিতে পারো না। তোমাকে যেটা করতে হবে তা হলো আরেকটা পেয়ালা নেয়া। ধরো সেটা কাঠের পেয়ালা। এই কাঠের পেয়ালাটি তুমি একটা থেকে আরেকটাতে ঢালাঢালির কাজে ব্যবহার করবে। তাহলে এই অতিরিক্ত কাঠের পেয়ালা কাজে লাগিয়ে কীভাবে তোমার কাঁচ আর কাঁসার পেয়ালার ফলের রস অদল-বদল করা যায়, আমরা নীচে তা দেখি।

১. একদম শুরুতে কাঁচের পেয়ালায় রয়েছে আমাদের আঙুরের রস আর কাঠের পেয়ালা খালি। সুতরাং কাঁচের পেয়ালা থেকে আঙুরের রস কাঠের পেয়ালায় ঢালো। ফলে কাঠের পেয়ালায় থাকলো আঙুরের রস আর কাঁচের পেয়ালা খালি হয়ে গেলো।

২. কাঁচের পেয়ালা যেহেতু এখন খালি আর কাঁসার পেয়ালায় আছে কমলার রস, আমরা তাই কাঁসার পেয়ালার কমলার রস কাঁচের পেয়ালায় ঢালবো। ফলে কাঁচের পেয়ালায় থাকলো কমলার রস আর কাঁসার পেয়ালা খালি হয়ে গেলো।

৩. কাঁসার পেয়ালা যেহেতু এখন খালি আর কাঠের পেয়লায় আছে আঙুরের রস, আমরা তাই কাঠের পেয়ালার আঙুরের রস কাঁসার পেয়ালায় ঢালবো। ফলে কাঁসার পেয়ালায় থাকলো আঙুরের রস আর কাঠের পেয়ালা খালি হয়ে গেলো।

উপরের ধাপ তিনটি সম্পন্ন করলেই আমাদের এক পেয়ালার ফলের রস আরেক পেয়ালায় অদল-বদল হয়ে যাবে। তো পেয়ালা দুটোর রস অদল-বদলের মতোই আসলে আমাদের চলকদু-টোর মান অদল-বদল করতে হবে। একটা অতিরিক্ত পেয়ালার মতো আমাদের এখানেও লাগবে একটা অতিরিক্ত চলক। ধরে নেই আমাদের সেই অতিরিক্ত চলক হলো z। আমরা তাহলে এই অতিরিক্ত চলক কাজে লাগিয়ে আমাদের x আর y চলকের মান অদল-বদল করে ফেলি।

```
z = x;  // z হলো 12 আর x আছে 12, y আছে 13
x = y;  // x হলো 13 আর y আছে 13, z আছে 12
y = z;  // y হলো 12 আর z আছে 12, x আছে 13
```

তো উপরের তিনটি বিবৃতি চালালেই আমাদের x আর y চলক দুটোর মান অদল-বদল হয়ে গেলো। তবে পেয়ালা আর ফলের রসের অদল বদলের সাথে চলক আর মানের অদল-বদলের কিন্তু কিছুটা তফাৎ আছে। তফাৎটা হলো ফলের রস এক পেয়ালা থেকে আরেক পেয়ালয় ঢাললে যেটা থেকে ঢালা হলো সেই পেয়ালা খালি হয়ে যায়। কিন্তু চলকের ক্ষেত্রে z = x; করলে চলক x এর মান চলক z এ আরোপ হয় ঠিকই, কিন্তু চলক x কিছুতেই খালি হয় না, বরং তার যে মান ছিলো সেটাই থাকে। চলকের মান বদলে যায় কেবল যখন এতে নতুন মান আরোপ করা হয়।

৫.৫ আরোপণের বাম ও ডান (Assignment Left and Right)

কোন চলকের (variable) বাম-মান (l-value) ও ডান-মান (r-value) বলতে কী বুঝো? কোন চলকে মান আরোপণ করতে গেলে আমরা আরোপ (assign) = চিহ্ন দিয়ে বামে ও ডানে কিছু লিখি যেমন y = x;। এখানে বামেরটিকে বলা হয় লক্ষ্য (target) আর ডানেরটিকে বলা হয় উৎস (source)। আরোপণের ফলে ডান পাশের উৎস থেকে মান বাম পাশের লক্ষ্যে আরোপিত হয়। কথা হচ্ছে আরোপ = চিহ্নের বামে আমরা কী কী দিতে পারবো বা পারবো না, আর ডানেই বা কী কী দিতে পারবো বা পারবো না? তাছাড়া একটা চলকের নাম আরোপ = চিহ্নের বাম বা ডানপাশে লিখলে এই দুই ক্ষেত্রে চলকের ভুমিকায় আসলে কোন তফাৎ হয় কিনা?

এই আলোচনায় যাওয়ার আগে আমরা একটু পরের উদ্ধৃতাংশটুকু বিবেচনা করি। "ঢাকার মামা হালিম বিখ্যাত। চল আমরা মামা হালিম খাই। তুমি খাবে এক বাটি, আমি খাব এক বাটি। আমার বাটিটা পরিস্কার নয়, তোমার বাটিটা পরিস্কার।" তো এইখানে বাটি মানে কখন আসলে হালিম আর কখন আসলে সেটা পাত্র? আমরা বুঝতে পারি "তুমি খাবে এক বাটি, আমি খাব এক বাটি" এই কথাগুলোতে বাটি বলতে আসলে সত্যি সত্যি পাত্রটাকে কামড়ে কামড়ে খাওয়ার কথা বলা হচ্ছে না, বরং তুমি এক বাটি পরিমান হালিম খাবে আর আমি এক বাটি পরিমান হালিম খাবো তাই বুঝানো হচ্ছে। এক বাটি হালিম মানে একটা বাটিতে থাকা হালিম। বিষয়গুলোকে চলক আর তার মানের সাথে মিলাও। বাটি ঠিক যেন চলকের মতো আর হালিম হল তার মানের মতো। আবার "আমার বাটিটা পরিস্কার নয়, তোমার বাটিটা পরিস্কার।" এই অংশে বাটি মানে আসলে বাটি নামের পাত্রটা, সেই পাত্রে ঢালা হালিম নয় কোন ভাবেই। তাহলে দেখা যাচ্ছে বাটি বলতে কখনো কখনো আসলে পাত্রটাকে বুঝানো হয় আর কখনো কখনো পাত্রটাতে থাকা হালিমকে বুঝানো হয়। একই ভাবে চলকের নাম উল্লেখ করলে কখনো কখনো চলকটির মানকে বুঝানো হয়, কখনো কখনো আসলে চলকটির জন্য স্মরণিতে (memory) বরাদ্দ জায়গাটুকু বুঝানো হয়।

x = 3; এখানে চলক x বলতে আমরা আসলে চলক x এর জন্য স্মরনিতে (memory) নেয়া জায়গাটুকু বুঝি যেখানে মান 3 কে রাখা হবে। এখানে কোন ভাবেই চলক x এ আগে থেকে বিদ্যমান মানকে বুঝানো হচ্ছে না। খেয়াল করো এখানে চলক x আরোপ = চিহ্নের বাম পাশে আছে। যখন চলক x আসলে স্মরনিতে বরাদ্দকৃত জায়গাকে বুঝায় তখন এটাকে আমরা স্রেফ চলক না বলে আরো স্পষ্ট করে বলবো চলকের বাম-মান (l-value)। তাহলে মনে রেখো চলকের বাম মান দিয়ে আমরা বুঝাবো চলকের জন্য স্মরনিতে নেয়া জায়গাটুকু।

y = x; এখানে চলক y বলতে আমরা চলক y এর জন্য স্মরনিতে বরাদ্দ পাওয়া জায়গাটুকুকে বুঝি। আর চলক y আরোপ = চিহ্নের বামে আছে তাই এখানে চলক y এর বাম-মান ব্যবহৃত হয়েছে। তবে চলক x বলতে এখানে আমরা কেবল তার মানটাকে বুঝি। খেয়াল করো চলক x এর মানটাইতো চলক y এর স্মরনির জায়গাটাতে জমা হবে, চলক x এর জন্য বরাদ্দ জায়গাতো আর গিয়ে চলক y এর জায়গায় লেখা হবে না। আমরা দেখছি এখানে চলক x আরোপ = চিহ্নের ডানে রয়েছে। যখন চলক x আসলে তার মানটাকে বুঝায় তখন আমরা এটাকে বলব চলকের ডান-মান (r-value)। চলকের ডান মান দিয়ে আমরা তাহলে বুঝাবো চলকের যে মান সেটিকে, স্মরণিতে থাকা জায়গাটিকে নয়।

উপরের আলোচনা থেকে আমরা একটা বিষয়ই পরিস্কার করতে চেয়েছি সেটা হলো, আরোপ = চিহ্নের বামে আমরা কেবল এমন কিছু দিতে পারবো যার জন্য স্মরনিতে জায়গা দখল করা আছে, অর্থাৎ যার বাম-মান (l-value) আছে। আর আরোপ চিহ্নের ডান পাশে আমরা এমন কিছু দিতে পারবো যার মান আছে অর্থাৎ ডান-মান (r-value) আছে। একটা বিষয় খেয়াল করো যার বাম-মান আছে অর্থাৎ স্মরনিতে যার জায়গা আছে তার একটা মানও থাকবেই অর্থাৎ তার ডান-মান

থাকবেই, যেমন যে কোন চলকের। কথা হচ্ছে এমন কিছু কি আছে যার ডান মান আছে কিন্তু বাম মান নাই। উত্তর ধরে নিতে পারো আছে। যেমন x = 3; এইখানে 3 এর ডান মান আছে কিন্তু বাম মান নাই। কাজেই কেউ চাইলে 3 = x; লিখতে পারবে না, সংকলন (compile) করার সময় ত্রুটি দেখাবে, বলবে "error: lvalue required as left operand of assignment"। একই ভাবে কেউ চাইলে আরোপণ হিসাবে y+3 = x;ও লিখতে পারবে না, একই ত্রুটি (error) দেখাবে, কারণ চলক y এর বাম মান সম্ভব হলেও y + 3 করলে ওইটা আর চলক y থাকে না হয়ে যায় একটা রাশি যার মান হবে y এর মান যোগ 3, কাজেই সেটার কেবল মান থাকে, তার জন্য স্মরনিতে কোন জায়গা থাকে না। বুঝাই যাচ্ছে অন্যদিকে আরোপণ হিসাবে x = y + 3; লিখা যাবে কারণ y + 3 এর ডান-মান আছে অপর দিকে চলক x এর বাম-মান আছে।

৫.৬ আত্ম-শরন আরোপণ (Self-Referential Assignment)

ক্রমলেখ (program) দেখলে আমাদের সাধারণত x = x + 1; বা এই জাতীয় অদ্ভুত কিছু বিষয় নজরে আসে। মুল কথা হলো এই সব ক্ষেত্রে একই চলক (variable) আরোপ (assignment) = চিহ্নের বামেও রয়েছে আবার ডানেও রয়েছে। আমরা সকলে গণিত জানি কম বা বেশী। সেখানে সমীকরণ নিয়ে আমাদের যে ধারনা আছে সেই অনুযায়ী তো x কখনো x + 1 এর সমান হতে পারে না। তাহলে ক্রমলেখতে x = x + 1; এর মতো অর্থহীন বিষয় কেন থাকে?

```
x = x + 1;   // চিহ্ন = গণিতের সমান চিহ্ন নয়, এটি গণনার আরোপণ।
```

আসলে = চিহ্নটি গণিতে আমরা ব্যবহার করি দুটো সংখ্যা তুলনা করে যদি দেখি তারা একে অপরের সমান তাহলে। আমরা তাই ওটাকে গণিতে সমান (equal) চিহ্ন বলে থাকি। কিন্তু গণনার জগতে = চিহ্নটিকে সমান চিহ্ন হিসাবে ব্যবহার না করে বরং আরোপণ (assignment) চিহ্ন হিসাবে ব্যবহার করা হয়। কাজেই কোন ক্রমলেখতে আমরা যখন x = x + 1; দেখি তখন আসলে ওটা কোন ভাবেই গণিতের সমীকরণ নয়, বরং ওইটা গণনার জগতের আরোপণ। সুতরাং গণিতের জগতে ওইটা কোন অর্থ তৈরী না করলেও গণনার জগতে ওটার সুনির্দিষ্ট অর্থ আছে।

আমরা আরোপণ (assignment) নিয়ে আগেই আলোচনা করেছি। ওই আরোপণগুলোর সবগুলোতে বাম আর ডান উভয় পাশে চলক থাকলেও আলাদা আলাদা চলক ছিল। আর x = x + 1ও আরোপণ তবে এখানে একই চলক আরোপ চিহ্নের বামেও আছে ডানেও আছে। এইরকম আরোপণ যেখানে একই চলক বামেও আছে ডানেও আছে সেটাকে আমরা বলবো আত্মশরন আরোপণ (self-referential assignment) অর্থাৎ যেখানে একটা চলক নিজের মানের জন্য নিজেরই শরনাপন্ন হয়। আত্মশরণ আরোপণে ডানপাশে চলকটির ডান-মান (r-value) ব্যবহৃত হয়, আর বামপাশে চলকটির বাম-মান (l-value) ব্যবহৃত হয়। এই রকম আরোপণে আসলে কী ঘটে?

```
int x = 3;        // চলক x এ আদি মান আরোপ করা হলো
x = x + 1;        // এখানে আত্ম–শরণ আরোপণ করা হচ্ছে
cout << x << endl;   // চলক x এর মান ফলন দেওয়া হচ্ছে
```

এই রকম আরোপণ বুঝতে গেলে আমরা x = x + 1; বিবৃতিটিকে দুইটি ঘটনায় বিভক্ত করে নিতে পারি। একটা ঘটনা হল ডান পাশে x + 1 হিসাব করা অর্থাৎ x+1 এর মান বা আরো পরিস্কার করে বললে ডান-মান হিসাব করা। আর অন্য ঘটনাটা হল বাম পাশে x এর বাম-মানে অর্থাৎ স্মরনিতে (memory) x এর জন্য বরাদ্দ করা জায়গায় ডান পাশ থেকে পাওয়া মানটি লিখে দেওয়া। তো এই দুটো ঘটনার প্রথমটি আগে ঘটবে আর দ্বিতীয়টি পরে ঘটবে। উপরে আমরা x এর আদি মান নিয়েছি 3। এরপর যখন x= x+1; নির্বাহিত (execute) হবে তখন প্রথম ঘটনাটি ঘটবে

আগে অর্থাৎ x + 1 মান হিসাব হবে। x এর মান যেহেতু এই অবস্থায় 3 তাই x + 1 হবে 4। মনে করে দেখো এই 4 এর কিন্তু কেবল ডান-মান আছে এর জন্য স্মরনিতে কোন জায়গা দখল করা নেই বা এর কোন বাম-মান নেই। অর্থাৎ এই 4 কোন ভাবেই x চলকের জায়গায় নেই, অন্য কোথাও আছে। যাইহোক এমতাবস্থায় এরপর ঘটবে দ্বিতীয় ঘটনাটি অর্থাৎ এই 4 মানটি গিয়ে লেখা হয়ে যাবে x এর জন্য বরাদ্দ জায়গাতে। আমরা তাই x এর পুরনো মান 3 বদলে সেখানে পাবো এর নতুন মান 4। তাহলে x = x + 1; আত্ন-শরন আরোপণের ফলে চলকের মান এক বেড়ে গেলো।

আত্নশরন আরোপণের আরো নানারকম জটিল অবস্থা আছে যেমন x = x * 3; বা x = x* x + x + 1;। এগুলোর প্রতিটি ক্ষেত্রে আগে ডানপাশের মান হিসাব করা হবে আর তারপর সেই মান বাম পাশে লিখে দেয়া হবে, ফলে চলকটিতে নতুন একট মান থাকবে।

৫.৭ অনুশীলনী সমস্যা (Exercise Problems)

ধারণাগত প্রশ্ন: নীচে কিছু ধারণাগত প্রশ্ন রয়েছে। প্রশ্নগুলোর উত্তর নিজে নিজে বের করবে।

১. সরাসরি ক্রমলেখয়ের (program) ভিতরে উপাত্ত দিয়ে দিলে সমস্যা কী?

২. উপাত্ত (data) কেনো যোগান (input) নিতে হবে? সুবিধা-অসুবিধা কী কী?

৩. যোগান যাচনা (input prompt) কী? যোগান নেয়ার আগে কেন যাচনা করা উচিৎ?

৪. চলকে (variable) মান আরোপণে (assignment) লক্ষ্য ও উৎসে কী ঘটে বর্ণনা করো।

৫. চলকের বাম-মান আর ডান-মান বলতে কী বুঝো? উদাহরণ দিয়ে ব্যাখ্যা করো।

৬. আরোপণে = চিহ্নের বামে কেন এমন কিছু দেয়া যায় না যার কেবল ডান মান আছে?

৭. আত্ন-শরণ (self-referential) আরোপণ কী উদাহরণ সহ ব্যাখ্যা করো।

৮. দুটি চলকে (variable) থাকা মান বদলাবদলি করবে কেমনে ব্যাখ্যা করো।

পরিগণনার সমস্যা: নীচে আমরা কিছু পরিগণনার সমস্যা দেখবো। এই সমস্যাগুলো আগে ধৈর্য্য ধরে নিজে নিজে সমাধান করতে চেষ্টা করবে। যখন একেবারেই পারছো না বলে মনে হয় তখনই কেবল সমাধান দেখে নিতে পারো। সমাধানগুলো পরিগণনার প্রশ্নগুলোর শেষে আছে।

১. এমন একটি ক্রমলেখ (program) রচনা করো যেটি একটি পূর্ণক (int) ও একটি ভগ্নক (float) যোগান (input) নিয়ে সেগুলো আবার ফলনে (output) দেখায়।

২. এমন একটি ক্রমলেখ (program) রচনা করো যেটি দুটি ভগ্নক (float) সংখ্যা যোগান (input) নিয়ে সংখ্যা দুটি ও তাদের যোগফল ফলনে (output) দেখায়।

৩. এমন একটি ক্রমলেখ (program) রচনা করো যেটি তিনটি পূর্ণক (int) যোগান (input) নিয়ে তাদেরকে যে ক্রমে যোগান নেয়া হয়েছে সেই ক্রমে আবার উল্টোক্রমে দেখাবে। যেমন ভুক্ত সংখ্যা তিনটি যদি হয় পর পর 2 3 1 তাহল সিধা ক্রমে দেখাবে 2 3 1 আবার তাদের উল্টাক্রমে দেখাবে 1 3 2। খেয়াল করো আমরা কিন্তু মানের ক্রম বলছি না।

৪. এমন একটি ক্রমলেখ (program) রচনা করো যেটি একদম ঠিক ঠিক নীচের মতো যোগান (input) ও ফলন (output) উৎপন্ন করে। তুমি কিন্তু পরীক্ষার নম্বরগুলো যোগান নিবে, আর আমরা একেকবার চালানোর সময় এক এক রকম সংখ্যা যোগান দিবো।

```
folafol nirnoyer kromolekho
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _
prothom porikkhai koto? 90
ditiyo porikkhai koto? 75
tritiyo porikkhai koto? 91
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _
shorbo mot number holo 256
```

পরিগণনা সমাধান: এবার আমরা পরিগণনার সমস্যাগুলোর সমাধান দেখবো। মনে রাখবে সমাধানগুলো দেয়া হয়েছে তুমি যদি একান্তই নিজে নিজে করতে পারছোনা বলে মনে হয়, তখন কেবল একটু সাহায্য যাতে পেতে পারো তাই। কাজেই পারতপক্ষে সমাধান দেখবে না।

আমরা ধরে নিচ্ছি যে তুমি দরকারী শির নথি (header) অন্তর্ভুক্ত করা, নামাধার (namespace) std ব্যবহার করা, main বিপাতকের কংকাল লেখা আর সেটার শেষে return EXIT_SUCCESS; লিখে মান ফেরত দেয়া ইত্যমধ্যে ভালো করে শিখে ফেলেছো। তো তুমি যদি নীচে লেখা ক্রমলেখগুলো সংকলন (compile) করে চালাতে (run) চাও, তোমাকে কিন্তু আগে include, namespace, main, return এগুলো লিখে নিতে হবে, তারপর main বিপাতকের ভিতরে return এর আগে তুমি আমাদের নীচের অংশগুলো লিখে নিবে। তারপর সংকলন করে ক্রমলেখ চালাবে। আমরা এখন থেকে মোটামুটি এইভাবে ক্রমলেখ দেখাবো।

১. এমন একটি ক্রমলেখ (program) রচনা করো যেটি একটি পূর্ণক (int) ও একটি ভগ্নক (float) যোগান (input) নিয়ে সেগুলো আবার ফলনে (output) দেখায়।

ফিরিস্তি ৫.৩: যোগান ও ফলনের ক্রমলেখ (Input Output Program)

```
int purnok;
float vognok;

cout << "purnok koto? ";
cin >> purnok;

cout << "vognok koto? ";
cin >> vognok;

cout << "puronok holo " << purnok << endl;
cout << "vognok holo " << vognok << endl;
```

২. এমন একটি ক্রমলেখ (program) রচনা করো যেটি দুটি ভগ্নক (float) সংখ্যা যোগান (input) নিয়ে সংখ্যা দুটি ও তাদের যোগফল ফলনে (output) দেখায়।

ফিরিস্তি ৫.৪: যোগান প্রকিয়ন ফলন (Input Process Output)

```
float prothom, ditiyo;

cout << "songkhya duti koto? ";
cin >> prothom >> ditiyo;

float jogfol = prothom + ditiyo;

cout << "songkhya duti "; //  কোন endl নাই
cout << prothom << " " << ditiyo << endl;

cout << "tader jogfol " << jogfol << endl;
```

৩. এমন একটি ক্রমলেখ (program) রচনা করো যেটি তিনটি পূর্ণক (int) যোগান (input) নিয়ে তাদেরকে যে ক্রমে যোগান নেয়া হয়েছে সেই ক্রমে আবার উল্টোক্রমে দেখাবে। যেমন ভুক্ত সংখ্যা তিনটি যদি হয় পর পর 2 3 1 তাহল সিধা ক্রমে দেখাবে 2 3 1 আবার তাদের উল্টাক্রমে দেখাবে 1 3 2। খেয়াল করো আমরা কিন্তু মানের ক্রম বলছি না।

ফিরিস্তি ৫.৫: যোগানের সিধা ক্রম উল্টা ক্রম (Input Order Reverse Order)

```
int prothom, ditiyo, tritiyo;

cout << "songkhya tinti koto? ";
cin >> prothom >> ditiyo >> tritiyo;

cout << "sidha krome " << prothom << " ";
cout << ditiyo << " " << tritiyo << endl;

cout << "ulta krome " << tritiyo << " ";
cout << ditiyo << " " << prothom << endl;
```

৪. এমন একটি ক্রমলেখ (program) রচনা করো যেটি একদম ঠিক ঠিক নীচের মতো যোগান (input) ও ফলন (output) উৎপন্ন করে। তুমি কিন্তু পরীক্ষার নম্বরগুলো যোগান নিবে, আর আমরা একেকবার চালানোর সময় এক এক রকম সংখ্যা যোগান দিবো।

```
folafol nirnoyer kromolekho
- - - - - - - - - - - - - - -
prothom porikkhai koto? 90
ditiyo porikkhai koto? 75
tritiyo porikkhai koto? 91
- - - - - - - - - - - - - - -
shorbo mot number holo 256
```

ফলাফল প্রক্রিয়াকরণের ক্রমলেখটি আমরা নীচে দেখাচ্ছি।

ফিরিস্তি ৫.৬: ফলাফল প্রক্রিয়ার ক্রমলেখ (Result Processing Program)

```
int prothom, ditiyo, tritiyo;

cout << "folafol nirnoyer kromolekho" << endl;

cout << "- - - - - - - - - - - - - - - -" << endl;

cout << "prothom porikkhai koto? ";
cin >> prothom;

cout << "ditiyo porikkhai koto? ";
cin >> ditiyo;

cout << "tritiyo porikkhai koto? ";
cin >> tritiyo;

cout << "- - - - - - - - - - - - - - - -" << endl;

int folafol = prothom + ditiyo + tritiyo;

cout << "shorbo mot number holo ";
cout << folafol << endl;
```

৫.৮ গণনা পরিভাষা (Computing Terminologies)

- সূত্র (formula)
- যাচনা (prompt)
- অদল-বদল (swap)
- বাম-মান (l-value)
- ডান-মান (r-value)
- আত্ম-শরণ (self-reference)

অধ্যায় ৬

গাণিতিক প্রক্রিয়াকরণ (Mathematical Processing)

গাণিতিক প্রক্রিয়াকরণে রাশিতে (expression) গাণিতিক অণুক্রিয়া (operator) ও বিপাতক (function) সমুহ কী ভাবে হিসাব করা হয় আমাদের তা জানতে হবে।

৬.১ একিক অণুক্রিয়া (Unary Operators)

সিপিপিতে একিক (unary) অণুক্রিয়ক ধনাত্নক (positive) + আর ঋণাত্নক (negative) – কী ভাবে কাজ করে? যথাযথ ক্রমলেখ লিখে উদাহরণ সহ বুঝিয়ে দাও। একিক (unary) অণুক্রিয়ক (operator) একটা উপাদানের (operand) ওপর প্রযুক্ত হয়ে ফলাফল উৎপন্ন করে।

ফিরিস্তি ৬.১: পাটিগণিতের ধনাত্নক ও ঋণাত্নক (Arithmetic Positive Negative)

```
int a = 5;     int const b = -9; // a চলক b ধ্রুবক
cout << "+7 = " << +7 << "     -7 = " << -7 << endl;
cout << "+a = " << +a << "     -a = " << -a << endl;
cout << "+b = " << +b << "     -b = " << -b << endl;
cout << endl;
cout << "+(a*b) = " << +(a*b);     //  a*b হল রাশি
cout << "     -(a*b) = " << (a*b) << endl;
cout << "+abs(b) = " << +abs(b);  //abs() বিপাতক
cout << "     -abs(b) = " << -abs(b) << endl;
```

ফলন (output)

```
+7 = 7     -7 = -7
+a = 5     -a = -5
+b = -9     -b = 9

+(a*b) = -45     -(a*b) = -45
+abs(b) = 9     -abs(b) = -9
```

কোন সংখ্যা, চলক (variable), ধ্রুবক (constant), বিপাতক (function), বা রাশির (expression) সামনে ধনাত্মক চিহ্ন থাকলে তার যে মান সেটিই থাকে, কিন্তু ঋণাত্মক চিহ্ন থাকলে তার চিহ্ন বদলে যায় অর্থাৎ আগে ধনাত্মক থাকলে পরে ঋণাত্মক হয়ে যায় আর আগে ঋণাত্মক থাকলে পরে ধনাত্মক হয়ে যায়। চলক ও ধ্রুবক আগেই জানো। বিপাতক (function) হলো এমন একটা জিনিস যে কিছু যোগান (input) নিয়ে কিছু ফলন (output) দেয়। যেমন cstdlib নামক শির নথিতে (header file) abs(x) নামে একটা বিপাতক আছে যেটি একটি সংখ্যা যোগান নিয়ে তার চিহ্নটুকু বাদ দিয়ে কেবল মানটুকু ফলন হিসাবে ফেরত দেয়। অর্থাৎ abs(3) হলো 3 আবার abs(-3)ও 3। একই ভাবে abs(3.5) হলো 3.5 আবার abs(-3.5)ও 3.5। রাশি (expression) হল সংখ্যা, ধ্রুবক, চলক, অণুক্রিয়ক, বিপাতক মিলে যখন একটা জিনিস তৈরী হয় যার মান হিসাব করা যায় যেমন 3 + x $*$ abs(y) একটি রাশি যেখানে x আর y হল চলক।

৬.২ দুয়িক অণুক্রিয়া (Binary Operators)

সিপিপিতে দুয়িক (binary) অণুক্রিয়কগুলো যোগ +, বিয়োগ $-$, গুণ $*$, কী ভাবে কাজ করে? যথাযথ ক্রমলেখ লিখে উদাহরণ সহ বুঝিয়ে দাও। দুয়িক অণুক্রিয়ক (operator) দুটো উপাদানের (operand) ওপর প্রযুক্ত হয়ে ফলাফল উৎপন্ন করে।

ফিরিস্তি ৬.২: পাটিগণিতের যোগ বিয়োগ গুণ (Arithmetic Plus Minus Times)

```
cout << "5 + 3 = " << 5 + 3 << endl;
cout << "5.1 + 3 = " << 5.1 + 3 << endl;
cout << "5.1 + 3.2 = " << 5.1 + 3.2 << endl;
cout << endl;

cout << "5 - 3 = " << 5 - 3 << endl;
cout << "5.1 - 3 = " << 5.1 - 3 << endl;
cout << "5.1 - 3.2 = " << 5.1 - 3.2 << endl;
cout << endl;

cout << "5 * 3 = " << 5 * 3 << endl;
cout << "5.1 * 3 = " << 5.1 * 3 << endl;
cout << "5.1 * 3.2 = " << 5.1 * 3.2 << endl;
cout << endl;
```

উপরের ক্রমলেখ (program) খেয়াল করো। আর তার সাথে নীচের ফলন (output) মিলিয়ে নাও। লক্ষ্য করো আমরা তিনটি করে যোগ, বিয়োগ, আর গুণ করেছি। যোগ, বিয়োগ, বা গুণ আমরা ভালোই জানি, নতুন করে শেখার কিছু নাই। তবে একটি বিষয় খেয়াল করতে হবে। সেটি হলো উপাত্তের ধরণ কেমন? আর এ কারণেই আমরা প্রতিটি অণুক্রিয়ার (operator) জন্যে তিনটি করে উদাহরণ নিয়েছি। প্রতিটি অণুক্রিয়ার উদাহরণগুলোর প্রথম সারিতে খেয়াল করো। সেখানে উপাদান (operand) হিসাবে আমরা দুটো পূর্ণকের যোগ, বিয়োগ বা গুণ করেছি, যেমন 5 + 3, 5 $-$ 3 আর 5 $*$ 3। ফলাফল হিসাবে যা পেয়েছি তাও একটি পূর্ণক, যেমন 8, 2, আর 15। এবার প্রতিটি অণুক্রিয়ার জন্য তৃতীয় সারিতে খেয়াল করো। সেখানে উপাদান (operand) হিসাবে আমরা দুটো ভগ্নক যোগ, বিয়োগ বা গুণ করেছি, যেমন 5.1 + 3.2, 5.1 $-$ 3.2 আর 5.1 $*$ 3.2। ফলাফল হিসাবেও আমরা পেয়েছি একটি ভগ্নক যেমন 8.3, 1.9, আর 16.32। তারপর

প্রতিটি অণুক্রিয়ার জন্য দ্বিতীয় সারিতে খেয়াল করো। উপাদান হিসাবে একটি ভগ্নক ও একটি পূর্ণক যোগ, বিয়োগ বা গুণ করা হয়েছে যেমন 5.1 + 3, 5.1 − 3, আর 5.1 * 3। আর ফলাফল এসেছে একটি ভগ্নক যেমন 8.1, 2.1, আর 15.3, যেগুলোর কোনটিই পূর্ণক নয়। উপাদানদুটো একটা ভগ্নক হওয়ায় ফলাফলও ভগ্নক হয়ে গেছে।

ফলন (output)

```
5 + 3 = 8
5.1 + 3 = 8.1
5.1 + 3.2 = 8.3

5 − 3 = 2
5.1 − 3 = 2.1
5.1 − 3.2 = 1.9

5 * 3 = 15
5.1 * 3 = 15.3
5.1 * 3.2 = 16.32
```

তাহলে উপরের আলোচনা থেকে আমরা দেখলাম কোন অণুক্রিয়ার (operator) যদি দুটি উপাদানই (operand) একরকম হয় তাহলে ফলাফলও সেই রকমই হয়। যেমন উপাদান দুটোই int হলে ফলাফলও int; উপাদান দুটোই float হলে ফলাফলও float। আর যদি দুটো উপাদান দুরকম হয় যেমন একটি পূর্ণক বা int আর একটি ভগ্নক বা float তাহলে ফলাফল হবে ভগ্নক বা float। গণিতে আমরা জানি পূর্ণক সংখ্যাগুলো একই সাথে ভগ্নকও যেমন 3 আসলে 3.0, কিন্তু একটি ভগ্নক কিন্তু পূর্ণক নাও হতে পারে যেমন 5.1 ভগ্নক কিন্তু একে পূর্ণক হিসাবে লেখা সম্ভব নয়। আর এ কারণে কোন অণুক্রিয়া (operator) প্রয়োগের পূর্বে উপাদান (operand) দুটো দুরকম হলে প্রথমে পূর্ণকটিকে ভিতরে ভিতরে ভগ্নকে রূপান্তর করে নেয়া হয়, আর তারপর যোগ, বিয়োগ বা গুণ করা হয় দুটোকে ভগ্নক হিসাবে নিয়েই। এই যে ভিতরে ভিতরে পূর্ণকটি ভগ্নকে রূপান্তর করা হয় এটা এক রকমের উপাত্ত প্রকারান্তর (type casting)। উপাত্ত প্রকারান্তর নিয়ে আমরা পরে আরো বিস্তারিত জানবো, আপাতত পূর্ণক থেকে ভগ্নকে প্রকারান্তর মনে রাখো।

৬.৩ ভাগফল ও ভাগশেষ (Division and Remainder)

সিপিপিতে দুয়িক (binary) অণুক্রিয়ক ভাগফল (division) ও ভাগশেষ (remainder) কী ভাবে কাজ করে? যথাযথ ক্রমলেখ লিখে উদাহরণ সহ বুঝিয়ে দাও। তুমি ইত্যমধ্যে জেনেছো দুয়িক অণুক্রিয়ক (operator) দুটো উপাদানের (operand) ওপর প্রযুক্ত হয়ে ফলাফল উৎপন্ন করে।

ফিরিস্তি ৬.৩: পাটিগণিতের ভাগফল অণুক্রিয়া (Arithmetic Division Operation)

```
cout << "13 / 5 = " << 13 / 5 << endl;
cout << "13.0 / 5 = " << 13.0 / 5 << endl;
cout << "13 / 5.0 = " << 13 / 5.0 << endl;
cout << "13.0 / 5.0 = " << 13.0 / 5.0 << endl;
```

ভাগফলের উপাত্ত প্রকরণ (data type) কেমন হবে সেই নিয়ম আসলে যোগ, বিয়োগ, বা গুণের মতো একই। যদি দুটো উপাদানই (operand) এক রকমের হয় তাহলে ফলাফলও হবে

সেই রকমেরই। কিন্তু উপাদান দুটোর একটি যদি হয় পূর্ণক বা int আরেকটি ভগ্নক বা float তাহলে ফলাফল হবে একটি ভগ্নক বা float। এখানেও ভিতরে ভিতরে int প্রথমে float এ প্রকারান্তর (type casting) হয়ে যাবে, ভাগের কাজটি হবে উপাত্ত প্রকারান্তর হবার পরে। উপাত্ত প্রকারান্তর ছাড়াও ভাগের ক্ষেত্রে ভাগশেষ থাকবে কি থাকবে না সেটার একটা ব্যাপার আছে।

ফলন (output)

```
13 / 5 = 2
13.0 / 5 = 2.6
13 / 5.0 = 2.6
13.0 / 5.0 = 2.6
```

উপরের ফলন লক্ষ্য করো, যদি ভাগের উপাদান দুটোর যেকোন একটিও ভগ্নক হয়, যেমন শেষের তিন সারি, তাহলে কিন্তু ভাগশেষের কোন ব্যাপার থাকে না, ফলে আমরা সেক্ষেত্রে ভাগফল পাই 2.6। কিন্তু ভাগের ক্ষেত্রে যদি দুটো উপাদানই পূর্ণক হয়, যেমন প্রথম সারি তাহলে ভাগটি কিন্তু একটু আলাদা। যেমন 13 / 5 করলে আমরা ফলাফল পাই 2 কারণ আমরা জানি এক্ষেত্রে 3 অবশিষ্ট থাকে। ভাগের ক্ষেত্রে আরো একটি গুরুত্বপূর্ণ বিষয় আছে তা হলো উপাদানের পূর্ণকগুলো ধনাত্মক না ঋণাত্মক। কারণ ঋণাত্মক সংখ্যার ভাগ একটু বিটকেলে হতে পারে। সব মিলিয়ে পূর্ণ সংখ্যার ভাগ আরো বিস্তারিত করে আমরা ভাগশেষের সাথে মিলিয়ে নীচে আলোচনা করবো। তবে একটা কথা মনে রাখবে ভাগের ক্ষেত্রে যদি ভাজক শুন্য হয় যেমন 13 / 0 তাহলে তোমার ক্রমলেখ চালানোর (run) সময় divide by zero বা শুন্য দিয়ে ভাগ নামে ত্রুটিবার্তা (error message) দেখিয়ে বন্ধ হয়ে যাবে। এই রকম ত্রুটি সংকলনের (compile) সময় ধরা পড়ে না, কেবল চালানোর (run) সময় বা নির্বাহ (execute) করার সময় ধরা পড়ে, তাই এদেরকে বলা হয় চলা-কালীন (run-time) বা নির্বাহ-কালীন (execution-time) ত্রুটি।

ফিরিস্তি ৬.৪: পাটিগণিতের ভাগশেষ অণুক্রিয়া (Arithmetic Remainder Operation)

```
cout << "13 / 5 = " << 13 / 5 << "    ";
cout << "13 % 5 = " << 13 % 5 << endl;

cout << "13 / -5 = " << 13 / -5 << "    ";
cout << "13 % -5 = " << 13 % -5 << endl;

cout << "-13 / 5 = " << -13 / 5 << "    ";
cout << "-13 % 5 = " << -13 % 5 << endl;

cout << "-13 / -5 = " << -13 / -5 << "    ";
cout << "-13 % -5 = " << -13 % -5 << endl;

// নীচের সারিগুলো সংকলন (compile) হবে না, ভগ্নকে ভাগশেষ হয় না
// cout << "13.0 % 5 = " << 13.0 % 5 << endl;
// cout << "13.0 % 5.0 = " << 13.0 % 5.0 << endl;
// cout << "13.0 / 5.0 = " << 13.0 / 5.0 << endl;
```

যাইহোক সবশেষে আমরা ভাগশেষ দেখি। ভাগের ক্ষেত্রে আমরা আলোচনা করেছি ভগ্নক বা float এর জন্য ভাগশেষের কোন ব্যাপার নেই। কাজেই ভাগশেষ অণুক্রিয়ার (operator) উপাদান (operand) দুটোর যে কোন একটিও যদি ভগ্নক হয়, তাহলে ভাগশেষ মোটামুটি অর্থহীন

হয়ে যায়। কাজেই এমন কিছু আমাদের ক্রমলেখতে (program) লিখলে সংকলন (compile) করার সময় ত্রুটি (error) আসবে। নীচের ক্রমলেখয়ের শেষের তিনটি সারি দেখতে পারো যেগুলো টীকা হিসাবে রাখা আছে। ওইগুলো টীকা না করে সামনের // হেলানো দাগ দুটো তুলে দিলে ক্রমলেখয়ের অংশ হয়ে যাবে, আর তখন সংকলন করলে ত্রুটি আসবে, করে দেখতে পারো।

একটা বিষয় খেয়াল করেছো, এখানে আমরা কিন্তু টীকার (comment) হেলানো // চিহ্ন দুটোর একরকমের অপব্যবহার করেছি। উপরের ক্রমলেখয়ের শেষ তিনটি সারি আসলে কোন ভাবেই প্রকৃত টীকা নয়। ওগুলোতো বাংলায় বা ইংরেজীতে লেখা নয়, ওগুলো সিপিপিতে লেখা আর টীকা চিহ্ন তুলে নিলেই ওগুলো ক্রমলেখয়ের অংশ হয়ে যাবে সহজেই। তবু কেন এখানে আমরা ওগুলোকে টীকার ভিতরে রাখলাম? এটা আসলে একটা খুবই উপকারী কৌশল। টীকার ভিতরে রাখলে যেহেতু সেটা ক্রমলেখয়ের ঠিক অংশ থাকে না, সংকলন হয় না, কোন ত্রুটি আসার ব্যাপার নাই, আমরা তাই মাঝে মাঝে কিছু কিছু সিপিপিতে লেখা অংশও টীকার ভিতরে রাখি। ক্রমলেখ (program) লেখার সময় আমরা নানান কিছু পরীক্ষা নিরীক্ষা করি, এভাবে করি, ওভাবে করি। তখন যে অংশগুলো ওই সময় দরকার নাই, চাইলে সেগুলো তো মুছে ফেলা যায়, কিন্তু মুছে ফেললেই তো তোমাকে পরে আবার কষ্ট করে লিখতে হতে পারে। এমতাবস্থায় তুমি যদি ওই অদরকারী অংশটুকুতে টীকা দিয়ে (commenting) দাও, ব্যস হয়ে গেলো। কোন ঝামেলা নাই, পরে ওই অংশটুকু আবার দরকার হলেই টীকা তুলে (uncomment) নিবে। কী চমৎকার কৌশল তাই না! আমরা সবাই এটি হরদম ব্যবহার করি। এখন থেকে এই কৌশল কাজে লাগাবে, কেমন!

ফলন (output)

```
13  /  5  =  2      13  %  5  =  3
13  / −5  = −2      13  % −5  =  3
−13 /  5  = −2      −13 %  5  = −3
−13 / −5  =  2      −13 % −5  = −3
```

এবারে ভাগশেষের ফলাফলের দিকে নজর দেই। ভাগফল সহ আলোচনার সুবিধার জন্য উপরের ক্রমলেখ (program) আর ফলন (output) আমরা ভাগশেষের সাথে সাথে ভাগফলও দেখিয়েছি। আমরা আগেই আলোচনা করেছি ভাগশেষ করা যায় কেবল পূর্ণকের জন্য। ভাগ করলে যা অবশেষ থাকে তাই ভাগশেষ। কিন্তু পূর্ণক তো ধনাত্মকও (positive) হতে পারে, ঋণাত্মকও (negative) হতে পারে। আসলে ঋণাত্মক সংখ্যার ভাগশেষ নিয়েই যতো জটিলতা সৃষ্টি হয়। ঋণাত্মক সংখ্যার ভাগশেষ নিয়ে নানান রকম নিয়ম আছে, আমরা এখানে আলোচনা করছি cpp.sh এ যে নিয়মে ভাগশেষ হয়, সেটা নিয়ে। তুমি যে সংকলক (compiler) দিয়ে ক্রমলেখ সংকলন (compile) করবে, জেনে নিও সেখানে কেমন হয়। কারো কাছে থেকে জেনে নিতে পারো। অথবা নিজেই উপরের ক্রমলেখ (program) এর মতো করে ক্রমলেখ তৈরী করে চালিয়ে দেখে নিতে পারো। তেমন কঠিন কিছু নয়।

যাইহোক উপরের ফলন খেয়াল করো। সেখানে কিন্তু কোন ভগ্নক নেই, সবগুলোই পূর্ণক, তবে ধনাত্মক ও ঋণাত্মক আছে। খেয়াল করো ভাগফল ও ভাগশেষ উভয় ক্ষেত্রে মানটা ঠিক পাওয়া যায় চিহ্ন বিবেচনা না করলে। যেমন চারটা ব্যাপারের সবগুলোতেই চিহ্ন বাদ দিলে ভাজক (divisor) আর ভাজ্য (dividend) হয় কেবল 5 আর 13। 13 কে 5 দিয়ে ভাগ করলে ভাগফল হয় 2 আর ভাগশেষ হয় 3। এই পর্যন্ত সবগুলো ব্যাপারেই ঠিক আছে, কিন্তু গোলমাল বাঁধে কেবল চিহ্ন নিয়ে, ভাগফল বা ভাগশেষ কখন ধনাত্মক + হবে আর কখন ঋণাত্মক $-$ হবে। ভাগফলের ক্ষেত্রে খেয়াল করো যখনই সংখ্যা দুটোর চিহ্ন একই রকম তখন ভাগফল ধনাত্মক যেমন প্রথম ও চতুর্থ সারি, আর যখনই তারা বিপরীত চিহ্নের তখনই ভাগফল ঋণাত্মক যেমন দ্বিতীয় ও তৃতীয় সারি। ভাগশেষের ক্ষেত্রে চিহ্ন নির্ভর করে ভাজ্য (dividend) এর ওপর, ভাজকের ওপর নয়। ভাজ্য যখনই ধনাত্মক যেমন 13, ভাগশেষ তখন ধনাত্মক + হয়েছে। আর ভাজ্য যখন ঋণাত্মক যেমন -13 তখন

ভাগশেষ ঋণাত্মক $-$ হয়েছে। ভাগশেষের চিহ্ন 5 বা -5 এর চিহ্নের ওপর নির্ভর করে নাই। একটা বিষয় আগেই বলেছি, ভাগফল ও ভাগশেষের ক্ষেত্রে ভাজক যদি শুন্য হয় তাহলে তোমার ক্রমলেখ চালানোর সময় divide by zero বা শুন্য দিয়ে ভাগ নামে ত্রুটিবার্তা দেখিয়ে বন্ধ হয়ে যাবে। এই রকম ত্রুটি সংকলনের (compile) সময় ধরা পড়ে না, কেবল চালানোর (run) সময় ধরা পড়ে, তাই এদেরকে বলা হয় চলা-কালীন ত্রুটি (run-time error)।

উপরের উদাহরণগুলোতে আমরা যদিও কেবল সংখ্যাই সরাসরি ব্যবহার করেছি, তুমি কিন্তু চাইলে কোন চলক (variable) বা ধ্রুবক (constant) ব্যবহার করতে পারতে। তুমি চাইলে কোন রাশি (expression) বা বিপাতক (function) ও ব্যবহার করতে পারতে। আসলে ডান-মান (r-value) আছে এরকম যে কোন কিছুই এখানে ব্যবহার করা যেতে পারে। এই আলোচনাগুলো একিক অণুক্রিয়ার সময়ই আলোচনা করা হয়েছে, তবুও আবার বলি। বিপাতক (function) এমন একটা জিনিস যে কিছু যোগান (input) নিয়ে কিছু ফলন (output) দেয়। যেমন cstdlib নামক শির নথিতে (header file) abs(x) নামে একটা বিপাতক আছে যেটি একটি সংখ্যা যোগান নিয়ে তার চিহ্নটুকু বাদ দিয়ে কেবল মানটুকু ফলন হিসাবে ফেরত দেয়। অর্থাৎ abs(3) হলো 3 আবার abs(-3)ও 3। একই ভাবে abs(3.5) হলো 3.5 আবার abs(-3.5)ও 3.5। রাশি (expression) হল সংখ্যা, ধ্রুবক, চলক, অণুক্রিয়ক, বিপাতক মিলে যখন একটা কিছু তৈরী করা হয় যার মান আছে সেটি, যেমন 3 + x $*$ abs(y) একটি রাশি যেখানে x আর y হল চলক।

```
int a = 4, b = -3;
int const c = 5;

a + 3, c / b, b * c;      //চলক, ধ্রুবক, সংখ্যা
a = c % abs(b);           // abs(b) হল বিপাতক
a = a - ( b * c);         // b * c হল রাশি
```

৬.৪ আরোপণ অণুক্রিয়া (Assignment Operator)

আরোপণে (assignment) চলকের জন্য স্মরণিতে (memory) বরাদ্দকৃত স্থানে মান ভরে দেয়ার ব্যাপারটা আমরা আগে দেখেছি। কিন্তু আরোপণ আসলে একটা অণুক্রিয়াও (operator) বটে। আরোপণ একটা অণুক্রিয়া এই কথার মানে কী? আমরা আরোপণ নিয়া কী কী করতে পারবো?

আরোপণ (assignment) একটা অণুক্রিয়া (operator) এই কথার মানে হলো আরোপণ কিছু উপাদানের (operand) ওপর প্রযুক্ত হয়ে একটি ফলাফল উৎপন্ন করে। সত্যি বলতে গেলে যোগ, বিয়োগ, গুণ বা ভাগের মতো আরোপণও আসলে একটা দুয়িক (binary) অণুক্রিয়া। কাজেই এটি দুটি উপাদানের (operand) ওপর প্রযুক্ত হয়। আরোপণের বাম পাশে একটা উপাদান থাকে যার বাম-মান থাকতে হবে অর্থাৎ যার জন্য স্মরণিতে (memory) জায়গা বরাদ্দ থাকতে হবে, যেমন চলক। আর আরোপণের ডানে থাকতে হবে এমন কিছু যার ডান-মান বা মান আছে, যেমন চলক (variable), ধ্রুবক (constant), বিপাতক (function) বা রাশি (expression)। কথা হচ্ছে আরোপণের ফলে উৎপন্ন হওয়া ফলাফলটা কী? আসলে যে মানটি আরোপণের বামপাশের চলকে আরোপিত হয় সেই মানটিই আরোপণ অণুক্রিয়ার ফলাফল হিসাবেও বিবেচনা করা হয়।

```
int v = 3, w = -5, x, y, z; //  ভগ্নকও নেয়া যেতে পারে
x = v + 5;     //  চলক x এর মান 8, আরোপণের ফলাফলও 8
y = abs(w);    //  চলক y এর মান 5, আরোপণের ফলাফলও 5
z = x + y;     //  চলক z এর মান 13, আরোপণের ফলাফলও 13
```

উপরে ক্রমলেখতে v + 5 বা 3 + 5 অর্থাৎ 8 আরোপিত হয়েছে x এ। তারপর, abs(w) বিপাতক w বা −5 এর মান হতে চিহ্ন ছাড়া 5 ফেরত দিয়েছে যা আরোপিত হয়েছে y চলকে। আর শেষে x + y বা 8 + 5 অর্থাৎ 13 আরোপিত হয়েছে z চলকে।

তাহলে অন্যান্য অণুক্রিয়ার মতো আরোপণ অণুক্রিয়ারও যেহেতু একটি ফলাফল আছে কাজেই সেই ফলাফলটি অন্য কোন চলক যার বাম-মান আছে তাতে আবারও আরোপন করা সম্ভব!

```
int v = 3, w = -5, x, y, z; //  ভগ্নকও নেয়া যেতে পারে
x = (v + w);// যোগ অণুক্রিয়ার ফলাফল একটি চলকে আরোপণ
z = (y = x);// ডানের আরোপণের ফলাফল বামেরটিতে আরোপণ
z = v * w;  // গুণ আগে হবে, গুণফল আরোপণ তারপরে হবে
z = y = x;  // ডানের আরোপন আগে, সেই ফল নিয়ে বামের আরোপন
```

সুতরাং কেউ যেমন অনেকগুলো যোগ পরপর লিখতে পারে x + y + z + 3, ঠিক তেমনি চাইলেই কেউ অনেকগুলো আরোপণও পরপর লিখতে পারে যেমন z = y = x = w। তবে কোন বন্ধনী নাই ধরে নিলে, যোগের ক্ষেত্রে সাধারণত সবচেয়ে বামের যোগটি থেকে শুরু হয়ে যোগগুলো পরপর বাম থেকে ডানে একে একে হতে থাকে। আর আরোপণের (assignment) ক্ষেত্রে সবচেয়ে ডানের আরোপণ হতে শুরু করে আরোপণগুলো ডান থেকে বামে একে একে হতে থাকে।

```
int x = 1, y = 2; z = 3;  // আদি মান আরোপণ

x + (y = 3);      // y হলো 3, ফলাফল 1 + 3 বা 4
y = x + (z = 4);  // z হলো 4, y হলো 1 + 4 বা 5
z = 5 + (y = z -3); // y হলো 4 - 3 বা 1, z হলো 5 + 1
```

উপরের উদাহরণের শেষ তিনটি সারি খেয়াল করো। চলক ঘোষনার পরের সারির বিবৃতিতে (statement) x + (y = 3); প্রথমে বন্ধনীর ভিতরে y এর মান 3 আরোপণ (assign) হবে আর আরোপণের (assignment) ফলাফলও হবে 3, যা x এর মান 1 সাথে যোগ হয়ে যোগফল হবে 4। এই 4 হলো পুরো রাশিটির মান। এরপরের বিবৃতিতে y = x + (z = 4); প্রথমে বন্ধনীর ভিতরে z এর মান আরোপ হবে 4 আর ফলাফল ও 4, আর তারপর 4 ও x এর মান 1 এর সাথে যোগ হয়ে হবে 5 যা গিয়ে y চলকে আরোপিত হবে। এবারে আসি শেষ বিবৃতিতে z = 5 + (y = z −3); প্রথমে বন্ধনীর ভিতরে z − 3 হিসাব হবে, z এর মান ঠিক আগের সারিতে হয়েছে 4 সাথে 3 বিয়োগ হলে হয় 1 যা y এ আরোপিত হবে আর আরোপণের ফলাফলও (result) হবে 1। এরপর সেই 1 আর 5 যোগ হয়ে ফল হবে 6 যা z এর ভিতরে আরোপিত হবে।

৬.৫ যৌগিক আরোপণ (Compound Assignment)

যৌগিক আরোপণ (compound assignment) কী? সিপিপিতে যৌগিক আরোপণ কী ভাবে আরোপণের সাথে অন্য একটি অণুক্রিয়ার (operator) যোজন (composition) ঘটায়? আত্ম-শরণ (self referential) আরোপণের সাথে যৌগিক আরোপণের সম্পর্ক কী?

যৌগিক আরোপন হলো আরোপনের সাথে আর একটি অণুক্রিয়ার যোজন (composition)। আরোপন = এর সাথে যোগ + এর যোজন ঘটানোর ফলে নতুন যে অণুক্রিয়ক তৈরী হয় সেটি যোগ-আরোপণ +=। একই ভাবে আরোপন = ও বিয়োগ − যুক্ত হয়ে তৈরী হয় বিয়োগ-আরোপণ −=, তারপর একই ভাবে গুণ-আরোপণ *=, ভাগফল-আরোপণ /= আর ভাগশেষ-আরোপণ %=।

```
x += 13;        // এর মানে আসলে x = x + 13;
x -= 7;         //  এর মানে আসলে x = x - 7;
y *= x;         // এর মানে আসলে y = y * x;
z /= x + y;     // এর মানে আসলে z = z / (x + y);
z %= abs(3);    // এর মানে আসলে z = z % abs(3);
```

তাহলে উপরের উদাহরণগুলো থেকে দেখা যাচ্ছে প্রতিটি যৌগিক আরোপণ আসলে এক এক-টি আত্ন-শরণ আরোপন (self-referential assignment)। যৌগিক আরোপণের বাম পাশে যে চলকটি থাকে সেটির মানের সাথে সংশ্লিষ্ট পাটিগণিতীয় অণুক্রিয়া যেমন যোগ, বিয়োগ, গুণ, ভাগফল, বা ভাগশেষ হিসাব করা হয়, আর তারপর ফলাফলটি ওই চলকটিতেই আরোপ করা হয়। আসলে যৌগিক আরোপণগুলো ক্রমলেখ রচনার সময় কষ্ট স্রেফ কিঞ্চিত কমানোর জন্য তৈরী করা হয়েছে। অনেক সময় আরোপনের বাম পাশে যেটি থাকবে সেটি সহজ সরল চলক না হয়ে অন্য কিছু হতে পারে যেটি হয়তো খুবই বড়, সেটির অবশ্যই বাম-মান (l-value) আছে অর্থাৎ তার জন্য স্মর-ণিতে (memory) জায়গা দখল করা আছে। যেমন ধরো নীচের উদাহরণে আমরা সাজন (array) ব্যবহার করছি, শ্রেণী (class) ব্যবহার করছি, এগুলো কী এখনই তা জানতে চেয়ো না, আমরা পরে বিস্তারিত করে শিখবো ওগুলো। খালি খেয়াল করো প্রথম দু সারিতে কী ভাবে লম্বা একটা জিনিস আরোপ = চিহ্নের বাম ও ডান উভয় পাশেই আছে। আর খেয়াল করো শেষের সারির বিবৃতিটি: যৌগিক আরোপণ ব্যবহার করে ওই একই বিষয় কত চমৎকার করে সংক্ষেপে লেখা গেছে।

```
this->amarSajonCholok[suchok] =
    this->amarSajonCholok[suchok] + amarbriddhi;

this->amarSajonCholok[suchok] += amarbriddhi;
```

তাহলে দেখলে তো একই জিনিস আরোপ = চিহ্নের বাম পাশে একবার আবার পরক্ষণেই আরোপ = চিহ্নের ডানপাশেও একবার লিখতে হবে, এটি বেশ বিরক্তিকর, আর দেখতেও কত বিরক্তিকর লাগে। তারচেয়ে যৌগিক আরোপণ সংক্ষিপ্ত আর বুঝাটাও সহজ। ফলাফলের হিসাবে উভয় ক্ষেত্রে কিন্তু আমরা একই ফলাফল পাবো। তবে মনে রেখো ক্রমলেখ (program) চালাতে সময় কম লাগবে নাকি বেশী লাগবে সেইক্ষেত্রে কিন্তু যৌগিক আরোপণের কোন ভুমিকা নেই।

৬.৬ হ্রাস ও বৃদ্ধি অণুক্রিয়া (Increment and Decrement)

সিপিপিতে লেখা ক্রমলেখতে (program) আমরা ++ বা -- প্রায়ই দেখতে পাই। এইগুলো কী? একটা যোগ বা বিয়োগ চিহ্ন দেখেছি কিন্তু দুটো যোগ বা বিয়োগ একসাথে তো আজব ব্যাপার! দুটো যোগ বা বিয়োগ এক সাথে দেয়ার সুবিধা-অসুবিধা কী? ক্রমলেখ কি এতে দ্রুত চলে?

```
int x = 6, y; //  দুটো চলক একটার আদিমান আছে, আরেকটার নাই
++x;        // এক বেড়ে x হলো 7, y জানিনা কারণ আদিমান নেই
x++;        // এক বেড়ে x হলো 8, y জানিনা কারণ আদিমান নেই
y = ++x;    // এক বেড়ে x হলো 9, তারপর y এ 9 আরোপিত হলো
y = x++;    // প্রথমে y হলো x এর সমান বা 9, পরে x হলো 10
```

উপরের ক্রমলেখ (program) খেয়াল করো। দুটো চলক (variable) নেয়া হয়েছে x আর y। চলক x এর আদিমান (initial value) দেয়া হয়েছে 6, কিন্তু y এর আদি মান দেয়া হয় নি। এরপর

দ্বিতীয় আর তৃতীয় বিবৃতিতে রয়েছে ++x; আর x++;, খেয়াল করো উভয় ক্ষেত্রে x এর মান এক করে বেড়েছে, এ কারণে অবশ্য ++ কে বলা হয় বৃদ্ধি অণুক্রিয়ক (increment operator)। বৃদ্ধি অণুক্রিয়ক ++ চলকের আগেই দেয়া হউক আর পরেই দেয়া হউক ফলাফল কিন্তু একই। অবশ্য বৃদ্ধি ++ আগে ব্যবহার করলে এটিকে পূর্ব-বৃদ্ধি (pre-increment) আর পরে ব্যবহার করলে এটিকে উত্তর-বৃদ্ধি (post-increment) বলা হয়।

তবে বলে রাখি বৃদ্ধি অণুক্রিয়কের (increment operator) সাথে কিন্তু এমন কিছু ব্যবহার করতে হবে যার বাম-মান (l-value) রয়েছে অর্থাৎ স্মরণিতে (memory) জায়গা দখল করা আছে। চলকের (variable) যেহেতু বাম-মান আছে তাই আমরা চলক x ব্যবহার করতে পারলাম। কিন্তু তুমি যদি চাও ++3 বা 3++ লিখবে যাতে 4 পাওয়া যায় অথবা লিখবে (x+3)++ বা ++(x+3), তা লিখতে পারবে না, সংকলন (compile) ত্রুটি হবে। ত্রুটি হওয়ার কারণ 3 সংখ্যা (number) বা x+3 রাশির (expression) ডান-মান (r-value) তথা মান (value) আছে কিন্তু তাদের বাম-মান (l-value) তথা স্মরণিতে (memory) জায়গা দখল করা নেই। দরকার নেই তবুও বলে রাখি, তুমি কিন্তু ++ এর সাথে চলক x এর বদলে ধ্রুবক (constant) জাতীয় কিছু তো এমনিতেই ব্যবহার করতে পারবে না, কারণ ধ্রুবকের তো মান বদলানো যায় না।

যাইহোক ++ আগেই দেই আর পরেই দেই ++x বা x++ আসলে x+=1; অর্থাৎ x = x+1; এর সমতুল এবং সংক্ষিপ্ত রূপ বলতে পারো। লক্ষ্য করো বৃদ্ধিতে ++ যে 1 বৃদ্ধি ঘটে সেই ব্যাপারটা কিন্তু উহ্য থাকে। ফলে ++ কেবল একটা উপাদানের (operand) ওপর প্রযুক্ত হয় বলে মনে হয়। আর তাই ++ কে একটি একিক (unary) অণুক্রিয়ক (operator) বলা হয়। কথা হচ্ছে এই একিক অণুক্রিয়ার ফলাফলটা কী? ফলাফল তো আমরা আগেই দেখেছি, মান এক বেড়ে যাওয়া। সেটা ঠিক, কিন্তু তাছাড়াও বৃদ্ধি অণুক্রিয়ার (increment operator) ফলাফলে কিছু গুরুত্বপূর্ণ বিষয় আছে যে কারণে পূর্ব-বৃদ্ধি (pre-increment) আর উত্তর-বৃদ্ধি (post-increment) আলাদা।

পূর্ব-বৃদ্ধি (pre-increment) আর উত্তর বৃদ্ধি (post-increment) যে আলাদা তা পরিস্কার হবে উপরের ক্রমলেখয়ের (program) শেষের সারি দুটো দেখলে। যখন y = ++x; করা হয়েছে তখন x এর মান আগে বেড়ে হয়েছে 9 আর তারপর xএর সেই বেড়ে যাওয়া মান 9ই y এ আরোপিত (assign) হয়েছে। কিন্তু যখন y = x++; তখন কিন্তু খেয়াল করো আগে x এর মান y এ আরোপিত হয়েছে ফলে y হয়েছে 9 আর তারপর x এর মান বেড়েছে 1 ফলে হয়েছে 10। আচ্ছা y = ++x; আর y = x++; এ দুটোকে যদি আমরা বৃদ্ধি ++ ব্যবহার না করে লিখতাম তাহলে কেমন হতো? আমাদের অবশ্যই দুটো করে বিবৃতি লিখতে হতো। নীচে লক্ষ্য করো y = ++x; এ আগে মান বাড়ানো পরে আরোপণ, আর y = x++; এ আগে আরোপণ পরে মান বাড়ানো। আশা করা যায় পূর্ব-বৃদ্ধি (pre-) ও উত্তর-বৃদ্ধির (post-increment) তফাৎ পরিস্কার হয়েছে।

```
x = x + 1;   //  y= ++x; এ x এর মান বৃদ্ধি আগে ঘটবে
y = x;       //  y= ++x; এ y তে x এর মান আরোপন পরে

y = x;       //  y= x++; এ y তে x এর মান আরোপন আগে
x = x + 1;   //  y= x++; এ x এর মান বৃদ্ধি তার পরে
```

পূর্ব-বৃদ্ধি (pre-increment) আর উত্তর-বৃদ্ধির (post-increment) আরো একটা পার্থক্যও জানা দরকার অবশ্য। সেটা হলো পূর্ব-বৃদ্ধির ফলাফল আসলে একটা বাম-মান (l-value) এক্ষেত্রে চলকটির বাম-মান, অন্যদিকে উত্তর-বৃদ্ধির ফলাফল আসলে একটা ডান-মান (r-value)। আগেই বলেছি বৃদ্ধি অণুক্রিয়ার সাথে ব্যবহৃত উপাদানের (operand) অবশ্যই বাম-মান থাকতে হবে। ফলে উত্তর-বৃদ্ধির ফলাফলের ওপরে আবার কোন বৃদ্ধিই চালানো যায় না, কিন্তু পূর্ব-বৃদ্ধির ফলাফলের ওপর চালানো যায়। তুমি যদি পরীক্ষা করতে চাও তাহলে ++++x; বা (++x)++; চে-

ষ্টা করো, সংকলন (compile) হয়ে যাবে, কিন্তু x++++ বা ++(x++) চেষ্টা করো, সংকলন হবে না, ত্রুটি (error) আসবে পরের বৃদ্ধিটার জন্য "l-value required"। তুমি যদি স্রেফ ++x++; লিখো, এটা কিন্তু সংকলন হবে না, ত্রুটি দেখাবে, কারণ হলো পূর্ব ও উত্তর বৃদ্ধির মধ্যে উত্তর বৃদ্ধির অগ্রগণ্যতা (precedence) আগে, ফলে ++x++ আসলে ++(x++) এর সমতুল। অগ্রগণ্যতার ক্রমের (precedence order) নিয়মগুলো আমরা পরের এক পাঠে বিস্তারিত জানবো।

এবারে আমরা বৃদ্ধি ব্যবহারে ক্রমলেখয়ের গতির ওপর প্রভাব নিয়ে একটু আলোচনা করি। বৃদ্ধি (increment) ++x বা x++ সাধারণত x+=1 বা x=x+1 এর চেয়ে দ্রুতগতির, এর কারণ মুলত একদম যন্ত্র পর্যায়ে x++ বা ++x বিশেষ ভাবে নির্বাহিত হয় কিন্তু x+=1 বা x=x+1 সাধারণ যোগের মতো করে নির্বাহিত হয়। সাধারণতো পূর্ব-বৃদ্ধি (pre-increment) আর উত্তর-বৃদ্ধির (post-increment) মধ্যে পূর্ব-বৃদ্ধি দ্রুত গতির। কারণ হলো, উত্তর-বৃদ্ধির ফলাফল যেহেতু x এর মান বৃদ্ধি করবার আগের মান, তাই ওই আগের মানটি প্রথমে কোথাও ক্ষণস্থায়ী (temporarily) ভাবে রেখে দিতে হয়, আর x এর মান বৃদ্ধিটা তারপর ঘটে, আর তারপর ক্ষণস্থায়ী ভাবে রাখা মানটা ফলাফল হিসাবে আসে যেটি y = x++; এর ক্ষেত্রে y এ আরোপিত হয়। কিন্তু পূর্ব-বৃদ্ধির ক্ষেত্রে মান বৃদ্ধি আগে ঘটে আর ফলাফলটাও সেই বৃদ্ধিপ্রাপ্ত মানই, কাজেই ক্ষণস্থায়ী ভাবে আগের মান রেখে দেওয়ার কোন বোঝা (overhead) এখানে নেই। মোটকথা পূর্ব-বৃদ্ধি সরাসরি বাম-মানের ওপরই কাজ করে অর্থাৎ ++x এ সরাসরি চলকটার ওপরই কাজ করে, আর কোন ক্ষণস্থায়ী কিছু দরকার হয় না। এ কারণে পূর্ব-বৃদ্ধি ++x; উত্তর-বৃদ্ধি x++; এর চেয়ে বেশী দ্রুতগতির হয়ে থাকে। কাজেই তুমি পারতো পক্ষে ++x ব্যবহার করবে, x++ ব্যবহার করবে না।

ক্রমলেখতে বৃদ্ধি ব্যবহারে এবারে একটা পরামর্শ দেই। পূর্ব-বৃদ্ধি ও উত্তর-বৃদ্ধি নিয়ে অনেক রকম খেলা যায়, যেমন তুমি চাইলে x = (++x)++ + ++x; এর মতো অনেকগুলো + চিহ্ন দিয়ে কিছু একটা লিখতে পারো। এই রকম জটিল বিবৃতিগুলো হয়তো সংকলন (compile) হবে। এর ফলে ফলাফলও কিছু একটা আসবে, যেটা চাইলে বুঝা সম্ভব, কিন্তু বুঝতে গেলে মাথা বেশ গরম হয়ে যায়। আমার পরামর্শ হলো এইরকম জটিল বিবৃতি পারতো পক্ষে লেখবে না। সবসময় এমন ভাবে সংকেত (code) লিখবে যাতে পরে তুমি বা অন্য কেউ তেমন কোন কষ্ট ছাড়াই তোমার সংকেত দেখে বুঝতে পারে। মনে রাখবে সংকেত যত জটিল, তার ভুল বের করাও তত কঠিন।

উপরের পুরো আলোচনাতে আমরা কেবল বৃদ্ধি (increment) নিয়ে আলোচনা করেছি। আসলে হ্রাস (decrement) −− নিয়ে আলোচনাটা একদম একই রকম। আমরা তাই পুনরাবৃত্তি করবো না। কেবল জেনে রাখো হ্রাসের (decrement) ফলে মান 1 কমে যায়। তাই −−x বা x−− হলো x −= 1 বা x = x − 1 এর সমতুল। আমরা −−x কে পূর্ব হ্রাস (pre-decrement) আর x−− কে উত্তর-হ্রাস (post-decrement) বলি। পূর্ব-হ্রাসের তুলনায় উত্তর-হ্রাসের অগ্রগণ্যতা (precedence) বেশী। গতির দিক বিবেচনায় পূর্ব-হ্রাস, উত্তর-হ্রাসের চেয়ে শ্রেয়তর।

৬.৭ বির্তি অণুক্রিয়া (Comma Operator)

সিপিপিতে বির্তি অণুক্রিয়া (comma operator) কয়েকটি রাশি (expression) কে এক সাথে পরপর লেখায় সাহায্য করে। বির্তি (comma) অণুক্রিয়ার বামপাশের উপাদানের (operand) মান সব সময় নর্থক (void) হয় আর উপেক্ষিত হয়। এর অর্থ হচ্ছে ডান পাশের উপাদানটির (operand) মানই বির্তি অণুক্রিয়ার (comma operator) ফলাফল হয়।

একটা উদাহরণ দেখি x = (y=3, y+1); এই বিবৃতির ফলে বন্ধনীর ভিতরে প্রথমে বির্তির বাম পাশের রাশি হিসাবে y এর মান আরোপিত (assign) হবে 3। যদিও আরোপনের কারণে আমরা y এ 3 আরোপণের পাশাপাশি ফলাফলও পাই 3, কিন্তু বির্তির (comma) কারণে সেই

ফলাফল বাদ গিয়ে ফলাফল হয়ে যাবে নর্থক (void)। যাইহোক এরপর বির্তির (comma) ডান পাশের রাশি হিসাবে y+1 এর মান 3+1 বা 4 হবে যেটি আসলে যোগেরও + ফলাফল। আর যোগের এই ফলাফল 4 ই শেষ পর্যন্ত x চলকে আরোপিত হবে। এখানে বন্ধনী দরকার কারণ বির্তি (comma) , সাধারণত আরোপণের (assignment) = পরে হিসাব করা হয়। আমরা বন্ধনীর ভিতরের আরোপণটি y = 3 বির্তির (comma) আগে করতে চাইলেও বন্ধনীর বাইরের চলক x এ আরোপণটি বির্তির পরে করতে চাই, আর এ কারণে বন্ধনী জরুরী। ব্যাপারটি আরো পরিস্কার বুঝতে চাইলে একই জিনিস বন্ধনী ছাড়া কী হবে দেখো x = y = 3, y + 1;। এখানে দুটো আরোপণই (assignment) বির্তির (comma) আগে নির্বাহিত (execute) হবে। ফলে প্রথমে y এর মান আরোপিত হবে 3, তারপর x এও মান 3ই আরোপিত হবে, তারপর y+1 হিসাব হবে 4। এই 4 বির্তির ফলাফল হলেও সেটি কিন্তু এখানে কিছুতে আরোপিত হয় নি।

বির্তি (comma) অণুক্রিয়া (operator) হিসাবে ব্যবহার হলেও এর আরো নানান ব্যবহার আছে সিপিপিতে। যেমন একাধিক চলক (variable) একসাথে ঘোষণা (declare) করতে আমরা বির্তি (comma) দিয়ে লিখি int x, y, z = 3; বির্তির (comma) এই রকম ব্যবহার আসলে অণুক্রিয়া হিসাবে নয়, বরং তালিকার পৃথকী (separator) হিসাবে ব্যবহার। আমরা যখন পরে জন্য-ঘূর্ণী (for-loop) ও পরামিতি (parameter) নিয়ে আলোচনা করবো তখনও তালিকা পৃথকী (list separator) হিসাবে বির্তির (comma) ব্যবহার দেখতে পাবো।

৬.৮ অগ্রগণ্যতার ক্রম (Precedence Order)

অগ্রগণ্যতার ক্রম (precedence order) কী? সিপিপিতে এ পর্যন্ত পরিচিত হওয়া অণুক্রিয়াগুলোর (operator) অগ্রগণ্যতার ক্রম (precedence order) আলোচনা করো।

ধরো তুমি 3 + 4∗5 + 6 এর মান হিসাব করবে। আগেকার দিনে এক রকম সস্তা কলনি (calculator) পাওয়া যেতো যেটি করতো কী, বাম থেকে হিসাব করতো একের পর এক। ফলে সেটা প্রথম 3 ও 4 যোগ করে 7 বের করতো, তারপর তার সাথে 5 গুণ করে বের করতো 35 আর শেষে তার সাথে 6 যোগ করে ফল দিতো 41। তুমি চাইলে উল্টো আরেক রকমের অবস্থা ভাবতে পারো, যেখানে ডান দিক থেকে একের পর এক হিসাব হবে। সুতরাং 5 ও 6 যোগ করে 11, তারসাথে 4 গুণ করে 44, শেষে 3 যোগ করে 47। কিন্তু ছোটবেলা থেকে সরলের নিয়ম আমরা শিখে এসেছি: গুণ আগে হবে যোগ পরে হবে। আমরা তাই হিসাব করি 4 ও 5 এর গুণ আগে ফল 20 তার সাথে বামের যোগ আগে, তাই 3 আগে যোগ হলো 23, শেষে ডানের যোগ তাই 6 যোগ করে হলো 29, যেটাকে আমরা সঠিক হিসাব বলে ধরে নেই। এই যে বাম থেকে ডানে বা ডান থেকে বামে হিসাব না করে গুণ যোগের আগে করতে হবে, আবার দুটো যোগ পর পর থাকলে বামের যোগ আগে করতে হবে। এই নিয়মগুলোকে অগ্রগণ্যতার ক্রম(precedence order) বলা হয়।

সরল অংকে অগ্রগণ্যতার ক্রম ছিল: বন্ধণী, এর, ভাগ, গুণ, যোগ, বিয়োগ। সবেচেয়ে ভিতরের বন্ধনী সবেচেয়ে আগে। ভাগ আর গুণ আসলে বাম থেকে যেটা আগে আসে। একই ভাবে যোগ ও বিয়োগ বাম থেকে যেটা আগে আসে। সিপিপিতে আমরা এ পর্যন্ত অনেকগুলো অণুক্রিয়ার (Operator) সাথে পরিচিত হয়েছি। এগুলো হলো একিক + − ++ −− দুয়িক + − ∗ / % = += −= ∗= /− %= , তো এদের মধ্যে একিক অণুক্রিয়ার ক্রম সবার আগে, তারপর দুয়িক অণুক্রিয়াগুলোর ক্রম। আমরা আপাতত কেবল এগুলোর অগ্রগণ্যতার ক্রম (precedence order) বিবেচনা করবো। অন্যান্য অণুক্রিয়া ও তাদের ক্রম সম্পর্কে আমরা পরে জানবো।

১. ++ −− ২টি একিক অণুক্রিয়া (unary operator) উত্তর-বৃদ্ধি ও উত্তর-হ্রাস (post-increment and post-decrement) x++, x−− এরা বাম-মানের (l- value) ওপরে প্রযুক্ত হয়ে ডান-মান (r-value) ফল দেয়। ফলে x++++ বা x−−−− করা যায় না।

২. ++ −− + − ৪টি একিক অণুক্রিয়া (unary operator) পূর্ব-বৃদ্ধি (pre-increment) ++x ও পূর্ব-হ্রাস (pre-decrement) −−x এরা বাম-মানের (l-value) ওপর প্রযুক্ত হয়ে বাম-মানই ফল দেয়। ফলে ++++x বা −−−−x করা যায়, আর সবেচেয়ে ডানের ++ বা −− আগে প্রযুক্ত হয়। (পূর্ব) একিক অণুক্রিয়া (unary operator) +x ধনাত্মক (positive) −x আর ঋণাত্মক (negative) এরা ডান-মানের (r-value) ওপর প্রযুক্ত হয়ে ডান-মানই দেয়। ফলে + + x বা − − x করা সম্ভব, খেয়াল করো দুটো + বা দুটো − এর মধ্যে ফাঁকা দিতে হয়েছে না হলে ওগুলো বৃদ্ধি বা হ্রাস হিসাবে চিহ্নিত হয়ে যাবে।

৩. ∗ / % ৩টি দুয়িক অণুক্রিয়া (binary operator) এরা দুটি ডান-মানের (r-value) উপাদানের (operand) ওপর প্রযুক্ত হয়ে ডান-মানই ফল দেয়। এই অণুক্রিয়াগুলো পরপর অনেকগুলো থাকলে বাম থেকে ডানে একে একে হিসাব হতে থাকে। যেমন 10 / 2 ∗ 4 % 6 এ বাম থেকে ডানে প্রথমে ভাগফল, তারপর গুণফল, তারপর ভাগশেষ হিসাব হবে।

৪. + − ২টি দুয়িক অণুক্রিয়া (binary operator) এরা দুটি ডান-মানের (r-value) উপাদানের (operator) ওপর প্রযুক্ত হয়ে ডান-মানই ফল দেয়। এই অণুক্রিয়াগুলো পরপর অনেকগুলো থাকলে বাম থেকে ডানে একে একে হিসাব হতে থাকে। যেমন 10 − 2 + 5 এ বাম থেকে ডানে প্রথমে বিয়োগফল, তারপর যোগফল হিসাব হবে।

৫. = += −= ∗= /= %= এই সব দুয়িক অণুক্রিয়া (binary operator) আরোপণগুলোর (assignment) বামপাশে এমন কিছু থাকতে হবে যার বাম-মান (l-value) আছে, আর ডান পাশে এমন কিছু থাকতে হয় যার ডান-মান (r-value) আছে। এই অণুক্রিয়াগুলো পরপর অনেগুলো থাকলে ডান থেকে বামে একে একে হিসাব হতে থাকে। যেমন x += y = z ∗= 3 তে প্রথমে ডানের ∗= এর কারণে z এর সাথে 3 গুণ হবে, তারপর মাঝের = এর কারণে z এর মান y আরোপিত হবে, শেষে y এর মান x এর সাথে যোগ হবে।

৬. , বির্তি (comma) একটি দুয়িক অণুক্রিয়া (binary operator) যেটির ফলাফল কেবল ডানপাশের উপাদান (operand)। বাম পাশের উপাদানটি হিসাব হয়, কিন্তু তার ফলাফল হবে নর্থক (void)। এই অণুক্রিয়া একাধিক পরপর থাকলে, বাম থেকে ডানে একে একে হিসাব হতে থাকে। যেমন x + 2, y ∗ 3, z / 4 প্রথমে যোগ হবে, তারপর গুণ আর শেষে ভাগ, ফলাফল হবে একদম ডানের ভাগফলটিই।

দুটো একই বা একই ক্রমের অণুক্রিয়া পরপর থাকলে কোন পাশেরটি আগে হবে এইটি নির্ধারণ করে দেয়াকে বলা হয় সহযোজ্যতা (associativity)। যেমন x−y−z থাকলে আমাদের প্রথমে বামের বিয়োগ করতে হবে, তারপর ডানের বিয়োগ, কাজেই বিয়োগ হল বাম সহযোজ্য (left associative) অর্থাৎ x − y − z আর (x −y) − z একই। খেয়াল করো বিয়োগ কিন্তু ডান সহযোজ্য (right associative) নয় কারণ x − y − z আর x − (y − z) এক নয়। যোগ আবার বাম ও ডান উভয় সহযোজ্য কারণ x + y + z, (x + y) + z ও x + (y + z) একই। সাধারণত উভয় সহযোজ্যদের ক্ষেত্রে সুবিধার্থে তাদের বাম-সহযোজ্য হিসাবে বিবেচনা করা হয়। উপরের তালিকায় আলোচিত অণুক্রিয়াগুলোর ক্ষেত্রে একই রকম অণুক্রিয়া পরপর থাকলে কোন পাশেরটি আগে হবে, সেটাও কিন্তু আলোচনা করা হয়েছে। সেখান থেকে বুঝতো পারো কোন অণুক্রিয়া বাম সহযোজ্য (left associative), আর কোনটি ডান সহযোজ্য (right associative)?

সবেশেষে একটা গুরুত্বপূর্ণ বিষয় মনে রাখবে বন্ধনী () এর শক্তি কিন্তু সবচেয়ে বেশী। যে কোন স্থানে কোন রকমের দ্বিধাদ্বন্দ্ব থাকলে সেখানে বন্ধনী ব্যবহার করে দ্বিধা পরিস্কার করবে। অণুক্রিয়াগুলোর (operator) অগ্রগণ্যতার ক্রম (precedence order) ব্যবহার করে নানা রকম জটিল জটিল বিবৃতি ও রাশি (statement and expression) তৈরী করা যায়, যেগুলো ক্রম

বিবেচনায় নিয়ে বুঝতে গেলে মাথা গরম হয়ে যেতে পারে, ভুল হলে বের করা কঠিন হয়ে যাবে। কাজেই আমার পরামর্শ হচ্ছে তোমার বিবৃতি বা রাশি অবশ্যই সহজে পাঠযোগ্য হতে হবে, আর এ কাজে যত দরকার বন্ধনী ব্যবহার করবে। যেমন ধরো x += y − z এর চেয়ে x += (y − z) বুঝা আমাদের জন্য বেশী সহজ, কারণ এতে একদম পরিস্কার বিয়োগ আগে হবে।

৬.৯ গাণিতিক সমস্যা (Mathematical Problems)

দ্বিমাত্রিক স্থানাঙ্ক ব্যবস্থায় (two dimensional coordinate system) দুটি বিন্দুর স্থানাঙ্ক যোগান (input) নিয়ে তাদের মাঝে দূরত্ব ফলন (output) হিসাবে দেখাও। ধরো স্থানাঙ্কগুলো ভগ্নকে দেয়া আছে। তোমার নিশ্চয় জানা আছে যে দুটো বিন্দুর (x_1, y_1) ও (x_2, y_2) দূরত্ব হলো $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ অর্থাৎ ভুজদ্বয়ের দূরত্বের বর্গ ও কোটিদ্বয়ের দূরত্বের বর্গের যোগফলের বর্গমূল। বর্গমূল নির্ণয়ের জন্য cmath শির নথি (header file) থেকে sqrt বিপাতক ব্যবহার করো। আর বর্গ নির্ণয়ের জন্য তোমাকে একই জিনিস দুইবার গুণ করতে হবে।

ফিরিস্তি ৬.৫: দুটি বিন্দুর মধ্যের দূরত্ব (Distance Between Two Points)

```
// নীচের শির নথি main বিপাতকের বাইরে অন্তর্ভুক্ত করো

#include <cmath>  // বর্গমূল নির্ণয়ের জন্য  sqrt বিপাতক লাগবে

// নীচের অংশ main বিপাতকের ভিতরে return এর আগে লিখো

float x1, y1, x2, y2; // স্থানাঙ্ক দুটো (x1,y1), (x2,y2)

cout << "prothom bindur x y: ";      // যোগান যাচনা
cin >> x1 >> y1;                     // প্রথম বিন্দু যোগান
cout << "ditiyo bindur x y: ";       // যোগান যাচনা
cin >> x2 >> y2;                     // দ্বিতীয় বিন্দু যোগান

float xd = abs(x1 - x2);             // ভুজ দুটির দূরত্ব
float yd = abs(y1 - y2);             // কোটি দুটির দূরত্ব

float dd = sqrt(xd * xd + yd * yd); // দূরত্ব হিসাব করো
cout << "bindu dutor durotto: " << dd << endl;// ফলন
```

উপরের ক্রমলেখ খেয়াল করো। খুবই সাদামাটা। প্রথমে main বিপাতকের বাইরে cmath শির নথি অন্তর্ভুক্ত করতে হবে বলে দেখানো হয়েছে। তারপর main বিপাতকের ভিতরে বিন্দু দুটোর ভুজ ও কোটি ধারণ করার জন্য চারটি float ধরনের ভগ্নক চলক (variable) নেয়া হয়েছে। এরপর যোগান যাচনা (input prompt) দিয়ে বিন্দুদুটোর স্থানাঙ্ক যোগান (input) নেয়া হয়েছে। তারপর ভুজ দ্বয়ের দূরত্ব abs(x1 − x2) বের করে xd নামের আরেকটি চলকে নেয়া হয়েছে, একই ভাবে কোটিদ্বয়ের দূরত্ব abs(y1 − y2) বের করে yd নামের আরেকটি চলকে নেয়া হয়েছে। মনে করে দেখো abs বিপাতকটি (function) কোন সংখ্যার পরম মান (absolute value) অর্থাৎ চিহ্ন বাদ দিয়ে কেবল মানটুকু ফেরত দেয়। যাইহোক তারপর xd এর বর্গ ও yd এর বর্গের যোগফল নিয়ে তার বর্গমূল বের করা হয়েছে sqrt বিপাতক ব্যবহার করে আর রাখা

হয়েছে dd চলকে। সবশেষে দূরত্ব dd চলক থেকে ফলন (output) দেয়া হয়েছে। এখানে এক-টা কথা বলে রাখি sqrt(xd ∗ xd + yd ∗ yd) এর বদলে cmath শির নথি (header file) থেকেই hypot নামের বিপাতকও (function) আমরা ব্যবহার করতে পারতাম। সেক্ষেত্রে আমা-দের লিখতে হতো hypot(xd, yd) আর সেটি ঠিক একই কাজ করতো। বিপাতক hypot আসলে সমকোণী ত্রিভুজের অতিভুজের দৈর্ঘ্য নির্ণয় করে, কিন্তু তার সূত্র আর দুটো বিন্দুর দূরত্ব নির্ণয়ের সূত্রের মধ্যে মিল রয়েছে।

৬.১০ শির নথি cmath (Header File cmath)

শির নথি cmath এ গাণিতিক প্রক্রিয়াকরণে ব্যবহৃতব্য নানান বিপাতক (function) আছে। আম-রা এখানে ওই বিপাতকগুলোর সাথে সংক্ষিপ্ত আকারে পরিচিত হবো। এই বিপাতকগুলো কী তা বুঝতে তোমার উচ্চমাধ্যমিক গণিতের ধারণাবলী দরকার হবে। নীচের পরাবৃত্তীয় (hyperbolic) বিপাতকগুলো ছাড়া প্রায় সবগুলো বিপাতকই আমাদের প্রায়শই কাজে লাগে।

গাণিতিক বিপাতক (Mathematical Functions)

- abs(x): কোন সংখ্যা x এর পরম মান। abs(3) হবে 3 এবং abs(−3) হবে 3।

ত্রিকোণমিতিক বিপাতক (Trigonometric Functions)

- cos(x): লগ্নানুপাত (cosine) যেখানে x হল রেডিয়ানে।
- sin(x): লম্বানুপাত (sine) যেখানে x হল রেডিয়ানে।
- tan(x): স্পর্শানুপাত (tangent) যেখানে x হল রেডিয়ানে।
- acos(x): বিলগ্নানুপাত (arc-cosine) যেখানে ফেরত মান রেডিয়ানে।
- asin(x): বিলম্বানুপাত (arc-sine) যেখানে ফেরত মান রেডিয়ানে।
- atan(x): বিস্পর্শানুপাত (arc-tangent) যেখানে ফেরত মান রেডিয়ানে।
- atan2(x,y): বিস্পর্শানুপাত (arc-tangent) যেখানে $\frac{x}{y}$ এর x হল লব (numerator) আর y হল হর (denominator) আর ফেরত মান রেডিয়ানে।

পরাবৃত্তীয় বিপাতক (Hyperbolic Functions)

- cosh(x): পরাবৃত্তীয় লগ্নানুপাত (hyperbolic cosine) যেখানে x হল রেডিয়ানে।
- sinh(x): পরাবৃত্তীয় লম্বানুপাত (hyperbolic sine) যেখানে x হল রেডিয়ানে।
- tanh(x): পরাবৃত্তীয় স্পর্শানুপাত (hyperbolic tangent) যেখানে x হল রেডিয়ানে।
- acosh(x): পরাবৃত্তীয় বিলগ্নানুপাত (hyperbolic arc-cosine), ফেরত রেডিয়ানে।
- asinh(x): পরাবৃত্তীয় বিলম্বানুপাত (hyperbolic arc-sine), ফেরত রেডিয়ানে।

- atanh(x): পরাবৃত্তীয় বিস্পর্শানুপাত (hyperbolic arc-tangent), ফেরত রেডিয়ানে।

সুচক ও ঘাতাঙ্ক (Exponents and Logarithms)

- exp(x): e^x বা সুচকীয় বিপাতক (exponential function)
- log(x): $\log_e x$ বা ঘাতাঙ্ক বিপাতক (logarithmic function)
- log10(x): $log_{10}x$ বা ১০-ভিত্তিক ঘাতাঙ্ক (logarithm)
- exp2(x): 2^x বা ২-ভিত্তিক সুচকীয় (exponential) বিপাতক
- log2(x): log_2x বা ২-ভিত্তিক ঘাতাঙ্ক (logarithm)

শক্তি ও ঘাত (Power and Index)

- pow(x,y): x^y অর্থাৎ x এর y তম শক্তি যেমন pow(2,3) হল 2^3 বা 8
- sqrt(x): $\sqrt{x}$ অর্থাৎ x এর বর্গমূল যেমন sqrt(16.0) হল 4.0
- cbrt(x): $\sqrt[3]{x}$ অর্থাৎ x এর ঘনমূল যেমন cbrt(8.0) হল 2.0
- hypot(x,y): $\sqrt{x^2+y^2}$ অর্থাৎ x ও y কে সমকোনী ত্রিভুজের লম্ব (perpendicular) ও ভুমি (base) ধরলে অতিভুজের (hypotenuse) দৈর্ঘ্য

নৈকটায়নের বিপাতক (Rounding Functions)

- round(x): নৈকটায়ন বিপাতক x এর নিকটতম পূর্ণক।
- floor(x): মেঝে বিপাতক x এর সমান বা ঠিক ছোট পূর্ণকটি।
- ceil (x): ছাদ বিপাতক x এর সমান বা ঠিক বড় পূর্ণকটি।
- trunc(x): কর্তন বিপাতক x এর ভগ্নাংশটুকু কেটে ফেলবে।

উপরের বিপাতকগুলোর ফলাফল বুঝার জন্য নীচের সারণী লক্ষ্য করো।

মান x	নৈকটায়ন round(x)	মেঝে floor(x)	ছাদ ceil (x)	কর্তন trunc(x)
2.3	2.0	2.0	3.0	2.0
2.8	3.0	2.0	3.0	2.0
2.5	3.0	2.0	3.0	2.0
2.0	2.0	2.0	2.0	2.0
-2.3	-2.0	-3.0	-2.0	-2.0
-2.8	-3.0	-3.0	-2.0	-2.0
-2.5	-3.0	-3.0	-2.0	-2.0

৬.১১ অনুশীলনী সমস্যা (Exercise Problems)

ধারণাগত প্রশ্ন: নীচে কিছু ধারণাগত প্রশ্ন রয়েছে। প্রশ্নগুলোর উত্তর নিজে নিজে বের করবে।

১. বিপাতক (function) ও রাশি (expression) বলতে কী বুঝো? উদাহরণ দাও।

২. একিক (unary) ও দুয়িক (binary) অণুক্রিয়া (operation) বলতে কী বুঝো? কয়েকটা করে একিক (unary) ও দুয়িক (binary) অণুক্রিয়ার (operation) নাম বলো।

৩. উপাত্ত প্রকারান্তর (type casting) কী? দুয়িক অণুক্রিয়ায় (binary operation) কী ভাবে উপাত্ত প্রকারান্তর (type casting) হয়?

৪. নির্বাহ-কালীন ত্রুটি (execution-time error) বলতে কী বুঝো? ভাগফল ও ভাগশেষ নির্ণয়ের সময় কোন নির্বাহকালীন ত্রুটি ঘটতে পারে?

৫. ক্রমলেখতে অদরকারী সংকেত (code) মুছে না দিয়ে কীভাবে আমরা টীকা (comment) ব্যবহার করে সেগুলোকে অকার্যকর করে রাখতে পারি, ব্যাখ্যা করো।

৬. ঋণাত্মক পূর্ণকের (integer) ভাগফল ও ভাগশেষ নির্ণয়ের নিয়ম বর্ণনা করো।

৭. আরোপণ অণুক্রিয়ার (assignment operator) ফলাফল ঠিক কী? যৌগিক আরোপণ (compound assignment) বলতে কী বুঝো? কয়েকটি উদাহরণ দাও।

৮. সাধারণ যৌগিক আরোপন (compound assignment) যেমন x += 1 ব্যবহার না করে কেন বৃদ্ধি (increment) x++ বা ++x কেন ব্যবহার করা হয়?

৯. উত্তর-বৃদ্ধি (post-increment) ও পূর্ব-বৃদ্ধি (pre-increment) এর মধ্যে পার্থক্যগুলো আলোচনা করো। তুমি কোনটি ব্যবহার করতে চাইবে এবং কেন?

১০. বির্তি (comma) অণুক্রিয়ার কাজ কী? এর ফলাফলই বা কী?

১১. অগ্রগণ্যতার ক্রম (precedence order) ও সহযোজ্যতা (associativity) কী?

১২. সিপিপিতে এ পর্যন্ত শেখা অণুক্রিয়াগুলোর (operator) অগ্রগণ্যতার ক্রম (precedence order) ও সহযোজ্যতা (associativity) আলোচনা করো।

পরিগণনার সমস্যা: নীচে আমরা কিছু পরিগণনার সমস্যা দেখবো। এই সমস্যাগুলো আগে ধৈর্য্য ধরে নিজে নিজে সমাধান করতে চেষ্টা করবে। যখন একেবারেই পারছো না বলে মনে হয় তখনই কেবল সমাধান দেখে নিতে পারো। সমাধানগুলো পরিগণনার প্রশ্নগুলোর শেষে আছে।

১. একটি সমান্তর ধারার (arithmetic series) প্রথম পদ a সাধারণ অন্তর d হলে n-তম পদ কতো? n পদের সমষ্টিই বা কত? এর জন্য সিপিপিতে একটা ক্রমলেখ (program) তৈরী করো যেটি a, d, ও n যোগান (input) নিবে, আর n-তম পদ ও n পদের সমষ্টি ফলন (output) দিবে। এর জন্য তুমি সূত্র ব্যবহার করবে n-তম পদ = $a + (n-1) * d$ আর n পদের সমষ্টি = $n * (2a + (n-1) * d)/2$। প্রদত্ত বিভিন্ন ধারার জন্যে এই সূত্র a আর d বসালে আমরা ওই ধারাগুলোর জন্য সরাসরি সূত্র পেতে পারি।

- $1+2+3+\ldots$ ধারাতে $a=1, d=1$। সুতরাং n-তম পদ = $a+n-1$, n পদের সমষ্টি = $n(n+1)/2$। যেমন $n=10$ হলে 10-তম পদ 10, সমষ্টি 55।
- $2+4+6+\ldots$ ধারাতে $a=2, d=2$। সুতরাং n-তম পদ = $2n$, n পদের সমষ্টি = $n(n+1)$। যেমন $n=10$ হলে 10-তম পদ 20, সমষ্টি 110।
- $1+3+5+\ldots$ ধারাতে $a=1, d=2$। সুতরাং n-তম পদ = $2n-1$, n পদের সমষ্টি = n^2। যেমন $n=10$ হলে 10-তম পদ 19, সমষ্টি 100।

২. নীচের মতো ফলন (output) দেয় এরকম একটি ক্রমলেখ (program) তৈরী করো। ফলের স্তম্ভটিতে তুমি দুয়িক অণুক্রিয়াগুলো (binary operator) ব্যবহার করবে।

```
x=10  y=5

rashi  fol
x=y+3  x=  8
x=y-2  x=  3
x=y*5  x=  25
x=x/y  x=  2
x=x%y  x=  0
```

৩. এমন একটি ক্রমলেখ (program) রচনা করো যেটি একটি তিন অঙ্কের সংখ্যাকে উল্টো করে যেমন 326 হয়ে যায় 623। এ কাজে তুমি ভাগফল, ভাগশেষ, গুণ, যোগ ও বিয়োগ ব্যবহার করবে। 326 থেকে অঙ্কগুলো আলাদা করে তারপর 623 তৈরী করবে।

৪. একটি ত্রিভুজের তিন বাহুর দৈর্ঘ্য a, b, c যোগান (input) নিয়ে ত্রিভুজটির ক্ষেত্রফল নির্ণয় করো। তুমি হয়তো জানো ত্রিভুজের ক্ষেত্রফল = $\sqrt{s(s-a)(s-b)(s-c)}$ যেখানে s হলো অর্ধ পরিসীমা অর্থাৎ $s=(a+b+c)/2$।

৫. এমন একটি ক্রমলেখ (program) রচনা করো যেটি সেকেন্ড যোগান নিয়ে তাকে ঘন্টা-মিনিট-সেকেন্ডে রূপান্তর করে। এ কাজে তুমি ভাগফল ও ভাগশেষ ব্যবহার করবে।

৬. একটি ত্রিভুজের তিন বাহুর দৈর্ঘ্য a, b, c যোগান (input) নিয়ে এর কোণগুলো নির্ণয় করো। ধরো ত্রিভুজের কোন তিনটি A, B, C। এখান A, B, C যথাক্রমে a, b, c বাহুর বিপরীত কোণ। তুমি হয়তো জানো কোণ $C=\cos^{-1}((a^2+b^2-c^2)/(2ab))$, কোণ $B=\cos^{-1}((c^2+a^2-b^2)/(2ca))$ ও কোণ $A=\cos^{-1}((b^2+c^2-a^2)/(2bc))$। তোমার ক্রমলেখতে ত্রিভুজের কোনগুলোকে তুমি ডিগ্রীতে রূপান্তর করে ফলন দিবে।

৭. এমন একটি ক্রমলেখ (program) রচনা করো যেটি দুটো সময় ঘন্টা, মিনিট, সেকেন্ডে নিয়ে সময় দুটিকে যোগ করে। এ কাজে তুমি যোগ, ভাগফল ও ভাগশেষ ব্যবহার করবে।

৮. এমন একটি ক্রমলেখ রচনা করো যেটি দুটো সমীকরণ $ax+by=c$ ও $dx+ey=f$ এর a, b, c, d, e, f যোগান নিয়ে x ও y এর মান ফলন দেয়।

৯. একটি বাস u আদিবেগ ও a সমত্বরণ নিয়ে যাত্রা শুরু করলো। সময় t সেকেন্ড পরে বাসের গতিবেগ v নির্ণয় করো। t সময় পরে বাসটি অতিক্রান্ত দুরত্ব sও নির্ণয় করো। এ কাজে তুমি গতিবিদ্যার সূত্র $v=u+at$ ও $s=ut+\frac{1}{2}at^2$ ব্যবহার করবে।

৬.১১. অনুশীলনী সমস্যা (Exercise Problems)

১০. নীচের ছদ্ম-সংকেতের (pseudocode) জন্য একটি ক্রমলেখ (program) তৈরী করো।

ক) পড়ো (read) x ও y

খ) গণো (compute) $p = x * y$

গ) গণো (compute) $s = x + y$

ঘ) গণো (compute) $t = s^2 + p * (s - x) * (p + y)$

ঙ) লিখো (write) t

পরিগণনা সমাধান: এবার আমরা পরিগণনার সমস্যাগুলোর সমাধান দেখবো। মনে রাখবে সমাধানগুলো দেয়া হয়েছে তুমি যদি একান্তই নিজে নিজে করতে পারছোনা বলে মনে হয়, তখন কেবল একটু সাহায্য যাতে পেতে পারো তাই। কাজেই পারতপক্ষে সমাধান দেখবে না।

১. একটি সমান্তর ধারার (arithmetic series) প্রথম পদ a সাধারণ অন্তর d হলে n-তম পদ কতো? n পদের সমষ্টিই বা কত? এর জন্য সিপিপিতে একটা ক্রমলেখ (program) তৈরী করো যেটি a, d, ও n যোগান (input) নিবে, আর n-তম পদ ও n পদের সমষ্টি ফলন (output) দিবে। এর জন্য তুমি সূত্র ব্যবহার করবে n-তম পদ = $a + (n-1) * d$ আর n পদের সমষ্টি = $n * (2a + (n-1) * d)/2$। প্রদত্ত বিভিন্ন ধারার জন্যে এই সূত্র a আর d বসালে আমরা ওই ধারাগুলোর জন্য সরাসরি সূত্র পেতে পারি।

- $1 + 2 + 3 + \ldots$ ধারাতে $a = 1, d = 1$। সুতরাং n-তম পদ = $a + n - 1$, n পদের সমষ্টি = $n(n+1)/2$। যেমন $n = 10$ হলে 10-তম পদ 10, সমষ্টি 55।
- $2 + 4 + 6 + \ldots$ ধারাতে $a = 2, d = 2$। সুতরাং n-তম পদ = $2n$, n পদের সমষ্টি = $n(n+1)$। যেমন $n = 10$ হলে 10-তম পদ 20, সমষ্টি 110।
- $1 + 3 + 5 + \ldots$ ধারাতে $a = 1, d = 2$। সুতরাং n-তম পদ = $2n - 1$, n পদের সমষ্টি = n^2। যেমন $n = 10$ হলে 10-তম পদ 19, সমষ্টি 100।

আমরা এখানে কেবল সাধারণ সুত্রের জন্য ক্রমলেখ (program) তৈরী করবো। প্রদত্ত বিশেষ ধারার জন্য তুমি এই ক্রমলেখ (program) দরকার মতো বদলে নিতে পারবে।

ফিরিস্তি ৬.৬: সমান্তর ধারার সমস্যা (Arithmetic Series Problem)

```
int a, d, n;
cout << "prothom pod? "; cin >> a;
cout << "sadharon ontor? "; cin >> d;
cout << "kototom pod?"; cin >> n;

int t = a + (n - 1) * d;  //  n-তম পদ
cout << n << "-tom pod = " << t << endl;

int s = n * (2*a + (n - 1)*d) / 2; // সমষ্টি
cout << n << " poder somosti = " << s << endl;
```

যোগান-ফলন (input-output)

```
prothom pod? 1
sadharon ontor? 1
kototom pod? 10
10–tom pod = 10
10 poder somosti = 55
```

২. নীচের মতো ফলন (output) দেয় এরকম একটি ক্রমলেখ (program) তৈরী করো।

```
x=10 y=5

rashi fol
x=y+3 x= 8
x=y–2 x= 3
x=y*5 x= 25
x=x/y x= 2
x=x%y x= 0
```

ফিরিস্তি ৬.৭: দুয়িক অণুক্রিয়ার ফলাফল (Binary Operation Results)

```
int x = 10, y = 5;

cout << "x=" << x << " y=" << y << endl;
cout << endl; //  ফাঁকা সারি
cout << "rashi "<< "fol " << endl;
cout << "x=y+3" << " x= " << y+3 << endl;
cout << "x=y–2" << " x= " << y–2 << endl;
cout << "x=y*5" << " x= " << y*5 << endl;
cout << "x=x/y" << " x= " << x/y << endl;
cout << "x=x%y" << " x= " << x%y << endl;
```

৩. এমন একটি ক্রমলেখ (program) রচনা করো যেটি একটি তিন অঙ্কের সংখ্যাকে উল্টো করে যেমন 326 হয়ে যায় 623। এ কাজে তুমি ভাগফল, ভাগশেষ, গুণ, যোগ ও বিয়োগ ব্যবহার করবে। 326 থেকে অঙ্কগুলো আলাদা করে তারপর 623 তৈরী করবে।

```
int soja = 326;

int daner = soja % 10;  // ভাগশেষ 6
int bamer = soja / 100; // ভাগফল 3
int majher = soja / 10 % 10; //ফল 2

int ulta = bamer;      // উল্টা = 3
ulta += majher * 10;   // উল্টা = 23
ulta += daner * 100;   // উল্টা = 623
```

৬.১১. অনুশীলনী সমস্যা (Exercise Problems)

৪. একটি ত্রিভুজের তিন বাহুর দৈর্ঘ্য a, b, c যোগান (input) নিয়ে ত্রিভুজটির ক্ষেত্রফল নির্ণয় করো। তুমি হয়তো জানো ত্রিভুজের ক্ষেত্রফল = $\sqrt{s(s-a)(s-b)(s-c)}$ যেখানে s হলো অর্ধ পরিসীমা অর্থাৎ $s = (a+b+c)/2$।

ফিরিস্তি ৬.৮: ত্রিভুজের বাহু হতে ক্ষেত্রফল (Triangle's Area From Sides)

```
// main বিপাতকের বাইরে
#include <cmath>

// main বিপাতকের ভিতরে
float a, b, c;                  // বাহুগুলো
cout << "bahu a b c: "; // যোগান যাচনা
cin >> a >> b >> c;          // যোগান নেওয়া

float s = (a + b + c) / 2;  // অর্ধ পরিসীমা
float k = sqrt(s*(s-a)*(s-b)*(s-c));  // ক্ষেত্রফল

cout << "khetrofol = " << k << endl;  // ফলন
```

যোগান-ফলন (input-output)

```
bahu a b c: 100 60 90
khetrofol = 2666
```

৫. এমন একটি ক্রমলেখ (program) রচনা করো যেটি সেকেন্ড যোগান নিয়ে তাকে ঘন্টা-মিনিট-সেকেন্ডে রূপান্তর করে। এ কাজে তুমি ভাগফল ও ভাগশেষ ব্যবহার করবে।

ফিরিস্তি ৬.৯: সময়কে সেকেন্ডে প্রকাশ (Time in Seconds)

```
int motsekend = 38185;

int sekend = motsekend % 60; // ফল  25
int motminit = motsekend / 60;  // ফল 636

int minit = motminit % 60;  //  ফল 36
int ghonta = motminit / 60; // ফল 10
```

৬. একটি ত্রিভুজের তিন বাহুর দৈর্ঘ্য a, b, c যোগান (input) নিয়ে এর কোণগুলো নির্ণয় করো। ধরো ত্রিভুজের কোন তিনটি A, B, C। এখান A, B, C যথাক্রমে a, b, c বাহুর বিপরীত কোণ। তুমি হয়তো জানো কোণ $C = \cos^{-1}((a^2+b^2-c^2)/(2ab))$, কোণ $B = \cos^{-1}((c^2+a^2-b^2)/(2ca))$ ও কোণ $A = \cos^{-1}((b^2+c^2-a^2)/(2bc))$। তোমার ক্রমলেখতে ত্রিভুজের কোনগুলোকে তুমি ডিগ্রীতে রূপান্তর করে ফলন দিবে।

আমরা cmath শির নথি থেকে বিলগ্নানুপাতের (arccosine) জন্য acos বিপাতকটিকে (function) ব্যবহার করবো। কিন্তু এটি আমাদের রেডিয়ানে কোণ ফেরত দিবে। রেডিয়ান থেকে ডিগ্রীতে নিতে চাইলে আমাদের 180/pi দিয়ে গুণ করতে হবে। কথা হচ্ছে পাই

কেমনে পাবো। আমরা pai একটা ধ্রুবক ঘোষণা করতে পারি যার মান দিয়ে দিব 3.1416 অথবা আরো নিখুত মান পেতে চাইলে acos(−1) থেকেও মান বের করে নিতে পারি।

ফিরিস্তি ৬.১০: ত্রিভুজের বাহু হতে কোণ (Triangle's Angles From Sides)

```
// main বিপাতকের বাইরে
#include <cmath>

// main বিপাতকের ভিতরে
float a, b, c;                  //  বাহুগুলো
cout << "bahu a b c: "; // যোগান যাচনা
cin >> a >> b >> c;          // যোগান নেওয়া

//  কোণ নির্ণয় রেডিয়ানে
float C = acos((a*a + b*b − c*c)/(2*a*b));
float B = acos((c*c + a*a − b*b)/(2*c*a));
float A = acos((b*b + c*c − a*a)/(2*b*c));

//  ডিগ্রীতে রূপান্তর
float const pai = arccos(−1); // বিকল্প হলো 3.1416
C *= 180/pai; B *= 180/pai; A *= 180/pai;

cout << "kon A B C= ";  // ফলন
cout << A << " " << B << " " << C << endl;
```

যোগান-ফলন (input-output)

```
bahu a b c: 145 60 90
kon A B C= 149.703 12.049 18.2475
```

৭. এমন একটি ক্রমলেখ (program) রচনা করো যেটি দুটো সময় ঘন্টা, মিনিট, সেকেন্ডে নিয়ে সময় দুটিকে যোগ করে। এ কাজে তুমি যোগ, ভাগফল ও ভাগশেষ ব্যবহার করবে।

ফিরিস্তি ৬.১১: দুটি সময়ের যোগ (Adding Two Times)

```
int ghonta1, minit1, sekend1; // ১ম সময় যোগান নিবে
int ghonta2, minit2, sekend2; // ২য় সময় যোগান নিবে

int sekend = sekend1 + sekend2; //  সেকেন্ড দুটো যোগ
int minit = minit1 + minit2;     // মিনিট দুটো যোগ
int ghonta = ghonta1 + ghonta2; //  ঘন্টা দুটো যোগ

minit += sekend / 60; // মোট সেকেন্ড 60 এর বেশী হলে
sekend = sekend % 60; // মিনিট হওয়ার পরে অবশিষ্ট সেকেন্ড
ghonta += minit / 60; // মোট মিনিট 60 এর বেশী হলে
minit = minit % 60;    // ঘন্টা হওয়ার পরে অবশিষ্ট মিনিট
```

৮. এমন একটি ক্রমলেখ রচনা করো যেটি দুটো সমীকরণ $ax+by=c$ ও $dx+ey=f$ এর a, b, c, d, e, f যোগান নিয়ে x ও y এর মান ফলন দেয়। এরকম সহ সমীকরণ সমাধানের সূত্র হল $x=(ce-bf)/(ae-bd)$ আর $y=(af-cd)/(ae-bd)$।

ফিরিস্তি ৬.১২: সহ সমীকরণ সমাধান (Simultaneous Equations)

```
float a, b, c, d, e, f;

cout << "prothom somikoron a b c:";
cin >> a >> b >> c;

cout << "ditiyo somikoron e f g:";
cin >> d >> e >> e;

float x = (c*e - b*f)/(a*e - b*d);
float y = (a*f - c*d)/(a*e - b*d);

cout << "x = " << x << " ";
cout << "y = " << y << endl;
```

যোগান-ফলন (input-output)

```
prothom somikoron a b c: 2 1 4
ditiyo somikoron e f g: 1 -1 -1
x = 1.33333 y = 1.33333
```

৯. একটি বাস u আদিবেগ ও a সমত্বরণ নিয়ে যাত্রা শুরু করলো। সময় t সেকেন্ড পরে বাসের গতিবেগ v নির্ণয় করো। t সময় পরে বাসটি অতিক্রান্ত দুরত্ব sও নির্ণয় করো। এ কাজে তুমি গতিবিদ্যার সূত্র $v=u+at$ ও $s=ut+\frac{1}{2}at^2$ ব্যবহার করবে।

ফিরিস্তি ৬.১৩: গতির সমীকরণ সমাধান (Solving Motion Equations)

```
float u, a, t;

cout << "adibeg toron somoy: ";
cin >> u >> a >> t;

float v = u + a * t;
float s = u*t + a * t * t / 2;

cout << "beg: " << v << " ";
cout << "durutto: " << s << endl;
```

যোগান-ফলন (input-output)

```
adibeg toron somoy: 2 1 4
beg: 6 durotto: 16
```

১০. নীচের ছদ্ম-সংকেতের (pseudocode) জন্য একটি ক্রমলেখ (program) তৈরী করো।

ক) পড়ো (read) x ও y
খ) গণো (compute) $p = x * y$
গ) গণো (compute) $s = x + y$
ঘ) গণো (compute) $t = s^2 + p * (s - x) * (p + y)$
ঙ) লিখো (write) t

ফিরিস্তি ৬.১৪: ছদ্মসংকেত থেকে ক্রমলেখ তৈরী (Program from Pseudocode)

```
int x, y;   //  কেবল main বিপাতকের ভিতরের অংশটুকু

cin >> x >> y;  //  ধাপ ক
int p = x * y;  //  ধাপ খ
int s = x + y;  //  ধাপ গ
int t = s*s + p * (s - x) * (p + y);  //  ধাপ ঘ

cout << t << endl;  // ধাপ ঙ
```

৬.১২ গণনা পরিভাষা (Computing Terminologies)

- ধনাত্মক (positive)
- ঋণাত্মক (negative)
- অণুক্রিয়া (operator)
- উপাদান (operand)
- একিক (unary)
- দুয়িক (binary)
- উপাত্ত প্রকারান্তর (type casting)
- চলা-কালীন (run-time)
- নির্বাহ-কালীন (execution-time)
- টীকা দেয়া (commenting)
- টীকা তোলা (uncommenting)
- যোজন (composition)
- নর্থক (void)
- পৃথকী (separator)
- অগ্রগণ্যতা (precedence)
- সহযোজ্যতা (associativity)

অধ্যায় ৭

শর্তালি পরিগণনা (Conditional Programming)

আমাদের জীবনটা নাক বরাবর সোজা একটা পথ নয়, প্রতিটা মোড়ে মোড়ে এটা শাখায় শাখায় বিভক্ত। তোমাকে একটা শাখায় যেতে হবে, একসাথে একের বেশী শাখায় যেতে পারবে না। কোনটায় যাবে তার জন্য ভাবতে হবে, তোমার অবস্থা ও লক্ষ্য বিবেচনা করতে হবে। শর্তালি পরিগণনায় (conditional programming) আমরা শাখায় শাখায় ভাবা শিখবো, আমাদের সামনের গমন পথ বাছাই করা শিখবো, আমরা আমাদের জীবনের সিদ্ধান্ত নেয়া শিখবো।

৭.১ যদি তাহলে নাহলে (If Then Else)

ধরো গণিত পরীক্ষায় তুমি ৫০ বা বেশী পেলে পাশ করবে আর নাহলে করবে ফেল। আর যদি ৮০ বা বেশী পাও তাহলে তুমি তারকা (star) পাবে। এমন একটি ক্রমলেখ (program) তৈরী করো যেটি তোমার গণিতে পাওয়া নম্বর যোগান (input) নিয়ে তোমাকে পাশ না ফেল ফলন (output) দিবে। আর তুমি যদি তারকা পেয়ে থাকো সেটাও জানাবে, তারকা না পেলে কিছু জানাবে না।

ফিরিস্তি ৭.১: পাশ-ফেল-তারকা নম্বর নির্ণয় (Pass Fail Star Marks)

```
int nombor;                         //  চলক ঘোষণা

cout << "nombor? ";                 //  যোগান যাচনা
cin >> nombor;                      //  যোগান নেয়া

if (nombor >= 50)                   //  যদি পাশের নম্বর
  cout << "pash" << endl;           //  পাশ ফলন
else                                //  না হলে
  cout << "fell" << endl;           //  ফেল ফলন

if (nombor >= 80)                   //  যদি তারকা নম্বর
  cout << "taroka" << endl;         //  তারকা ফলন
```

উপরের অংশটুকু কোন ক্রমলেখতে (program) নিয়ে সংকলন (compile) করে চালিয়ে (run) দেখো। যদি ৫০ এর কম কোন নম্বর যোগান (input) দাও যেমন ৪৫ তাহলে ফলন (output) দেখাবে fell। আর যদি ৫০ ও ৭৯ এর মাঝের কোন নম্বর যোগান (input) দাও যেমন ৬৫ তাহলে ফলন (output) দেখাবে pash। আর যদি ৮০ বা বেশী কোন মান যোগান (input) দাও যেমন ৮৫ তাহলে দুই সারি ফলন (output) দেখাবে: প্রথম সারিতে pash আর পরের সারিতে taroka। নীচের যোগান-ফলনে (input-output) এই ক্রমলেখটি (program) তিন বার চালিয়ে বামে, মাঝে, ও ডানে এই তিনটি ব্যাপার দেখানো হয়েছে।

যোগান-ফলন (input-output)

```
nombor? 45          nombor? 65          nombor? 85
fell                pash                pash
                                        taroka
```

এবার ক্রমলেখটি (program) বিশ্লেষণ করি। চলক ঘোষণা (variable declaration), যোগান যাচনা (input prompt), ও যোগান নেয়া (taking input) তো তুমি আগেই শিখেছো। এর পরে খেয়াল করো আমরা লিখেছি if (nombor >= 50) অর্থাৎ যদি নম্বর ৫০ বা তার বেশী হয় তাহলে কী করতে হবে সেটা কিন্তু তার পরপরই বলেছি cout << "pash" << endl; অর্থাৎ পাশ ফলন (output) দেখাতে হবে। তারপরের সারি খেয়াল করো else মানে হলো না হলে অর্থাৎ নম্বর যদি ৫০ বা তার বেশী না হয় মানে ৫০ এর কম হয়, আমাদের ফেল ফলনে দেখাতে হবে যা বলা হয়েছে ঠিক পরের সারিতে cout << "fell" << endl;। যে কোন নম্বর হয় ৫০ এর কম হবে না হয় বেশী বা সমান হবে, এই দুটো ছাড়া আর ভিন্ন কিছু হতে পারে না, এমনকি ওই দুটো একসাথেও সত্যি হতে পারে না। কাজেই আমাদের ক্রমলেখতে (program) হয় cout << "pash" << endl; না হয় cout << "fell" << endl; নির্বাহিত (execute) হবে, দুটোই একসাথে হতে পারবে না। ঠিক যেন দুটো শাখা তৈরী হয়ে গেলো।

আমরা উপরের ক্রমলেখ হতে দেখতে পেলাম প্রাপ্ত নম্বরের ওপর ভিত্তি করে ফলাফল পাশ না ফেল দেখাতে হবে অর্থাৎ ফলন (output) দেখানোর ওই দুটো বিবৃতির মধ্যে কোনটা নির্বাহিত হবে সেটা আমরা নম্বর ৫০ এর কম না বেশী বা সমান এই শর্তটি পরীক্ষা করে বাছাই করতে পারলাম। অনেক পরিগণনা ভাষায় (programming language) (nombor > 50) এর পরে cout << "pash" << endl; এর আগে then লিখতে হয়, কিন্তু c++ এ এটা লিখতে হয় না। এখানে বরং শর্ত nombor >= 50 এটাকে দুটো () বন্ধনী দিয়ে বন্দী করতে হয়। বন্ধনী দেয়ার ব্যাপারটা মনে রাখবে, কারণ প্রথম প্রথম তুমি এটা নিয়ে প্রায়ই ভুল করে সংকলন ত্রুটি (compilation error) পাবে, আর তোমাকে তখন এটি ঠিক করতে হবে। বন্ধনী দুটো এখানে আশে পাশের শর্তকে পৃথক করে, যা সফল সংকলনের (compile) জন্যে জরুরী।

উপরের ক্রমলেখতে (program) খেয়াল করো পাশ ফেল দেখানোর if এর পরে আরো একটা if আছে যেটি দিয়ে আমরা প্রাপ্ত নম্বরটি তারকা (star) কিনা তা দেখাই। এই if এ শর্ত হচ্ছে (nombor >= 80) অর্থাৎ নম্বর যদি ৮০ বা এর বেশী হয় তাহলে ফলন (output) দেখাবে taroka। কিন্তু আর একটু সতর্ক ভাবে খেয়াল করো এই শর্ত মিথ্যা হলে বা পূরণ না হলে কী দেখাবে সেটা কিন্তু নাই। সোজা কথায় এই if এর সাথে কোন else ব্যবহার করা হয় নি। মানে নম্বর যদি ৮০ এর কম হয় তাহলে স্রেফ কিছুই দেখানোর দরকার নাই। তাহলে আমরা জানলাম if এর শর্ত পূরণ হলে আমাদের কী করতে হবে সেটা লিখতে হবে, কিন্তু শর্ত পূরণ না হলে আমরা দরকার মতো কী করতে হবে সেটা লিখবো, অথবা দরকার না হলে কিছুই লিখবো না।

এবার নীচে ক্রমলেখটি খেয়াল করো। এখানে আমরা উপরের পাশ-ফেল দেখানো অংশটিই আবার দেখিয়েছি, তবে একটু ভিন্ন ভাবে। ভিন্নতাটা হলো উপরে যেমন else এর পরে সরাসরি cout << "fell" << endl; লিখেছিলাম, এখানে তা না করে else এর পরে if (nombor

< 50) লিখেছি। তোমাদের কাছে মনে হতে পারে, এটা তো সুন্দর, বুঝতে সুবিধা কারণ ঠিক যেন মানুষের ভাষায় আমরা যে ভাবে বলি যেমন যদি নম্বর 50 বা বেশী হয় ফলন দেখাও পাশ নাহলে যদি নম্বর 50 এর কম হয় ফলন দেখাও ফেল ঠিক তার মতো। কথা সত্য আমাদের বুঝা সুবিধা হয় এ ভাবে। কিন্তু আমরা এভাবে লিখবো না, কারণ else এর পরে ওই if (nombor < 50) লিখা আসলে অদরকারী আর সে কারণে তোমার ক্রমলেখ (program) খামোকা শ্লথ (slow) হয়ে যাবে। ওই if (nombor < 50) লেখাটা অদরকারী কারণটা আগেই খানিকটা জেনেছি তবুও আরেকবার বলি else এর শাখায় আসা মানে হলো nombor >= 50 এই শর্তটি মিথ্যা হয়েছে। আর এই শর্তটি মিথ্যা হওয়া মানে nombor < 50 শর্তটি অবশ্যই সত্য। কাজেই এটি আবার আর একটি if লাগিয়ে পরীক্ষা করার কোন প্রয়োজন নাই।

```
if (nombor >= 50)                  // যদি পাশের নম্বর
  cout << "pash" << endl;          // পাশ ফলন
else if (nombor < 50)              // না হলে
  cout << "fell" << endl;          // ফেল ফলন
```

তুমি যদি একান্তই মানুষের বুঝার সুবিধার্থে ওই if (nombor < 50) টা লিখতে চাও, সেটা টীকার (comment) ভিতরে লিখতে পারো। নীচে যেমন লিখে দেখালাম। এতে তোমার ক্রমলেখও (program) ধীর গতির হলো না, আবার তোমার পক্ষে ক্রমলেখ পড়তেও সহজ হয়ে গেলো। আমরা এ রকমই প্রায়ই করে থাকি। তবে অনেক ক্ষেত্রে else এর পরে ওইরকম একটা if দেওয়া অবশ্যম্ভাবীও হয়ে যায়, এটা আমরা পরের একটা পাঠেই বিস্তারিত দেখবো।

```
if (nombor >= 50)                  // যদি পাশের নম্বর
  cout << "pash" << endl;          // পাশ ফলন
else //if (nombor < 50)            // না হলে
  cout << "fell" << endl;          // ফেল ফলন
```

এই আলোচনার শেষ করি আরেকটা ব্যাপার দিয়ে, সেটা হলো ছাড়ন দেয়া (indentation)। ছাড়ন নিয়ে আগে একবার আমরা আলোচনা করেছিলাম। খেয়াল করো আমরা if (nombor >=50) এর পরে এই শর্ত সত্য হলে যে cout << "pash" << endl; টা নির্বাহ (execute) করতে হবে সেটা পরের সারিতে একটু ভিতরে থেকে লিখতে শুরু করেছি। এটা করলে গণনির (computer) জন্য কিন্তু কোন লাভ বা ক্ষতি নেই, কিন্তু আমরা সহজে চোখে দেখেই কেমন বুঝতে পারি যে ওই cout এর সারিটি আসলে তার আগের সারির if এর সাথের শর্ত সত্য হওয়ার ওপরে নির্ভরশীল। তারপর দেখো পরের সারিতে থাকা else আবার একটু ভিতর থেকে শুরু না হয়ে if বরাবরই শুরু হয়েছে। এটা দিয়ে আমরা বুঝাতে চাই এই else টা আসলে ওই if এর শর্তটা মিথ্যা হলে প্রযোজ্য হবে। লম্বা ক্রমলেখতে যখন অনেক if আর অনেক else থাকবে তখন কোন else কোন if এর সাথে তা মিলানো (match) আমাদের পক্ষে চোখে দেখে কঠিন হয়ে যেতে পারে। ওই মিলানোর সুবিধার্থে if আর তার সাথের else এক বরাবর লিখা হয়। সবশেষে খেয়াল করো else এর পরের সারির cout আবার একটু ভিতর থেকে লেখা, কারণ এটা নির্বাহ হবে কিনা তা নির্ভর করে else এর ওপরে। একটু ভিতর থেকে লেখা শুরু করে সেইটাই বুঝানো হয়।

৭.২ অন্বয়ী অণুক্রিয়া (Relational Operators)

অন্বয়ী অণুক্রিয়া (relational operators) কী? সিপিপি ভাষার ছয়টি অন্বয়ী অণুক্রিয়া >= > == != < <= রয়েছে দুটো রাশির তুলনা করার জন্য। এই অন্বয়ী অণুক্রিয়াগুলো আলোচনা করো।

```
cout << "x y x>=y x>y x==y x!=y x<y x<=y" << endl;

cout << 3 << " " << 4 << "    ";
cout << (3>=4) << "    " << (3>4) << "    ";
cout << (3==4) << "     " << (3!=4) << "     ";
cout << (3<4) << "    " << (3<=4) << endl;

cout << 4 << " " << 4 << "    ";
cout << (4>=4) << "    " << (4>4) << "    ";
cout << (4==4) << "     " << (4!=4) << "     ";
cout << (4<4) << "    " << (4<=4) << endl;

cout << 4 << " " << 3 << "    ";
cout << (4>=3) << "    " << (4>3) << "    ";
cout << (4==3) << "     " << (4!=3) << "     ";
cout << (4<3) << "    " << (4<=3) << endl;
```

উপরের ক্রমলেখতে (program) প্রথমে আমরা দুটো অসমান সংখ্যার তুলনা করেছি যেখানে আগেরটি পরেটি থেকে ছোট। তারপরে আমরা দুটো সমান সংখ্যার তুলনা করেছি। সবশেষে আবারো দুটো অসমান সংখ্যার তুলনা করেছি কিন্তু এখানে আগেরটি বড়, পরেরটি ছোট। উক্ত ক্রমলেখয়ের প্রেক্ষিতে ফলন (output) কী হবে তা নীচে দেখানো হয়েছে।

ফলন (output)

```
x y x>=y x>y x==y x!=y x<y x<=y
3 4   0    0    0     1     1    1
4 4   1    0    1     0     0    1
4 3   1    1    0     1     0    0
```

এখানে ছয়টি অন্বয়ী অণুক্রিয়া (relational operators) ব্যবহার করা হয়েছে। এগুলো হল >= বড় বা বেশী (greater or equal), > বড় (greater), == সমান (equal), != অসমান (unequal), < ছোট (smaller), <= ছোট বা কম (smaller or equal)। একটা বিষয় খেয়াল করো এখানে == সমান চিহ্ন কিন্তু দুটো = চিহ্ন দিয়ে, একটা দিয়ে নয়। আরোপণ (assignment) হলো একটা = দিয়ে। ক্রমলেখ (program) লিখতে গেলে আমাদের প্রায়শই এই ভুলটি হয়ে যায়, আর ক্রমলেখ ঠিকঠাক কাজ করে না। তোমরা এদিকে বিশেষ নজর রাখবে সব সময়।

যাইহোক উপরের ফলনে (output) দেখো, যখনই কোন তুলনার ফলাফল সত্য হয়েছে, ফলনে (output) সেটি এসেছে 1 হিসাবে আর যেখানে তুলনার ফলাফল মিথ্যা তখনই এসেছে 0। আসলে এই অন্বয়ী অণুক্রিয়াগুলো (relational operators) বুলক (Boolean) নামের এক প্রকারের মান ফলাফল হিসাবে দেয়। বুলক যেটাকে সিপিপিতে bool হিসাবে লেখা হয় সেটা হলো এক রকমের উপাত্ত প্রকরণ (data type)। বুলক মান কেবল সত্য আর মিথ্যা হতে পারে। সিপিপিতে মান দুটো হল মিথ্যা 0 ও সত্য 1। ক্রমলেখয়ের (program) ভিতরে অবশ্য মিথ্যা আর সত্যকে 0 আর 1 দিয়ে না বুঝিয়ে আমরা চাইলে স্পষ্টাকারে false ও true দিয়ে বুঝাতে পারি, কিন্তু ফলনে (output) দেখালে ওটা 0 আর 1 হিসাবে দেখানো হয়ে যাবে।

উপরের ক্রমলেখতে যদিও কেবল পূর্ণক (integer) ব্যবহার করা হয়েছে, অন্বয়ী অণুক্রিয়াগুলো (relational operators) আসলে ভগ্নকের (fractioner) সাথেও একই ভাবে কাজ করে।

চাইলে একটা ভগ্নক ও একটা পূর্ণকও ওই অণুক্রিয়াগুলোতে এক সাথে ব্যবহার করা যায়, আর সে-ক্ষেত্রে পূর্ণকটি (int) প্রথমে ভগ্নকে (float) প্রকারান্তরিত (type cast) হয়ে যাবে, তারপর তুলনাটি হবে দুটো ভগ্নক (float) এর মধ্যে। ফলাফল অবশ্যই হবে একটি বুলক (bool) অর্থাৎ 0 বা 1। অন্বয়ী অণুক্রিয়াগুলো (relational operators) বুলক (bool) এর ওপরও কাজ করে। সে-ক্ষেত্রে false আর true কে স্রেফ 0 আর 1 ধরে পূর্ণক হিসাবে বিবেচনা করলে তুলনার যে ফলাফল আসার কথা তাই আসবে। উপরের ক্রমলেখতে তুমি 3 ও 4 এর বদলে নানা রকম বুলক (bool) বা পূর্ণক (int) বা ভগ্নক (float) মান দুটো উপাদানই (operand) একরকম বা দুটো দুইরকম করে বসিয়ে ফলাফলগুলো পর্যবেক্ষণ করতে পারো।

৭.৩ যদি-নাহলে মই (If-Else Ladder)

যদি-নাহলে মই (If-Else Ladder) কী? যদি-নাহলে মই ব্যবহার করে কোন প্রদত্ত বছর অধিবর্ষ (leap year) কিনা তা নির্ণয়ের জন্য একটি ক্রমলেখ (program) তৈরী করো।

প্রথমে আমরা দেখি একটা বছর কখন অধিবর্ষ (leap year) হয়। একটি প্রদত্ত বছর যদি ৪০০ দ্বারা বিভাজ্য হয় তাহলে এটি অধিবর্ষ, যেমন ১৬০০ ও ২০০০। তা নাহলে অর্থাৎ বছরটি যদি ৪০০ দিয়ে বিভাজ্য না হয় কিন্তু এটি যদি ১০০ দিয়ে বিভাজ্য হয় তাহলে এটি অধিবর্ষ নয়, যেমন ১৮০০ ও ১৯০০। তাও নাহলে অর্থাৎ বছরটি ১০০ দ্বারাও বিভাজ্য নয় কিন্তু যদি ৪ দ্বারা বিভাজ্য তাহলে এটি অধিবর্ষ, যেমন ২০১২ বা ২০১৬। তাও নাহলে অর্থাৎ বছরটি যদি ৪ দ্বারা বিভাজ্য না হয় তাহলে এটি অধিবর্ষ নয় অর্থাৎ সাধারণ বর্ষ যেমন ২০১৪ বা ২০১৫। এই কথাগুলোকে সংক্ষেপে লিখলে দাঁড়ায় "যদি ৪০০ দ্বারা বিভাজ্য হয় তাহলে অধিবর্ষ, নাহলে যদি ১০০ দ্বারা বিভাজ্য হয় তাহলে অধিবর্ষ নয়, নাহলে যদি ৪ দ্বারা বিভাজ্য হয় তাহলে অধিবর্ষ, নাহলে অধিবর্ষ নয়।"

ফিরিস্তি ৭.২: অধিবর্ষ নির্ণয় (Leap Year Determination)

```
int bosor;
cout << "bosor koto? ";
cin >> bosor;

if (bosor % 400 == 0) // ৪০০ দিয়ে বিভাজ্য
  cout << "odhiborsho hoy" << endl;
else if (bosor % 100 == 0)  // ১০০ দিয়ে বিভাজ্য
  cout << "odhiborsho noy" << endl;
else if (bosor % 4 == 0)  // ৪ দিয়ে বিভাজ্য
  cout << "odhiborsho hoy" << endl;
else    //  if (bosor % 4 != 0) ৪ দিয়ে বিভাজ্য নয়
  cout << "odhiborsho noy" << endl;

cout << "kee chomotkar!" << endl;
```

এবার আমাদের ক্রমলেখয়ের (program) দিকে তাকাই। উপরে সংক্ষেপে ঠিক যে ভাবে অধিবর্ষ নির্ণয় করার নিয়ম বর্ণনা করেছি, আমাদের ক্রমলেখতে আমরা যেন তাই লিখেছি। মিলিয়ে নাও। পাটিগণিতীয় অণুক্রিয়াগুলোর (arithmetical operators) পাঠ থেকে মনে করো দেখো % অণুক্রিয়া আমাদের ভাগশেষ ফলাফল দেয়। তো ভাগশেষ যদি শুন্য হয় তাহলে আমরা বিভাজ্যতা বুঝতে পারবো, আর ভাগশেষ শুন্য না হলে অবিভাজ্যতা। আমরা প্রথমে ৪০০ দিয়ে

বিভাজ্যতা পরীক্ষা করেছি, না হলে তারপর ১০০ দিয়ে বিভাজ্যতা, তাও নাহলে তারপর ৪ দিয়ে বিভাজ্যতা পরীক্ষা করেছি। কেমন হুবহু একই রকম করে ক্রমলেখটি লেখা গেছে!

খেয়াল করো বিভাজ্য হওয়া bosor %400 == 0 আর অবিভাজ্য হওয়া bosor %400 != 0 এই দুটোতো বিপরীত শর্ত। ক্রমলেখতে (program) প্রথম শর্ত ব্যবহার করলে ওই শর্ত সত্য (অথবা মিথ্যা) হলে যা করতে হবে, একই কাজ দ্বিতীয় শর্ত ব্যবহার করলে সেই শর্ত মিথ্যা (অথবা সত্য) হলে করতে হবে। প্রশ্ন হলো পরস্পর বিপরীত এই দুটোর মধ্যে কোন শর্তটা ব্যবহার করা সুবিধাজনক। তাছাড়া 400 দিয়ে বিভাজ্যতাই বা আগে কেন করবো, 4 বা 100 দিয়ে বিভাজ্যতাও তো আগে করতে পারি? এসবের উত্তর হল ব্যতিক্রম ও বেশী ব্যতিক্রমগুলো তাহলেতে রাখো, আর বাঁদবাকী কম ব্যতিক্রমগুলো সব রাখো নাহলেতে, তাতে চিন্তা করা সহজ হয়ে যায়, ক্রমলেখ (program) তৈরীও সহজ হয়। যেমন ৪০০ দিয়ে বিভাজ্য হলে অধিবর্ষ, এটা অনেক বেশী ব্যতিক্রম, তুলনামুলক অল্প সংখ্যক বছর ৪০০ দিয়ে বিভাজ্য হবে। ১০০ দিয়ে বিভাজ্য হওয়া আর একটু কম ব্যতিক্রম মানে তুলনামুলক ভাবে অনেক বছরই ১০০ দিয়ে বিভাজ্য। ৪ দিয়ে বিভাজ্য হওয়া আরো কম ব্যতিক্রম মানে তুলনামুলক ভাবে অনেক বেশী সংখ্যক বছর ৪ দিয়ে বিভাজ্য। আর ৪ দিয়ে বিভাজ্য না হওয়া মোটামুটি সাধারণ ঘটনা ধরা যায়, বাদবাঁকী সব বছরই ৪ দিয়ে অবিভাজ্য। খেয়াল করো ক্রমলেখ (program) সেভাবেই ব্যতিক্রম মাথায় রেখেই লেখা হয়েছে। সব চেয়ে বেশী ব্যতিক্রমী ব্যাপার সবচেয়ে আগে, সবচেয়ে কম ব্যতিক্রম সবচেয়ে পরে।

আমাদের ক্রমলেখতে ছাড়ন (indentation) দেয়ার ব্যাপারটা একটু খেয়াল করো। যদিও আমরা জানি ছাড়ন দেয়া না দেওয়া অথবা ফাঁকা দিয়ে দিয়ে লেখা বা না লেখাতে ক্রমলেখয়ের (program) ফলাফলে কোন পরিবর্তন হয় না। আমরা কেবল মানুষের বোঝার সুবিধার্থে ওগুলো করি। তারপরও খেয়াল করো আমাদের বুঝার সুবিধার্থে আমরা প্রথমের if, তারপরের else if গুলো, সবশেষের else আর তাদের শর্ত সত্য হলে যা করতে হবে সব মিলিয়ে কী সুন্দর একটা ধাঁচ (pattern) তৈরী করেছি। এই ধাঁচটি একটি মইয়ের মতো কারণ আমাদের প্রথম if দিয়ে শুরু করে শর্ত পরীক্ষা করতে করতে নীচের দিকে নামতে হবে। আর যে কোন একটি শর্ত পূরণ হলেই তার জন্য যে কাজটি করতে হবে পাশের দিকে গিয়ে সেটি করলেই পুরো ধাঁচটির কাজই আসলে শেষ হয়ে যাবে। মানে একটা শর্ত সত্য হলে নীচের দিকের আরো কোন শর্ত আর পরীক্ষা করা হবে না, পুরো ধাঁচের কাজ শেষ হয়ে যাবে। আর ঠিক এর পরে যে বিবৃতি (statement) নির্বাহিত (execute) হবে সেটি হলো এই পুরো ধাঁচের বাইরে থাকা কোন বিবৃতি। যেমন উপরের ক্রমলেখতে লক্ষ্য করো cout << "kee chomotkar!" << endl; হলো পুরো ধাঁচের বাইরে, সুতরাং যদি-নাহলে মই থেকে বের হয়েই ওইটি নির্বাহিত হতে শুরু করবে।

৭.৪ অন্তান্তি যদি-নাহলে (Nested If-Else)

অন্তান্তি যদি-নাহলে (nested if-else) কী? অন্তান্তি যদি-নাহলে ব্যবহার করে কোন প্রদত্ত বছর অধিবর্ষ (leap year) কিনা তা নির্ণয়ের জন্য একটি ক্রমলেখ (program) তৈরী করো।

একটা বছর কখন অধিবর্ষ (leap year) হয়, সেটা আগের পাঠেই জেনেছি, তবুও আরেকবার: বছরটি যদি ৪০০ দ্বারা বিভাজ্য হয় তাহলে এটি অধিবর্ষ, যেমন ১৬০০ ও ২০০০। তা নাহলে অর্থাৎ বছরটি যদি ৪০০ দিয়ে বিভাজ্য না হয় কিন্তু এটি যদি ১০০ দিয়ে বিভাজ্য হয় তাহলে এটি অধিবর্ষ নয়, যেমন ১৮০০ ও ১৯০০। তাও নাহলে অর্থাৎ বছরটি ১০০ দ্বারাও বিভাজ্য নয় কিন্তু যদি ৪ দ্বারা বিভাজ্য তাহলে এটি অধিবর্ষ, যেমন ২০১২ বা ২০১৬। তাও নাহলে অর্থাৎ বছরটি যদি ৪ দ্বারা বিভাজ্য না হয় তাহলে এটি অধিবর্ষ নয় অর্থাৎ সাধারণ বর্ষ যেমন ২০১৪ বা ২০১৫। আগের পাঠে দেখানো আমাদের নীচের ক্রমলেখটি সে ভাবেই যদি-নাহলে মই ব্যবহার করে লেখা।

```
if (bosor % 400 == 0) // ৪০০ দিয়ে বিভাজ্য
```

```
    cout << "odhiborsho hoy" << endl;
  else if (bosor % 100 == 0)   // ১০০ দিয়ে বিভাজ্য
    cout << "odhiborsho noy" << endl;
  else if (bosor % 4 == 0)   // ৪ দিয়ে বিভাজ্য
    cout << "odhiborsho hoy" << endl;
  else     //   if (bosor % 4 != 0) ৪ দিয়ে বিভাজ্য নয়
    cout << "odhiborsho noy" << endl;
```

উপরের ক্রমলেখয়ের (program) দেখো দ্বিতীয় if বিবৃতিতে (statement) ১০০ দিয়ে বিভাজ্যতা পরীক্ষা করা হয়েছে কিন্তু ৪০০ দিয়ে অবিভাজ্য হওয়ার পরে। তো আমরা যদি bosor % 400 == 0 লিখে বিভাজ্যতা পরীক্ষা না করে তার উল্টোটা bosor % 400 != 0 লিখে অবিভাজ্যতা পরীক্ষা করতাম তাহলে ক্রমলেখটি কেমন হতো? তাহলে সালটি যে অধিবর্ষ সেটা দেখানোর cout চলে যেতো else এর সাথে। নীচের ক্রমলেখয়ের সাথে মিলিয়ে নাও।

```
  if (bosor % 400 != 0) // ৪০০ দিয়ে অবিভাজ্য
    if (bosor % 100 == 0) // ১০০ দিয়ে বিভাজ্য
      cout << "odhiborsho noy" << endl;
    else if (bosor % 4 == 0)   // ৪ দিয়ে বিভাজ্য
      cout << "odhiborsho hoy" << endl;
    else     //   if (bosor % 4 != 0) ৪ দিয়ে বিভাজ্য নয়
      cout << "odhiborsho noy" << endl;
  else       // if (bosor % 400 == 0) ৪০০ দিয়ে বিভাজ্য
    cout << "odhiborsho hoy" << endl;
```

তুমি এবার জিজ্ঞেস করতে পারো, আচ্ছা আমি কি একই ভাবে ১০০ বা ৪ দিয়ে বিভাজ্য হওয়ার if গুলোকেও ১০০ বা ৪ দিয়ে অবিভাজ্যতার if দিয়ে লিখতে পারতাম? হ্যাঁ অবশ্যই। নীচের ক্রমলেখ (program) খেয়াল করো। আমরা প্রতিটি if এর শর্তই বদলে এখন অবিভাজ্যতার শর্ত দিয়ে দিয়েছি। যদি-নাহলের মইতে আমরা if এর সাথে থাকা শর্ত মিথ্যা হলে তার else এর পরপরই একটা if দেখতে পেতাম। এখানে দেখো উল্টোটা, if এর শর্ত সত্য হলে বরং তার পরপরই আরেকটা if দেখা যাচ্ছে। এটাকে আমরা বলবো অন্তান্তি যদি-নাহলে (nested if-else) অর্থাৎ একটা যদি-নাহলের ভিতরে আরেকটা যদি-নাহলে, তার ভিতরে আরেকটা!

```
  if (bosor % 400 != 0) // ৪০০ দিয়ে অবিভাজ্য
    if (bosor % 100 != 0) // ১০০ দিয়ে অবিভাজ্য
      if (bosor % 4 != 0) // ৪ দিয়ে অবিভাজ্য
        cout << "odhiborsho noy" << endl;
      else   //   if (bosor % 4 == 0) ৪ দিয়ে বিভাজ্য
        cout << "odhiborsho hoy" << endl;
    else     //   if (bosor % 100 == 0) ১০০ দিয়ে বিভাজ্য
      cout << "odhiborsho noy" << endl;
  else       //   if (bosor % 400 == 0) ৪০০ দিয়ে বিভাজ্য
    cout << "odhiborsho hoy" << endl;
```

অন্তান্তি যদি-নাহলে লেখায় ছাড়ন (indentation) দেয়ার ব্যাপারটি খেয়াল করো। সবচেয়ে ভিতরের else সবচেয়ে ভিতরের if এর সাথে। মাঝের else মাঝের if এর সাথে আর সবচেয়ে

বাইরের else বাইরের if এর সাথে। টীকার (comment) অংশগুলো দেখে মিলিয়ে নাও। ক্রম-লেখ (program) লিখতে ছাড়ন (indentation) দেয়া যে কতটা গুরুত্বপূর্ণ সেটা এখান থেকে তোমার বেশ বুঝতে পারার কথা। ছাড়ন না থাকলে ক্রমলেখ বুঝা আমাদের জন্য দুরহ হবে।

উপরের আলোচনায় একটা জিনিস আমরা দেখেছি: যদি-নাহলে মই (if-else ladder) আর অন্তান্তি যদি-নাহলে (nested if-else) খানিকটা পরস্পরের বিপরীত। তুমি কিন্তু চাইলে এ দুটোর মিশ্রণ ঘটাতে পারো মানে পুরোটাই মই না করে বা পুরোটাই অন্তান্তি না করে দুইরকমটাই ব্যবহার করলে! যেমন ধরো আমরা যদি প্রথমে ১০০ দিয়ে বিভাজ্যতা পরীক্ষা করি। তাহলে শর্ত সত্য হলেই আমরা অধিবর্ষ বলতে পারি না। আমাদের দেখতে হবে ৪০০ দিয়ে বিভাজ্য কিনা। আর ১০০ দিয়ে বিভাজ্য না হলে আমাদের দেখতে হবে ৪ দিয়ে বিভাজ্য কিনা। তো সেই অনুসারে নীচের ক্রমলেখ (program) খেয়াল করো এখানে অন্তান্তি যদি-নাহলেও (nested if-else) আছে আবার যদি-নাহলে মইও (if-else ladder) আছে। ছাড়ন (indentation) দেখে চিনতে পারছো? তুমি কিন্তু আরো নানান ভাবে অধিবর্ষ (leap year) নির্ণয় নিয়ে আর যদি-নাহলে নিয়ে খেলতে পারো। কোন শর্ত আগে, কোনটা পরে, কোনটা মাঝে, কোনটাকে অন্তান্তি করবে, কোনটাকে মইয়ে দিবে, চেষ্টা করে দেখবে, মজাও পাবে, বিষয়গুলো শিখবেও!

```
if (bosor % 100 == 0) // ১০০ দিয়ে বিভাজ্য
  if (bosor % 400 == 0) // ৪০০ দিয়ে বিভাজ্য
      cout << "odhiborsho hoy" << endl;
  else              //  ৪০০ দিয়ে অবিভাজ্য
      cout << "odhiborsho noy" << endl;
else if (bosor % 4 == 0)  // ৪ দিয়ে বিভাজ্য
    cout << "odhiborsho hoy" << endl;
else                //   ৪ দিয়ে অবিভাজ্য
    cout << "odhiborsho noy" << endl;
```

৭.৫ ঝুলন্ত নাহলে (Dangling Else)

ডাক বিভাগ সারাদেশকে অনেক অঞ্চলে ভাগ করে প্রতিটি অঞ্চলের একটা করে ক্রমিক নম্বর দিয়ে দেয়। ঢাকার অঞ্চলগুলোর ক্রমিক নম্বর ১০০ পর্যন্ত, তার মধ্যে ১৩ দিয়ে বিভাজ্য নম্বরগুলো হলো সংরক্ষিত অঞ্চল যেমন ১৩, ২৬, ৩৯, ৫২, ৬৫, ৭৮, ৯১। ঢাকার ভিতর থেকে ডাকে চিঠি পাঠানোর খরচ সারাদেশের যে কোন জায়গায় হলে ৪ টাকা। কিন্তু গন্তব্য ঠিকানা ঢাকার ভিতরেই হলে খরচ ২টাকা, আর ঢাকার ভিতরেই কিন্তু সংরক্ষিত অঞ্চলে হলে খরচ ৩ টাকা। তুমি বেশীর ভাগ সময় ঢাকার ভিতরেই কোথাও না কোথাও চিঠি পাঠাও, তবে মাঝে মাঝে অন্যত্রও পাঠাও। তো তোমাকে একটি ক্রমলেখ (program) লিখতে হবে যেটি তোমার চিঠির গন্তব্য কত নম্বর অঞ্চলে যোগান (input) নিয়ে তোমাকে চিঠি পাঠানোর খরচ ফলনে (output) দেখাবে। তোমার ক্রমলেখতে (program) তুমি অবশ্যই যদি নাহলে (if-else) ব্যবহার করবে কিন্তু তাতে যেন কোন ভাবেই ঝুলন্ত নাহলে (dangling else) দিয়ে ভুল না করে বসো, সেটা খেয়াল রাখবে।

এই ক্রমলেখ (program) লেখা তো খুব সহজ। যদি-নাহলে মই (if-else ladder) ব্যবহার করে তুমি সহজেই লিখে ফেলতে পারো। প্রথমে পরীক্ষা করবে অঞ্চল ১০০ এর চেয়ে বড় কিনা। ১০০ এর বড় হলে খরচ ৪ টাকা, কারণ অঞ্চলটি ঢাকার বাইরে। আর নাহলে মানে অঞ্চলটি ঢাকার ভিতরে হলে এবার পরীক্ষা করে দেখবে ১৩ দিয়ে বিভাজ্য কিনা। ১৩ দিয়ে বিভাজ্য হওয়া মানে সংরক্ষিত অঞ্চল সুতরাং খরচ ৩ টাকা, আর ১৩ দিয়ে বিভাজ্য না হলে মানে অসংরক্ষিত এলাকা হলে খরচ ২ টাকা। নীচের ক্রমলেখয়ের (program) সাথে মিলিয়ে দেখো।

```
int onchol;

cout << "onchol koto? ";
cin >> onchol;

if (onchol > 100)              // ঢাকার বাইরে
  khoroch = 4;
else if (onchol % 13 == 0)     // সংরক্ষিত অঞ্চল
  khoroch = 3;
else                           // অসংরক্ষিত অঞ্চল
  khoroch = 2;
```

এই ক্রমলেখটি আরো নানান ভাবেই লেখা সম্ভব তুমি সেগুলো নিজে নিজে চেষ্টা করবে। তবে আমরা তো কেবল এটি সমাধানই শিখছি না, আমরা শিখবো ঝুলন্ত নাহলে (dangling else) ধাঁচটি কেমন সেটি। তো আমাদের সমস্যার বিবরণে খেয়াল করো একটা কথা আছে তুমি বেশীর ভাগ চিঠিই পাঠাও ঢাকায়। আর সেখানে অসংরক্ষিত এলাকার সংখ্যায় বেশী। এ থেকে আমরা ধরে নিতে পারি যে খরচ বেশীর ভাগ সময়ই ২ টাকা। কাজেই আমরা khoroch চলকটির মান শুরুতেই ২টাকা আদি আরোপণ (initial assignment) করে ফেলতে পারি। তারপর শর্ত পরীক্ষা করে যদি দেখি ঢাকার ভিতরে আর সংরক্ষিত তাহলে খরচ করে দিবো ৩ টাকা আর ঢাকার বাইরে হলে করে দেবো ৪ টাকা। নীচের ক্রমলেখটি দেখো। আমরা সে রকমটি করার চেষ্টা করেছি।

```
int khoroch = 2;  //  ঢাকার ভিতরে অসংরক্ষিত

if (onchol <= 100)  //  ঢাকার ভিতরে
  if (onchol % 13 == 0) //  সংরক্ষিত
    khoroch = 3;
else              // দেখতে মনে হয় ঢাকার বাইরে
  khoroch = 4;
```

উপরের অংশটুকু ব্যবহার করে কোন ক্রমলেখ (program) তৈরী করে চালালে সেটি সঠিক khoroch জানাবে না। ঢাকার বাইরের অঞ্চলগুলোর জন্য যেখানে খরচ ৪টাকা হওয়ার কথা, তা না হয়ে বরং ২টাকাই থাকবে। আর ঢাকার ভিতরের অসংরক্ষিত এলাকার জন্য যেখানে খরচ হওয়ার কথা ২টাকা তা না হয়ে খরচ ৪টাকা হবে। ক্রমলেখ (program) চোখে দেখে তো মনে হচ্ছে সব ঠিক আছে, তবে কেন এই বিপত্তি! আসলে বিপত্তি বাঁধিয়েছে else অংশটি। আমরা যেভাবে ছাড়ন (indentation) দিয়ে লিখেছি তাতে মনে হচ্ছে else অংশটুকু প্রথম if সাথের অর্থাৎ onchol <= 100 মিথ্যা হওয়ার সাথে জড়িত। কিন্তু আসলে তা নয়। প্রতিটি else তার পূর্বের নিকটতম সঙ্গীহীন if এর সাথে জড়িত। তার মানে এইখানে else টি পরের if এর সাথে জড়িত। অর্থাৎ onchol যদি 13 দিয়ে বিভাজ্য না হয় তার সাথে জড়িত।

অন্তান্তি যদি-নাহলে (nested if-else) আলোচনায় আমরা দেখেছিলাম সবচেয়ে ভিতরের else ঠিক সবচেয়ে ভিতরের if এর সাথে, মাঝের else ঠিক মাঝের if এর সাথে, আর বাইরের else ঠিক বাইরের if এর সাথে। আসলে কোন else কোন if এর সাথে যাবে এখানে ছাড়নের (indentation) কোন প্রভাবই নেই। যে else এর জন্য if মিলানো দরকার সেখান থেকে উপরের দিকে যেতে থাকলে প্রথম যে if পাওয়া যাবে যার সাথে কোন ইত্যমধ্যে else দেওয়া হয় নাই, সেই if-ই হলো আমাদের ওই else এর সাথের if। ছাড়ন কেবল আমাদের চোখের দেখার

জন্য, গণনির (computer) কাছে এর কোন অর্থ নেই। তাহলে সঠিকভাবে ছাড়ন দিয়ে লিখলে আমাদের উপরের ক্রমলেখ আসলে নীচের মতো হবে। সুতরাং বুঝতেই পারছো উল্টাপাল্টা ছাড়ন (indentation) দেখে তুমি ভাববে তোমার ক্রমলেখ এরকম কাজ করবে, কিন্তু আসলে সেটা কাজ করবে ভিন্ন রকম। আর ভুলটা কোথায় তা বের করতে তুমি গলদঘর্ম হয়ে যাবে!

```
int khoroch = 2;   //   ঢাকার ভিতরে অসংরক্ষিত

if (onchol <= 100)   //   ঢাকার ভিতরে
  if (onchol % 13 == 0) // সংরক্ষিত
    khoroch = 3;
  else                  // অসংরক্ষিত
    khoroch = 4;
```

এরকমের সমস্যা যেখানে else কার সাথে তা বুঝতে আমাদের ঝামেলা লাগে, সেই সমস্যাকে বলা হয় ঝুলন্ত নাহলে (dangling else)। উপরের সঠিক ছাড়ন (indentation) দিয়ে আমরা বুঝতে পারলাম সমস্যা কোথায় কিন্তু সমাধান কিন্তু আমরা এখনো জানিনা, else কিন্তু আসলেই আমরা বাইরের if এর onchol <= 100 মিথ্যা হলে কী হবে তার জন্য লিখতে চাই। উপায় কী? উপায় খুবই সহজ। ভিতরের if এর জন্য একটা else লাগিয়ে দাও, আর সেই else এর জন্য তো আমাদের কিছু করার নাই। কারণ ওই else এর জন্য খরচ ২টাকা সেটা তো আমরা আগেই আদিমান আরোপণের সময় দিয়ে এসেছি। কিছু করার নাই বুঝাতে আমরা সাধারণত শুন্য বিবৃতি (empty statement) ব্যবহার করি। আর কোন কিছু ছাড়া কেবল দির্তি (semicolon) ; দিয়ে আমরা শুন্য বিবৃতি বুঝাই। এবার তাহলে পরিস্কার হয়ে গেলো কোন else কোন if এর জন্যে।

```
int khoroch = 2;   //   ঢাকার ভিতরে অসংরক্ষিত

if (onchol <= 100)   // ঢাকার ভিতরে
  if (onchol % 13 == 0) //   সংরক্ষিত
    khoroch = 3;
  else                //    অসংরক্ষিত
    ;                 //   শুন্য বিবৃতি
else                  //   ঢাকার বাইরে
  khoroch = 4;
```

৭.৬ যৌগিক বিবৃতি (Compound Statement)

যৌগিক বিবৃতি (compound statement) বলতে কী বুঝো? যৌগিক বিবৃতি ও যদি-নাহলে (if-else) ব্যবহার করে একটি ক্রমলেখ (program) লিখো যেটি প্রথমে দুটি সংখ্যা যোগান (input) নিবে। তারপর প্রথম সংখ্যাটি 0 হলে পরের সংখ্যাটিকে ব্যাসার্ধ ধরে ক্ষেত্রফল ও পরিধি ফলন (output) দিবে। আর প্রথম সংখ্যাটি 1 হলে দ্বিতীয় সংখ্যাটিকে বর্গের এক বাহুর দৈর্ঘ্য ধরে বর্গটির ক্ষেত্রফল ও পরিসীমা ফলন (output) দিবে। প্রথম সংখ্যাটি 0 বা 1 ছাড়া অন্যকিছু হলে দেখাবে "osomorthito akriti" অর্থাৎ এর জন্য আমাদের ক্রমলেখ কাজ করবে না।

```
float nombor1, nombor2;    //   চলক ঘোষণা
```

```
cout << "nombor duti koto? "; //  যোগান যাচনা
cin >> nombor1 >> nombor2;     //  যোগান নেওয়া

if (nombor1 == 0)   //  যদি বৃত্ত হয়
{
  cout << "khetrofol holo: ";
  cout << 3.1416 * nombor2 * nombor2;
  cout << "poridhi holo: ";
  cout << 2 * 3.1416 * nombor2;
  cout << endl;
}
else if nombor == 1)  // যদি বর্গ হয়
{
  cout << "khetrofol holo: ";
  cout << nombor2 * nombor2;
  cout << "porishima holo: ";
  cout << 4 * nombor2;
  cout << endl;
}
else
  cout << "osomorthito akriti" << endl;

cout << "bah hoye gelo." << endl;
```

এই ক্রমলেখটি লেখা খুবই সহজ। কেবল একটাই ঝামেলা আছে সেটা হল যদি-নাহলেতে (if-else) শর্ত সত্য হোক বা মিথ্যা হোক আমাদের একটা বিবৃতির (statement) বদলে একগুচ্ছ বিবৃতি নির্বাহ (execute) করতে হবে। এর আগের সব উদাহরণে আমরা দেখেছি যদি-নাহলে (if-else) শর্ত সত্য বা মিথ্যা হলে কেবল একটা মাত্র বিবৃতি (statement) নির্বাহ করতে। তো ঝামেলাটার সমাধানও আসলে সহজ। একগুচ্ছ বিবৃতিকে { } বাঁকা বন্ধনীর (curly brackets) ভিতরে ঢুকিয়ে দিলেই হলো। এর আগে আমরা জেনেছিলাম দুটো বাঁকা বন্ধনীর (curly brackets) { } দিয়ে আমরা একটা মহল্লা (block) তৈরী করি। তো বাঁকা বন্ধনীর ভিতরে থাকা একগুচ্ছ বিবৃতিকে আমরা বলি যৌগিক বিবৃতি (compound statement)।

যৌগিক বিবৃতি (compound statement) হলেই যে তার ভিতরে একাধিক বিবৃতি থাকতে হবে এমন কথা নেই। মহল্লা তৈরী করে কেবল একটা বিবৃতিও তার ভিতরে লিখতে পারো।

```
if (nombor % 2 == 0)
  { cout << nombor << " holo jor" << endl; }
else
  { cout << nombor << " holo bejor" << endl; }
```

মহল্লা (block) তৈরীর ফলে অনেকসময় ঝুলন্ত নাহলের (dangling else) ঝামেলা সহজে এড়ানো সম্ভব হয়। আগের পাঠের চিঠি পাঠানোর খরচ নির্ণয়ের সমস্যাটি বিবেচনা করো। সেখানে else টি কোন if এর তা নিয়ে ঝামেলা তৈরী হয়েছিল আর আমরা শুন্য বিবৃতি (empty statement) দিয়ে সেটা সমাধান করেছিলাম। শুন্য বিবৃতি হল স্রেফ ; দির্তি (semicolon)

তার আগে কিছু নেই। নীচের ক্রমলেখতে (program) আমরা ওই ভিতরের if টিকে একটি মহল্লার (block) ভিতরে ঢুকিয়ে দিলাম। সুতরাং মহল্লার বাইরে থাকা else টি কোনভাবেই মহল্লার ভিতরের if এর সাথে মিলানো সম্ভব হবে না, কাজেই সেটা আর ঝুলন্ত থাকবে না।

```
int khoroch = 2; // ঢাকার ভিতরে অসংরক্ষিত
if (onchol <= 100)  //  ঢাকার ভিতরে
{
  if (onchol % 13 == 0) // সংরক্ষিত
    khoroch = 3;
}
else            // ঢাকার বাইরে
  khoroch = 4;
```

মহল্লা (block) তৈরী করে চাইলে তার ভিতরে কিন্তু আমরা কোন বিবৃতি একদমই না দিতে পারি অর্থাৎ কেবলই দুটি বাঁকা বন্ধনী (curly brackets) পরপর { }। সেক্ষেত্রে এটাও একরকমের শুন্য বিবৃতি (empty statement) তৈরী হবে। কাজেই শুন্য বিবৃতি তৈরীর দুটো উপায় আমরা শিখলাম একটা হলো কেবলই ; দির্তি (semicolon) দেয়া আরেকটি হলো { } দুটো বাঁকা বন্ধনীর (curly brackets) ভিতরে কিছু না লেখা। প্রথমটির ব্যবহার আগে দেখেছি আর নীচে দ্বিতীয়টি ব্যবহার করে ঝুলন্ত নাহলের (dangling else) আরেকটি সমাধান দেয়া হলো।

```
int khoroch = 2;  //  ঢাকার ভিতরে অসংরক্ষিত

if (onchol <= 100)  // ঢাকার ভিতরে
  if (onchol % 13 == 0) //  সংরক্ষিত
    khoroch = 3;
  else              //   অসংরক্ষিত
    {}              //  শুন্য বিবৃতি
else                //  ঢাকার বাইরে
  khoroch = 4;
```

তাহলে যেখানেই তুমি একটা বিবৃতি (statement) দিতে পারো, সেখানেই তুমি আসলে চাইলে একটা বিবৃতির বদলে একটা যৌগিক বিবৃতিও (compound statement) দিতে পারো, আবার একটা শুন্য বিবৃতিও (empty statement) দিতে পারো। এখন থেকে আমরা যখন বিবৃতি বলবো তখন তুমি সেটা মানে কেবল একটা বিবৃতি বুঝবে না, বরং দরকার মতো সেটা যে যৌগিক বিবৃতিও হতে পারে বা শুন্য বিবৃতিও হতে পারে, তা বুঝে নিবে কেমন!

৭.৭ ত্রুটি শনাক্তকরণ (Error Detection)

একটা দ্বিঘাত সমীকরণ $ax^2 + bx + c = 0$ এর সহগগুলো (coefficient) a, b ও ধ্রুবক (constant) c এর মান যোগান (input) নিয়ে x এর মান দুটি নির্ণয় করার জন্য একটি ক্রমলেখ (program) রচনা করো। এই ক্রমলেখতে তোমাকে সব রকমের নির্বাহকালীন (execution-time) ত্রুটি শনাক্ত (error detect) করে তা ব্যবহারকারীকে জানাতে হবে।

```
#include <cmath>  // বর্গমূলের জন্য sqrt বিপাতক লাগবে
```

```
//  ওপরের অংশ main বিপাতকের আগে, নীচের অংশ ভিতরে

float a, b, c;  //  সহগ ও ধ্রুবক গুলোর জন্য চলক

cout << "somikoron ax^2 + bx + c = 0" << endl;
cout << "a b c er man koto? ";  //  যোগান যাচনা
cin >> a >> b >> c;             //  যোগান নেয়া

float d = b*b - 4*a*c;          //  নিশ্চায়ক

float x1 = (-b + sqrt(d)) / (2*a);  //  প্রথম সমাধান
float x2 = (-b - sqrt(d)) / (2*a);  //  দ্বিতীয় সমাধান

cout << "prothom somadhan x1 = " << x1 << endl;
cout << "ditiyo somadhan x2 = " << x2 << endl;
```

দ্বিঘাত সমীকরণ $ax^2 + bx + c = 0$ এর সহগ ও ধ্রুবকের মান না জেনেও আমরা সমাধানের সূত্র বের করতে পারি $x = (-b \pm \sqrt{b^2 - 4ac})/(2a)$। এই সূত্রের বর্গমূল বের করার জন্য আমাদের cmath শির নথি (header file) থেকে sqrt বিপাতক ব্যবহার করতে হবে। বাদ বাঁকী অংশটুকু সহজ, উপরের ক্রমলেখতে (program) দেখানো হলো। প্রথমে সমীকরণটা দেখানো হয়েছে। খেয়াল করো x^2 দিয়ে আমরা কিন্তু x এর বর্গ বুঝিয়েছি। সহগ ও ধ্রবকগুলোর মান যোগান (input) নেয়ার পরে আমরা b*b - 4*a*c নির্ণয় করে চলক d তে নিয়েছি কারণ এটি দুইটি সমাধানের জন্য দুইবার ব্যবহার করতে হবে। যাইহোক ওপরের অংশটুকু ব্যবহার করে লেখা ক্রমলেখ (program) কাজ করবে যদি সমীকরণটা সহজ সোজা হয়, তাতে কোন ঝামেলা না থাকে! কী রকমের ঝামেলা থাকতে পারে, কিছু অনুমান করতে পারো?

আসলে ক্রমলেখ (program) তৈরীর সময় আমাদের ধরে নিতে হয় যে ব্যবহারকারী সঠিক যোগান (input) যেমন দিতে পারে তেমনি যা ইচ্ছা তা বেঠিক যোগানও দিতে পারে। এইটা সে ভুল করে করতে পারে, না জেনে করতে পারে, ইচ্ছা করেও করতে পারে। তোমার কাজ নষ্ট করে দেয়ার আরো নানাবিধ উদ্দেশ্যও থাকতে পারে। তবে আমরা আপাতত ধরে নিই ব্যবহারকারী ঝামেলা যা করার তা কেবল ওই সহগ ও ধ্রুবকের মান যোগান (input) দেওয়ার মাধ্যমেই করবে। আর ওই ঝামেলাগুলো করলে যা হবে তা হলো উপরের ক্রমলেখ আমাদের নিয়ন্ত্রণের বাইরে নির্বাহকালীন (execution-time) ত্রুটি দেখিয়ে বন্ধ হয়ে (abort) যেতে পারে। এরকম একটা ত্রুটি হলো শুন্য দিয়ে ভাগ (divide by zero), আর একটা ত্রুটি হতে পারে ঋণাত্মক সংখ্যার বর্গমূল বের করা! এই দুটো ত্রুটিই দ্বিঘাত সমীকরণ সমাধানের ক্ষেত্রে ঘটতে পারে।

এই সব ক্ষেত্রে আমাদের আগে থেকে বুঝতে পারতে হবে যে ওই রকম ত্রুটিপূর্ণ ঘটনা ঘটবে কী না, যদি ওইরকম ত্রুটি সত্যিই ঘটে সেটা আমাদের ব্যবহারকারীকে জানাতে হবে। আমাদের তরফে জানানোটা স্বাভাবিক। কিন্তু আমাদের অজান্তে যদি ত্রুটি ঘটে ক্রমলেখ (program) বন্ধ হয়ে যায় তাহলে সেটা কোনভাবেই গ্রহণযোগ্য নয়। সেটা একটা দুর্বল পরিগণনার (programming) উদাহরণ। আর আমরা ত্রুটিটা ঘটবার আগেই ধরতে পারলে সেটা ব্যবহারকারীকে জানিয়ে চাইলে আমাদের ক্রমলেখকে (program) তারপরেও নির্বাহ করা চালিয়ে যেতে দিতে পারবো। তো আমরা নীচে দ্বিঘাত সমীকরণ সমাধানের পুরো ক্রমলেখটি (program) লিখবো আর তাতে সব রকম ত্রুটি ধরে সেটা ব্যবহার কারীকে জানানোর চেষ্টা করবো। আর যখন ত্রুটি হচ্ছে তখন আমরা

return EXIT_SUCCESS; না করে return EXIT_FAILURE; করবো।

ফিরিস্তি ৭.৩: দ্বিঘাত সমীকরণ সমাধান (Solving Quadratic Equations)

```
#include <iostream>    //  cout ব্যবহার করার জন্য
#include <cstdlib>     //  EXIT_SUCCESS/FAILURE এর জন্য
#include <cmath>       //  sqrt বিপাতক ব্যবহার করার জন্য

using namespace std;   //  প্রমিত নামাধার

int main()
{
  float a, b, c;       //  সহগ রাখার জন্য চলক।

  cout << "somikoron ax^2 + bx + c = 0" << endl;
  cout << "a b c er man koto? ";  //  যোগান যাচনা
  cin >> a >> b >> c;             //  যোগান নেওয়া

  // a বা b যদি শুন্য হয় তখন কী হবে? c শুন্য হলে সমস্যা নেই!
  if (a == 0)   // a শুন্য হলে সমীকরণ দ্বিঘাত নয়, একঘাত!
  {
    if (b == 0) // b শুন্য হলে কোন বৈধ সমীকরণই নয়!
    {
      cout << "boidho somikoron noi!" << endl;
      return EXIT_FAILURE;  // ক্রমলেখ বিফল
    }

    //  b শুন্য নয়, কাজেই সমীকরণ কেবলই একঘাত
    cout << "dighat somikoron noi!" << endl;
    cout << "ekghat somikoron dhora holo." << endl;
    cout << "somadhan holo x = " << -c/b << endl;
    return EXIT_SUCCESS; // ক্রমলেখ তবুও সফল ধরা যায়
  }

  // a শুন্য নয়, কাজেই বৈধ দ্বিঘাত সমীকরণ

  float d = b*b - 4*a*c;  //  নিশ্চায়ক হিসাব করো

  if (d < 0)  // ঋণাত্নক নিশ্চায়কের বর্গমূল করা যায় না!
  {
    cout << "nischayok rinatok!" << endl;
    cout << "bastob somadhan nai!" << endl;
    return EXIT_FAILURE;  // ক্রমলেখ বিফল
  }
```

```
    if (d == 0) //  নিশ্চায়ক শুন্য হলে দুটো সমাধান সমান
    {
      cout << "duto somadhan ashole ekoi!" << endl;
      cout << "somapotito x = " << -b/(2*a) << endl;
      return EXIT_SUCCESS;  // ক্রমলেখ সফল
    }

    // দুটো সমাধান আছে, আর তারা অসমান

    float x1 = (-b + sqrt(d)) / (2*a);  // প্রথম সমাধান
    float x2 = (-b - sqrt(d)) / (2*a);  // দ্বিতীয় সমাধান

    cout << "prothom somadhan x1 = " << x1 << endl;
    cout << "ditiyo somadhan x2 = " << x2 << endl;

    return EXIT_SUCCESS;  // ক্রমলেখ সমাধান
}
```

যোগান (input) নেবার পর প্রথমে যেটি আমাদের বিবেচনা করতে হবে তা হলো সমীকরণটি আসলে দ্বিঘাত সমীকরণ কিনা? যদি a শুন্য হয়, তাহলে সমীকরণে কোন x^2 থাকে না, এটি হয়ে যায় $bx+c=0$ যেটি একটি একঘাত সমীকরণ। এমন অবস্থায় আমরা আরো পরীক্ষা করে দেখব bও শুন্য কিনা। যদি b শুন্য হয় তাহলে থাকে কেবল $c=0$, যেখানে কোন চলক (variable) নেই। কাজেই আমাদের ত্রুটি বার্তা দেখিয়ে return EXIT_FAILURE; বলে ফেরত যেতে হবে। নীচের যোগান-ফলন (input-output) খেয়াল করো, a ও b শুন্য হওয়ায় ত্রুটি বার্তা দিয়েছে।

```
somikoron ax^2 + bx + c = 0
a b c er man koto? 0 0 3
boidho somikoron noi!
```

যদি a শুন্য হয় কিন্তু b শুন্য না হয়, তাহলে আমরা if (a == 0) মহল্লার (block) ভিতরেই আছি, কিন্তু b শুন্য না হওয়ায় সমীকরণটি আসলেই একঘাত হয়েছে $bx+c=0$। আমরা কিন্তু দ্বিঘাত সমীকরণ সমাধান করতে চাই, কিন্তু আমাদের দেয়া হয়েছে একটা একঘাত সমীকরণ। তুমি চাইলে এখানে ত্রুটি বার্তা (error message) দেখিয়ে বিফল হয়ে ফেরত যেতে পারো। আবার উদারতা দেখিয়ে নীচের মতো ওই একঘাত সমীকরণের সমাধানই ফলন (output) দিতে পারো। তবে সাথে সতর্ক বার্তাও (warning message) দিয়ে দিলে যে এটা দ্বিঘাত সমীকরণ নয়!

```
somikoron ax^2 + bx + c = 0
a b c er man koto? 0 2 1
dighat somikoron noi!
ekghat somikoron dhora holo.
somadhan holo x = -0.5
```

যদি a শুন্য না হয় তাহলে এটা একটা বৈধ দ্বিঘাত সমীকরণ। সুতরাং প্রথমে আমরা নিশ্চায়ক (discriminant) নির্ণয় করে একটা চলকে নেবো। উপরের ক্রমলেখতে খেয়াল করো float d = b*b - 4*a*c; লিখে তাই করা হয়েছে। এখন নিশ্চায়ক যদি ঋণাত্মক (negative) হয়

তাহলে তো বর্গমূল নির্ণয় করা সম্ভব না, কিন্তু দ্বিঘাত সমীকরণের সমাধানের সূত্রে নিশ্চায়কের বর্গমূল আমাদের দরকার। কাজেই নিশ্চায়কের মান ঋণাত্মক হলে আমাদের পক্ষে সমাধান করা সম্ভব নয়। একটি ত্রুটি বার্তা (error message) দেখিয়ে return EXIT_FAILURE; ফেরত যাওয়া উচিৎ। নীচের যোগান-ফলন (input-output) খেয়াল করো, ঠিক তাই ঘটেছে।

```
somikoron ax^2 + bx + c = 0
a b c er man koto? 2 -5 2
nischayok rinatok!
bastob somadhan nai!
```

নিশ্চায়ক (discriminant) যদি ঋণাত্মক (negative) না হয়, তাহলে এবার দেখতে হবে এটি শুন্য কিনা। কারণ শুন্য হলে সেক্ষেত্রে আমাদের সমাধান দুটিই হবে, কিন্তু সমাধান দুটি আবার আলাদা আলাদা না হয়ে একই হবে। এইরকম অবস্থাকে বলা হয় সমাপতিত (coincidental) সমাধান। নীচের যোগান-ফলনে (input-output) এটি দেখানো হলো।

```
somikoron ax^2 + bx + c = 0
a b c er man koto? 1 -2 1
duto somadhan ashole ekoi!
somapotito x = 1
```

সবশেষের যে অবস্থা সেটি হলো নিশ্চায়ক ঋণাত্মকও নয়, শুন্যও নয়, তাহলে সেটি ধনাত্মক (positive)। আর এটিই হলো সেই অবস্থা আমরা যেটি ধরে নিয়ে একদম শুরুতে একটা ছোট ক্রমলেখ (program) দেখিয়েছিলাম। কাজেই আমরা সেই কাজটুকু করে দুটো সমাধান আমাদের জানা সূত্রানুযায়ী নির্ণয় করে ফলন দেখিয়ে return EXIT_SUCCESS; করে ক্রমলেখ শেষ করবো। নীচের যোগান-ফলনে (input-output) এই অবস্থা দেখানো হলো।

```
somikoron ax^2 + bx + c = 0
a b c er man koto? 2 -5 2
prothom somadhan x1 = 2
ditiyo somadhan x2 = 0.5
```

উপরের বিস্তারিত ক্রমলেখতে (program) খেয়াল করো প্রতিটি if এর শর্ত সত্য হলে যে মহল্লাটি (block) নির্বাহিত হবে সেই মহল্লা শেষ হয়েছে একটি return দিয়ে। তার মানে ওই শর্তগুলো সত্য হলে ক্রমলেখয়ের (program) নীচের কোন অংশ আর নির্বাহিত হবে না। আর একারণে সংশ্লিষ্ট if এর শর্ত মিথ্যা হলে যা হবে সেটি আর আমরা একটি else লিখে তারপর আরেকটি মহল্লায় (block) ঢুকিয়ে দেই নি। কারণ if এর শর্ত সত্য হলে যে মহল্লা (block) তার বাইরে পুরোটাইতো এখন else এর জন্য, কাজেই আলাদা করে মহল্লা করার দরকার নেই।

৭.৮ বুলক সংযোজক (Boolean Connectives)

একটি ক্রমলেখ (program) রচনা করো যেটি একটি পূর্ণক (integer) যোগান (input) নিবে। তারপর সংখ্যাটি যদি ৭ ও ১৩ উভয় দ্বারা বিভাজ্য হয় তাহলে ফলন (output) দিবে "মিশ্রভাগ্য সংখ্যা", যদি ৭ দ্বারাও বিভাজ্য না হয় আবার ১৩ দ্বারাও বিভাজ্য না হয় তাহলে ফলন দিবে "অভাগ্য সংখ্যা", যদি কেবল ৭ দ্বারা বিভাজ্য হয় কিন্তু ১৩ দ্বারা বিভাজ্য নয় তাহলে ফলন দিবে "সৌভাগ্য সংখ্যা", আর যদি কেবল ১৩ দ্বারা বিভাজ্য হয় কিন্তু ৭ দ্বারা নয় তাহলে ফলন দিবে "দুর্ভাগ্য সংখ্যা"।

যদি সংখ্যাটি ৭ বা ১৩ যে কোন একটি বা উভয়টি দ্বারা বিভাজ্য হয় তাহলে ফলন দিবে "ভাগ্য সংখ্যা"। একটি সংখ্যা একই সাথে উপরের এক বা একাধিক ভাগে পড়তেই পারে।

ফিরিস্তি ৭.৪: সৌভাগ্য ও দুর্ভাগ্যের সংখ্যা (Lucky & Unlucky Numbers)

```
int nombor;
cout << "sonkhya koto? ";
cin >> sonkhya;

if (sonkhya % 7 == 0 && sonkhya % 13 == 0)
  cout << "missrovaggo sonkhya" << endl;

if (sonkhya % 7 != 0 && sonkhya % 13 != 0)
  cout << "ovaggo sonkhya" << endl;

if (sonkhya % 7 == 0 && sonkhya % 13 != 0)
  cout << "souvaggo sonkhya" << endl;

if (sonkhya % 13 == 0 && sonkhya % 7 != 0)
  cout << "durvaggo sonkhya" << endl;

if (sonkhya % 7 == 0 || sonkhya % 13 == 0)
  cout << "vaggo sonkhya" << endl;
```

উপরের ক্রমলেখতে (program) && হলো "এবং" আর || হলো "অথবা"। তুমি চাইলে সিপিপিতে && এর বদলে and আর || এর বদলে or লিখতে পারো। আর বাংলায় কখনো কখনো আমরা "এবং" এর বদলে "ও" বা "আর" লিখবো, আর "অথবা" এর বদলে লিখবো "বা"। যাই হোক মনে রাখো && এর ফলাফল সত্য হয় যখন এর দুপাশের উপাদানই (operand) সত্য হয়, আর যেকোন একটা মিথ্যা হলেই ফলাফল মিথ্যা। অন্যদিকে || এর ফলাফল মিথ্যা হয় যখন এর দুপাশের উপাদানই (operand) মিথ্যা, আর যে কোন একটি সত্য হলেই ফলাফল সত্য। তো উপরের ক্রমলেখ বুঝার চেষ্টা করো। খুব কঠিন কিছু নয়। সমস্যাটি ঠিক যেমন করে বাংলায় বর্ণনা করা হয়েছে, ক্রমলেখতেও যেন ঠিক সে রকম করেই লেখা হয়েছে।

এবার ওই ক্রমলেখকে আমরা কিছু উন্নয়নের চেষ্টা করি। একটা বিষয় খেয়াল করো বিভাজ্য হওয়া বা বিভাজ্য না হওয়া আমরা বারবার হিসাব করেছি। এইটা তো হওয়া উচিৎ নয়। তাছাড়া ভাগশেষ বের করা অন্যন্য অনেক অণুক্রিয়া (operator) তুলনায় মোটামুটি সময় সাপেক্ষ কাজ। আমাদের তাই একবার ভাগশেষ হিসাব করে সেটাই বারবার ব্যবহার করা উচিৎ। তো সেই অনুযায়ী আমরা ক্রমলেখতে কিছু পরিবর্তন করতে পারি। মুলত ভাগশেষের জন্য আমাদের দুটো পূর্ণক (integer) চলক নিতে হবে int vagshesh7 = sonkhya % 7; আর int vagshesh13 = sonkhya % 13; আর তারপর প্রতিটি sonkhya % 7 এর বদলে vagshesh7 এবং প্রতিটি sonkhya % 13 এর বদলে vagshesh13 লিখতে হবে। খুবই সহজ পরিবর্তন।

কিন্তু আমরা আসলে এই পরিবর্তনের কথা বলছি না। আমরা ভাগশেষগুলো পূর্ণক (integer) চলকে না রেখে বরং সংখ্যাটি ৭ বা ১৩ দ্বারা বিভাজ্য কিনা তার সত্যতা বুলক (boolean) চলকে রাখতে চাই। এক্ষেত্রে আমরা bool bivajyo7 = sonkhya % 7 == 0; লিখবো। তাতে সংখ্যাটি ৭ দ্বারা বিভাজ্য হলে bivajyo7 চলকের মান হবে true বা 1 আর ৭ দ্বারা বিভাজ্য না হলে ওই চলকের মান হবে false বা 0। একই ভাবে bool bivajyo13 = sonkhya % 13 == 0; লিখ-

লে bivajyo13 এর মান হবে true বা 1 যদি সংখ্যাটি 13 দ্বারা বিভাজ্য হয় আর মান হবে false বা 0 যদি সেটি 13 দ্বারা বিভাজ্য না হয়। নীচে এই ক্রমলেখ দেখানো হলো।

```
int nombor;
cout << "sonkhya koto? ";
cin >> sonkhya;

bool bivajyo7 = sonkhya % 7 == 0;
bool bivajyo13 = sonkhya % 13 == 0;

if (bivajyo7 && bivajyo13)
  cout << "missrovaggo sonkhya" << endl;

if (!bivajyo7 && !bivajyo13)
  cout << "ovaggo sonkhya" << endl;

if (bivajyo7 && !bivajyo13)
  cout << "souvaggo sonkhya" << endl;

if (bivajyo13 && !bivajyo7)
  cout << "durvaggo sonkhya" << endl;

if (bivajyo7 || bivajyo13)
  cout << "vaggo sonkhya" << endl;
```

উপরের ক্রমলেখতে (program) খেয়াল করো bivajyo7 (বা একই ভাবে bivajyo13) এর মান সত্য নাকি মিথ্যা আমরা কিন্তু bivajyo7 == true অথবা bivajyo7 == 1 লিখে করি নাই, যদিও তা করতে পারতাম। আমরা বরং স্রেফ চলকটা ব্যবহার করেছি কারণ চলকটার মানই তো সরাসরি সত্য বা মিথ্যা। আবার আলাদা করে == অণুক্রিয়া (operator) দিয়ে সত্য বা মিথ্যা পরীক্ষা করার দরকার নেই। তবে খেয়াল করো যখন বিভাজ্য নয় পরীক্ষা করতে হবে তখন আমরা !bivajyo7 (বা একই ভাবে !bivajyo13) লিখে অর্থাৎ চলকের নামের সামনে ! লাগিয়ে দিয়েছি। এখানে ! হলো নয় বা না অণুক্রিয়া। তুমি চাইলে ! এর বদলে সিপিপিতে not লিখতে পারতে। নয় অণুক্রিয়া সত্যকে উপাদান (operand) হিসাবে নিয়ে মিথ্যা ফলাফল দেয় আর মিথ্যাকে উপাদান হিসাবে নিয়ে সত্য ফলাফল দেয়। আর সে কারণে bivajyo7 == false না লিখে আমরা !bivajyo7 লিখলেই আমাদের কাজ হয়ে যায়।

৭.৯ বুলক, পূর্ণক, ভগ্নক (Boolean, Integer, Float)

একটি সংখ্যা জোড় না বিজোড় তা নির্ণয়ের ক্রমলেখ (program) রচনা করো। তোমার ক্রমলেখতে তুমি কোন বুলক চলক (boolean variable) বা অন্বয়ী অণুক্রিয়া (relational operator) ব্যবহার করতে পারবে না। তোমাকে পূর্ণক মানকেই বুলক হিসাবে ব্যবহার করতে হবে।

```
int sonkhya = 41;    // তুমি চাইলে যোগান নিতে পারো।
```

```
if (sonkhya % 2 != 0)
  cout << "bejor" << endl;
else
  cout << "jor" << endl;
```

এই ক্রমলেখটি (program) তুমি চাইলে উপরের মতো করে লিখতে পারো। কোন সংখ্যা ২ দিয়ে ভাগ দিলে যদি ভাগশেষ শুন্য না হয় তাহলে সংখ্যাটি বিজোড়, আর ভাগশেষ শুন্য হলে সংখ্যাটি জোড়। কাজেই ক্রমলেখটি সহজেই লিখে ফেলা যায়। কিন্তু এতে অসমান নির্ণয়ের জন্য একটি অণুক্রিয়া != ব্যবহার করতে হচ্ছে, যেটি চাইলে আমরা ব্যবহার না করেও কাজ চালাতে পারি। এর কারণ হলো যে কোন সময় শুন্যকে আমরা মিথ্যা ধরে নিতে পারি আর যে কোন অশুন্য মানকে, সেটা ধনাত্নক হোক বা ঋণাত্নক হোক, আমরা সেটাকে সত্য ধরে নিতে পারি। তাতে আমাদের মানটি আলাদা করি শুন্য কিনা তা আর পরীক্ষা করার দরকার পড়ে না। কাজেই নীচের ক্রমলেখয়ের (program) মতো করে != বাদ দিয়ে স্রেফ if (sonkhya % 2) লেখা মানেই হলো if (sonkhya % 2 != 0) লেখা।

```
int sonkhya = 41;    // তুমি চাইলে যোগান নিতে পারো।

if (sonkhya % 2)     // ভাগশেষ অশুন্য হলে
  cout << "bejor" << endl;
else                 // ভাগশেষ শুন্য হলে
  cout << "jor" << endl;
```

উপরের উদাহরণে আমরা কেবল পূর্ণক ব্যবহার করেছি। কিন্তু ভগ্নক সংখ্যার ক্ষেত্রেও একই কথা প্রযোজ্য। যেমন নীচের ক্রমলেখতে ভগ্নক সংখ্যাটি শুন্য কিনা তা নির্ণয় করা হয়েছে। খেয়াল করে দেখো আমরা সংখ্যাটিকেই সরাসরি শর্ত হিসাবে ব্যবহার করেছি। সংখ্যাটি শুন্য না হলেই এটি সত্য হবে shunyo noi ফলন আসবে, আর সংখ্যাটি শুন্য হলে ফলন আসবে shunyo hoi।

```
float sonkhya = -3.5;    // তুমি চাইলে যোগান নিতে পারো।

if (sonkhya)     // সংখ্যাটি অশুন্য হলে
  cout << "shunyo noi" << endl;
else             // সংখ্যাটি শুন্য হলে
  cout << "shunyo hoi" << endl;
```

তাহলে এখানকার আলোচনায় আমরা দেখলাম উপাদান (operand) হিসাবে শুন্য হলো মিথ্যা (false) আর অন্য যেকোন ধনাত্নক (positive) বা ঋণাত্নক (negative) পূর্ণক (integer) বা ভগ্নক (float) হলো সত্য (true)। আর অন্বয়ী অণুক্রিয়া (relational operators) আলোচনার সময় জেনেছি ফলাফল (result) হিসাবে সবসময় false হলো 0 এবং true হলো 1। খেয়াল করো উপাদান (operand) হিসাবে true 0 ছাড়া যে কোন কিছু হলেও ফলাফল (result) হিসাবে true কেবল 1, false অবশ্য উভয় ক্ষেত্রেই কেবল 0।

৭.১০ বুলক বীজগণিত (Boolean Algebra)

দক্ষতার সাথে যদি নাহলে (if else) ব্যবহার করতে চাইলে আমাদের খানিকটা বুলক বীজগণিত (boolean algebra) জানা দরকার। অনেক সময় এবং &&, অথবা ||, নয় ! অণুক্রিয়া

(operators) বেশ কয়েকবার করে নিয়ে তুমি হয়তো জটিল একটা রাশি (expression) তৈরী করবে যেটা সরাসরি মূল্যায়ন (evaluate) করতে গেলে প্রত্যকটি অণুক্রিয়া (operator) ধাপে ধাপে মূল্যায়ন করতে হবে। কিন্তু বুলক বীজগণিত ব্যবহার করে সেটা হয়তো সরলীকরণ করে ছোট করে অনেক কম অণুক্রিয়া (operator) দিয়েই প্রকাশ করা সম্ভব। অণুক্রিয়ার সংখ্যা কম হওয়া মানে সেটা দক্ষ হবে, ক্রমলেখ নির্বাহ (program execution) করতে সময় কম লাগবে।

বুলক বীজগণিতে (boolean algebra) সত্যকে প্রমূর্তায়ন (representation) করা হয় true বা 1 দিয়ে আর মিথ্যাকে করা হয় false বা 0 দিয়ে। সিপিপিতে অণুক্রিয়াসমুহ (operator) ফলাফলের ক্ষেত্রে একদমএইরূপ প্রমূর্তায়নই (representation) মেনে চলে, তবে উপাদানের (operand) ক্ষেত্রে কিছুটা উদার হয়ে 0 ছাড়া যেকোন মানকেই true হিসাবে ধরে নেয়, false ধরে নেয় যথারীতি কেবল 0 কে। উপাদান ও ফলাফলের ক্ষেত্রে true এর এই ভিন্নতা মনে রাখবে। অনেক সময় এটি সুবিধাজনক, আবার অনেক সময় এটি অনেক ত্রুটির (error) জন্মদেয়।

বুলক বীজগণিতের (boolean algebra) প্রথম যে অণুক্রিয়া (operator) তাহলো নয়, না যেটা ! বা not লিখে প্রকাশ করা হয়। নয় অণুক্রিয়ার উপাদান (operand) ও ফলাফল (result) নীচে খেয়াল করো !true হলো false আর !false হলো true। আমরা এখানে ≡ বা সমতুল (equivalence) প্রতীক ব্যবহার করে বুঝাবো যে ওই প্রতীকের বাম ও ডানপাশ সমতুল।

- !true ≡ false
- !false ≡ true

ধরো দুটো ! পরপর আছে যেমন !!true বা !!false বা !!x, তাহলে ফলাফল কী হবে। এইসব ক্ষেত্রে আমাদের ডানের ! আগে হিসাব করতে হবে, তার ওপর বামের ! ধরে শেষ ফলাফল হিসাব করতে হবে। একারণে ! হলো ডান সহযোজ্য (right associative)। তো এখানে ডানের ! অণুক্রিয়া true বা false কে উল্টে দিবে আর বামের ! সেটাকে আবার সিধা করবে। সুতরাং !!true হবে true, !!false হবে false, আর !!x হবে x। বুলক বীজগণিতে এই বিধিকে বলা হয় দুনো ঋণায়ন (double negation)। তুমি কি তিন বা বেশী সংখ্যক ! পরপর থাকলে কী হবে বের করতে পারবে? অবশ্যই পারবে, প্রতি দুইটি ! পরস্পরকে বাতিল করে দিবে।

- !!x ≡ !(!x) ডান সহযোজ্য
- !!true ≡ true দুনো ঋণায়ন
- !!x ≡ x দুনো ঋণায়ন
- !!false ≡ false দুনো ঋণায়ন

বুলক বীজগণিতের (boolean algebra) দ্বিতীয় অণুক্রিয়া (operator) এবং, ও যেটা && বা and লিখে প্রকাশ করা হয়। লক্ষ্য করো এবং অণুক্রিয়ার ফলাফল (result) সত্য যখন উভয় উপাদানই (operand) সত্য, আর যেকোন একটি উপাদান মিথ্যা হলেই ফলাফল মিথ্যা।

- true && true ≡ true
- false && true ≡ false
- true && false ≡ false
- false && false ≡ false

একটি উপাদান সত্য বা মিথ্যা ধরে নিলে এবং && অণুক্রিয়ার জন্যে আমরা বেশ কিছু সরলীকরণ করে ফেলতে পারি যেগুলোকে আমরা সত্যের সরল (true simplification) ও মিথ্যার সরল (false simplification) বলবো। কোন একটি উপাদান আমরা যদি আগেই বুঝে ফেলি সেটি সত্য না মিথ্যা তাহলে আমরা এই সরলীকরণগুলো কাজে লাগাতে পারবো।

- x && true ≡ x সত্যের সরল
- x && false ≡ false মিথ্যার সরল
- true && x ≡ x সত্যের সরল
- false && x ≡ false মিথ্যার সরল

বুলক বীজগণিতের (boolean algebra) তৃতীয় অণুক্রিয়া (operator) অথবা, বা যেটা || বা or লিখে প্রকাশ করা হয়। লক্ষ্য করো অথবা অণুক্রিয়ার ফলাফল (result) মিথ্যা যখন উভয় উপাদানই (operand) মিথ্যা, আর যেকোন একটি উপাদান সত্য হলেই ফলাফল সত্য।

- true || true ≡ true
- false || true ≡ true
- true || false ≡ true
- false || false ≡ false

একটি উপাদান সত্য বা মিথ্যা ধরে নিলে অথবা || অণুক্রিয়ার জন্যে আমরা বেশ কিছু সরলীকরণ করে ফেলতে পারি যেগুলোকে আমরা সত্যের সরল (true simplification) ও মিথ্যার সরল (false simplification) বলবো। কোন একটি উপাদান আমরা যদি আগেই বুঝে ফেলি সেটি সত্য না মিথ্যা তাহলে আমরা এই সরলীকরণগুলো কাজে লাগাতে পারবো।

- x || true ≡ true সত্যের সরল
- x || false ≡ x মিথ্যার সরল
- true || x ≡ true সত্যের সরল
- false || x ≡ x মিথ্যার সরল

বুলক বীজগণিতে অণুক্রিয়াগুলোর (operator) অগ্রগণ্যতার ক্রম (precedence order) হলো প্রথমে নয় !, তারপর এবং &&, আর শেষে অথবা ||, এই ক্রমের অন্যথা করতে চাইলে প্রয়োজনে বন্ধনী ব্যবহার করতে হবে। তাছাড়া দ্ব্যর্থবোধকতা এড়াতে বন্ধনী ব্যবহার করা উচিৎ। নীচের উদাহরণ দুটি খেয়াল করো। প্রথমটিতে আগে ! তারপর &&, শেষে || করতে হবে, বন্ধনী ব্যবহার করে সেটাই বুঝানো হয়েছে। দ্বিতীয় উদাহরণটিতে ! আগে করলেও && আগে || করায় সেটা সঠিক হয় নি। এখানে ≢ দিয়ে বুঝানো হয়েছে দুইপাশ পরস্পরের সমতুল নয়।

- x && !y || z ≡ (x && (!y)) || z আগে !, মাঝে &&, পরে ||
- x && !y || z ≢ x && ((!y) || z) আগে !, মাঝে || নয়, পরেও && নয়

গাণিতিক অণুক্রিয়া (mathematical operators) ও আরোপণের (assignment) সাথে যদি যৌক্তিক অণুক্রিয়াগুলো মিলিয়ে দেখা হয় তাহলে সব মিলিয়ে অগ্রগণ্যতার ক্রম নিম্নরূপ:

১. ++ −− একিক উত্তর (unary postfix) বাম থেকে ডানে (left associative)

২. ++ −− + − ! একিক পূর্ব (unary prefix) ডান থেকে বামে (right associative)

৩. ∗ / % দুয়িক (binary) বাম থেকে ডানে (left associative)

৪. + − দুয়িক (binary) বাম থেকে ডানে (left associative)

৫. && দুয়িক (binary) বাম থেকে ডানে (left associative)

৬. || দুয়িক (binary) বাম থেকে ডানে (left associative)

৭. = += −= ∗= /= %= দুয়িক (binary) ডান থেকে বামে (right associative)

৮. , দুয়িক (binary) বাম থেকে ডানে (left associative)

৭.১১ বুলক সমতুল (Boolean Equivalence)

এবার আমরা বেশ কিছু সমতুল বিধি (equivalence law) দেখবো। এই বিধিগুলোর বামপাশ আর ডানপাশ সবসময় সমতুল। আমরা তাই এগুলো ব্যবহার করে বিভিন্ন সময়ে আমাদের যৌক্তিক রাশি (logical expression) সরল করার চেষ্টা করবো।

নীচের দুটো বিধি হলো এবং, অথবার বিনিময় বিধি (commutative law)। বিনিময় বিধিতে অণুক্রিয়ার (operator) উপাদানগুলো পার্শ পরিবর্তন করলেও ফলাফল একই থাকে।

- x && y ≡ y && x বিনিময়
- x || y ≡ y || x বিনিময়

নীচের দুটো বিধি হলো সহযোজন বিধি (associative law)। এই বিধিতে একই অণুক্রিয়া (operator) পরপর থাকলে আমরা যে কোনটি আগে মূল্যায়ন (evaluate) করে তার ফলাফলের সাথে অন্য অণুক্রিয়ার মূল্যায়ন করতে পারি, আর তাতে ফলাফল একই হবে।

- x && y && z ≡ (x && y) && z ≡ x && (y && z) সহযোজ্য
- x || y || z ≡ (x || y) || z ≡ x || (y || z) সহযোজ্য

নীচের দুটো বিধি হলো বন্টন বিধি (distributive law)। এই বিধিতে দুটি ভিন্ন অণুক্রিয়া (operator) পরপর থাকলে আমরা একটিকে আরেকটির ওপর বন্টন করে দিতে পারি। পাটিগণিতে বন্টন বিধির উদাহরণ হলো $x * (y + z) = x * y + x * z$।

- x && y || z ≢ x && (y || z) ≡ (x && y) || (x && z) বন্টন
- x || y && z ≢ (x || y) && z ≡ (x && z) || (y && z) বন্টন
- x || y && z ≡ x || (y && z) ≡ (x || y) && (x || z) বন্টন
- x && y || z ≡ (x && y) || z) ≡ (x || z) && (y || z) বন্টন

নীচের বিধিগুলো হলো শোষন বিধি (absorption law)। প্রথম চারটি বিধিতে খেয়াল করো x যদি true হয় তাহলে x || y বা y || x এর মানও true আর ফলে && এর ফলাফলও true। আবার x যদি false হয় তাহলে && এর ফলাফল অবশ্যই false। তাহলে বামদিকের রাশিগুলোর মান সবসময় x এর মান যা তাই। একই ভাবে শেষের চারটি বিধিতে খেয়াল করো x যদি false হয় তাহলে x && y বা y && x এর মানও false। আবার x যদি true হয় তাহলে || এর ফলাফল অবশ্যই true। তাহলে বামদিকের রাশিগুলোর মান সব সময় x এর মান যা তাই। কাজেই এই বিধিগুলো তোমাকরে বুলক রাশিকে কত সহজ ও ছোট করে ফেলে!

- x && (x || y) ≡ x শোষন
- x || (x && y) ≡ x শোষন
- x && (y || x) ≡ x শোষন
- x || (y && x) ≡ x শোষন
- (x || y) && x ≡ x শোষন
- (x && y) || x) ≡ x শোষন
- (y || x) && x ≡ x শোষন
- (y && x) || x) ≡ x শোষন

নীচের বিধি দুটোতে অণুক্রিয়াগুলোর (operator) উপাদানদুটো একই। এবং && ও অথবা || উভয়ের ফলাফল এক্ষেত্রে সবসময় উপাদানটির মান যা তাই হবে। একটি উপাদানের নিজের

সাথে নিজের ওপর কোন অণুক্রিয়া (operator) প্রযুক্ত হলে ফলাফল যদি উপাদানটিই হয় তাহলে অণুক্রিয়াটির এই ধর্মকে বলা হয় অস্বক্রিয়তা (idempotence)। সব অণুক্রিয়াই কিন্তু অস্বক্রিয় নয়, যেমন পাটিগণিতে সর্বাবস্থায় $x + x = x$ সত্য নয়, কাজেই যোগ $+$ অস্বক্রিয় নয়। বুলক বীজগণিতে এবং && ও অথবা || উভয়েই অস্বক্রিয় (idempotent)।

- x && x ≡ x অস্বক্রিয়তা
- x || x ≡ x অস্বক্রিয়তা

v

নীচের বিধি দুটোতে অণুক্রিয়াগুলোর (operator) উপাদানদুটো পরস্পরের বিপরীত। এবং && এর ফলাফল এক্ষেত্রে সবসময় false হবে, কারণ দুটো উপাদানের মধ্যে যে কোন একটি তো মিথ্যা হবেই, আর যে কোন একটি মিথ্যা হলেই এবং এর ফলাফল মিথ্যা। তাই এই বিধিকে বলা হয় অসঙ্গতি (contradiction)। আর অথবা || এর ফলাফল এক্ষেত্রে সবসময় true হবে, কারণ দুটো উপাদানের মধ্যে যে কোন একটি তো সত্য হবেই, আর যে কোন একটি সত্য হলেই অথবা এর ফলাফল সত্য। তাই এই বিধিকে বলা হয় নঞ মধ্যম (excluded middle)।

- x && !x ≡ false অসঙ্গতি
- x || !x ≡ true নঞ মধ্যম

নীচের বিধি দুটোর নাম ডি মরগানের বিধি (De Morgan's Law)। এই বিধিদুটো খুবই গুরুত্বপূর্ণ এবং প্রায়শই বুলক রাশির সরলীকরণে ব্যবহৃত হয়। এই বিধি অণুযায়ী এবং && এর ফলাফলের ওপর নয় ! করলে যে ফলাফল পাওয়া যায় তা আগে উপাদানগুলোর ওপরে নয় ! করে সেই ফলাফলের ওপর অথবা || চালিয়ে পাওয়া ফলাফলের সমতুল। একই ভাবে অথবা || এর ফলাফলের ওপর নয় ! করলে যে ফলাফল পাওয়া যায় তা আগে উপাদানগুলোর ওপরে নয় ! করে সেই ফলাফলের ওপর এবং && চালিয়ে পাওয়া ফলাফলের সমতুল।

- !(x && y) ≡ !x || !y ডি মরগান
- !(x || y) ≡ !x && !y ডি মরগান

৭.১২ সত্যক সারণী (Truth Table)

সমতুলের বিধিগুলো (equivalence law) যে সঠিক, অথবা যে কোন দুটো বুলক রাশি সমতুল কিনা, এইটা তুমি কীভাবে প্রমাণ করবে। প্রমাণ করাটা আসলে খুবই সহজ। উপাদানগুলোর মানের যত রকম সমাবেশ (combination) সম্ভব, প্রতিটির জন্য তোমাকে সমতুল বিধির বাম ও ডান পাশ সমান কিনা পরীক্ষা করে দেখতে হবে। আমরা সাধারণত সত্যক সারণী (truth table) ব্যবহার করে সেটা করে থাকি। চলো উদাহরণ হিসাবে আমরা ডি মরগানের বিধি দুটোর প্রথমটি প্রমাণ করি। একই পদ্ধতি অনুসরণ করে তুমি ডি মরগানের অন্য বিধিটি প্রমাণ করতে পারবে। আর চাইলে উপরের অন্যান্য যে কোন সমতুলের বিধিগুলোও নিজে নিজে প্রমাণ করবে।

ডি মরগানের প্রথম সূত্রটিতে চলক (variable) আছে দুইটি x ও y, আর চলক দুটির মান সম্ভব কেবল true ও false। এখন দুটি চলকের জন্যে দুটি মান নিয়ে আমরা চারটি সমাবেশ (combination) পেতে পারি। এর প্রতিটির জন্য আমরা বিধিটির বাম পাশ ও ডান পাশ মূল্যায়ন (evaluate) করে দেখবো। এখানে বলে রাখি কোন সমতুল বিধিতে যদি ৩টি চলক থাকে তাহলে সমাবেশ হবে ৮টি, ৪টি থাকলে হবে ১৬টি, অর্থাৎ n টি চলক থাকলে সমাবেশ হবে 2^nটি। আর এর প্রতিটি সমাবেশের জন্য সত্যক সারণীতে (truth table) একটি করে আড়ি (row) থাকবে। সত্যক সারণীতে খাড়িগুলো (column) হবে বিভিন্ন উপরাশির (subexpression) মান যে গুলোর মান আমাদের মূল্যায়ন করতে হবে যদি আমরা মুল রাশির (expression) মান পেতে চাই।

যেমন !(x && y) মূল্যায়ন করতে গেলে আমাদের x && y আগে মূল্যায়ন করতে হবে, তেমনি ভাবে !x || !y মূল্যায়ন করতে গেলে !x ও !y মূল্যায়ন করতে হবে।

সত্যক সারণী (Truth Table)

x	y	x && y	!(x && y)	!x	!y	!x || !y
true	true	true	false	false	false	false
true	false	false	true	false	true	true
false	true	false	true	true	false	true
false	false	false	true	true	true	true

উপরের সত্যক সারণীতে (truth table) প্রতিটি আড়ি (row) খেয়াল করো:

১. প্রথম আড়িতে (row) x ও y উভয়ের মানই true। সুতরাং x && y ও true, ফলে !(x && y) হবে false। তারপর !x আর !y উভয়ই হলো false, ফলে !x || !y হলো false। কাজেই !(x && y) আর !x || !y উভয়ের মান সমান।

২. দ্বিতীয় আড়িতে (row) x,y যথাক্রমে true, false, ফলে x && y হলো false আর !(x && y) হলো true। তারপর !x ও !y হবে যথাক্রমে false ও true, ফলে !x || !y হলো true। সুতরাং !(x && y) আর !x || !y এর মান সমান।

৩. তৃতীয় আড়িতে (row) x,y যথাক্রমে false, true, ফলে x && y হলো false আর !(x && y) হলো true। তারপর !x ও !y হবে যথাক্রমে true ও false, ফলে !x || !y হলো true। সুতরাং !(x && y) আর !x || !y এর মান সমান।

৪. চতুর্থ আড়িতে (row) x ও y উভয়ের মানই false। সুতরাং x && y ও false, ফলে !(x && y) হবে true। তারপর !x আর !y উভয়ই হলো true, ফলে !x || !y হলো true। কাজেই !(x && y) আর !x || !y উভয়ের মান সমান।

সুতরাং উপাদানগুলোর (operand) মান যাই হোক না কেন সর্বাবস্থায় !(x && y) আর !x || !y এর মান সমান, অর্থাৎ তারা একে অপরের সমতুল প্রমাণ হয়ে গেলো।

৭.১৩ বুলক সরলীকরণ (Boolean Simplification)

শর্তালি পরিগণনায় (conditional programming) বুলক বীজগণিত (boolean algebra) ঠিক কী কাজে লাগে? বুলক বীজগণিত ব্যবহার করে কিছু সরলীকরণের উদাহরণ দেখাও। আর এই সরলীকরণের কারণে ক্রমলেখতে (program) কী প্রভাব পড়ে সেটাও দেখাও।

ধরো তোমাকে একটি ক্রমলেখ (program) লিখতে হবে যেটি তুমি কোন শ্রেণীতে পড়ো আর তোমার বয়স কত এই দুটি যোগান (input) নিয়ে জানাবে তুমি মোরগ লড়াই খেলতে পারবে কি না। তুমি যদি পঞ্চম শ্রেণীতে পড়ো তাহলে তুমি মোরগ লড়াই খেলতে পারবে। আর তুমি যদি পঞ্চম শ্রেণীতে নাও হও কিন্তু তোমার বয়স যদি ১০ বছর হয় তাহলেও তুমি মোরগ লড়াই খেলতে পারবে। এই ক্রমলেখটি আমরা যদি-নাহলে দিয়ে খুব সহজে লিখে ফেলতে পারি।

নীচের ক্রমলেখ খেয়াল করো। আমরা দুটো চলক ব্যবহার করছি shreni ও boyosh, যে দুটো প্রথমে ঘোষণা (declare) করে তারপর যোগান যাচনা (input prompt) দেখিয়ে যোগান (input) নিতে হবে। ধরে নিই তুমি ওগুলো নিজে নিজে করতে পারবে। আমরা কেবল প্রাসঙ্গিক

অংশটুকু দেখি। প্রথমে if (shreni == 5) দিয়ে পরীক্ষা করা হলো পঞ্চম শ্রেণী কিনা, হলে ফলন (output) হবে "khelte parbe"। আর পঞ্চম শ্রেণী যদি না হয় কিন্তু বয়স যদি ১০ বছর হয় সেটা পরীক্ষা করার জন্য আমাদের লাগবে if (shreni != 5 && boyosh == 10) যেটি আমরা আগের if এর else এর সাথে লাগিয়ে দিবো। আর সবশেষে কোন if এর শর্তই সত্য না হলে আমরা ফলন (output) দেখাবো "khelte parbena"। একটা গুরুত্বপূর্ণ বিষয় খেয়াল করো, বাংলা ভাষায় যেটা "কিন্তু" সেটা সিপিপিতে গিয়ে হয়ে যাচ্ছে "এবং" &&।

```
if (shreni == 5)
  cout << "khelte parbe" << endl;
else if (shreni != 5 && boyosh == 10)
  cout << "khelte parbe" << endl;
else  // উপরের কোনটিই না হলে
  cout << "khelte parbena" << endl;
```

উপরের ক্রমলেখতে দুটো if এর শর্ত সত্য হলেই আমাদের একই ফলন দেখাতে হয়। আমরা তাই চেষ্টা করতে চাই একটা if দিয়ে বিষয়টা সামলাতে। সেটা করা খুবই সহজ যদি তুমি সমস্যাটা উল্টো দিক থেকে ভাবো। তুমি মোরগ লড়াই খেলতে পারবে যদি তুমি ৫ম শ্রেণী পড়ো অথবা তুমি ৫ম শ্রেণীতে না কিন্তু তোমার বয়স ১০ বছর হলে। তো এই থেকে তুমি খুব সহজে খেলতে পারার শর্ত লিখে ফেলতে পারো shreni == 5 || shreni != 5 && boyosh == 10, তাই না!

```
if (shreni == 5 || shreni != 5 && boyosh == 10)
  cout << "khelte parbe" << endl;
else  // উপরের শর্ত সত্য না হলে
  cout << "khelte parbena" << endl;
```

এখন কথা হচ্ছে এই যে খানিকটা জটিল একটা শর্ত আমরা লিখে ফেললাম, এটাকে কি কোন ভাবে সরলীকরণ করা যায়? সরলীকরণ করার জন্য চলো ধরে নিই p ≡ shreni == 5 আর q ≡ boyosh == 5। তাহলে shreni != 5 কে লেখা যায় !p। ফলে আমাদের শর্তটি দাঁড়ালো p || !p && q, আমরা এটিকে বুলক বীজগণিত (boolean algebra) দিয়ে সরল করবো।

p || !p && q	প্রদত্ত শর্ত যা সরল করতে হবে
≡ (p || !p) && (p || q)	বন্টন বিধি (distribution)
≡ true && (p || q)	নঞ মধ্যম (excluded middle)
≡ p || q	সত্যের সরল (true simplification)

সুতরাং উপরের সরলের ফলে প্রাপ্ত রাশি (expression) অনুযায়ী আমাদের ক্রমলেখ দাঁড়াবে নিম্নরূপ, যেখানে আমাদের একটি অতিরিক্ত শর্ত আর মূল্যায়ন করতে হচ্ছে না। আমরা p এর বদলে shreni == 5 আর q এর বদলে boyosh == 10 লিখবো।

```
if (shreni == 5 || boyosh == 10)
  cout << "khelte parbe" << endl;
else  // উপরের শর্ত সত্য না হলে
  cout << "khelte parbena" << endl;
```

একই রকম আরেকটি উদাহরণ দেখো। ধরো কোন একটা ক্রমলেখতে (program) শর্ত দাঁড়াচ্ছে `!(p && (!p || q)) || q`। এখন কথা হচ্ছে এটিকে সরল করলে কী দাঁড়াবে।

`!(p && (!p || q)) || q`	প্রদত্ত শর্ত যা সরল করতে হবে
≡ `!((p && !p) || (p && q)) || q`	বন্টন বিধি (distribution)
≡ `!(false || (p && q)) || q`	অসঙ্গতি (contradiction)
≡ `!(p && q) || q`	মিথ্যার সরল (false simplification)
≡ `(!p || !q) || q`	ডি মরগান (De Morgan)
≡ `!p || (!q || q)`	সহযোজন (associative)
≡ `!p || true`	নঞ মধ্যম (excluded middle)
≡ `true`	সত্যের সরল (true simplification)

উপরের সরলীকরণের ফলে আমরা `if (!(p && (!p || q)) || q)` না লিখে কেবল `if (true)` লিখতে পারবো। কিন্তু একটা বিষয় দেখেছো, সরলীকরণের ফলাফল একদম একটা ধ্রুবক মান `true` হয়ে গেছে। এর অর্থ প্রদত্ত শর্তের মান কখনো চলক `p` বা `q` এর ওপর নির্ভর করেনা। সুতরাং আমাদের আদৌ কোন `if` লাগানোর দরকার নাই। কারণ শর্ত সত্য হলে যেটি করতে হতো শর্ত সবসময় সত্য হওয়ায় তুমি সেটি এখন শর্ত পরীক্ষণ ছাড়াই করবে।

```
// if (true)  // শর্ত লেখার দরকার নাই, টীকায় আটকে দিয়েছি
   cout << "kee moja" << endl; // কেবল এটি লিখলেই হবে
```

তুমি এবার জিজ্ঞেস করতে পারো সরলীকরণের ফলে যদি `false` আসে তাহলে কী হবে? সত্যিই তো কী হবে? সেক্ষেত্রে আমাদের লিখতে হবে `if (false)` তাই না! কিন্তু সেটা মানে তো শর্ত সব সময় মিথ্যা, শর্তটির সত্য হওয়ার কোন সম্ভাবনা নেই। আর সেক্ষেত্রে শর্ত সত্য হলে যা করার কথা ছিলো সেটা কখনোই করতে হবে না। ফলে তুমি এই `if (false)` আর তারপর শর্ত সত্য হলে যা করতে তার সব ক্রমলেখ (program) থেকে মুছে দিতে দিতে পারো।

```
// if (false) // শর্ত লেখার দরকার নাই, টীকায় আটকে দিয়েছি
//  cout << "kee moja" << endl; // শর্ত সব সময় মিথ্যা
```

৭.১৪ মই, অন্তান্তি, সংযোজক (Ladder,Nesting,Connectives)

যদি-নাহলের মই (if-else ladder) ও অন্তান্তি যদি-নাহলে (nested if-else) ব্যবহার না করে কী ভাবে সংযোজক (connectives) এবং `&&`, অথবা `||`, নয় `!` ব্যবহার করে একই উদ্দেশ্য বাস্তবায়ন করা যায় তা আলোচনা করো। অথবা উল্টোটা অর্থাৎ সংযোজক ব্যবহার না করে কী ভাবে যদি-নাহলের মই বা অন্তান্তি যদি ব্যবহার করে কাজ চালানো যায় তা দেখাও।

নীচের উদাহরণগুলো খেয়াল করো। এগুলোতে দুটো করে স্তম্ভ আছে। বামপাশের স্তম্ভে যদি-নাহলে মই (if-else ladder) অথবা অন্তান্তি যদি-নাহলে (nested if-else) দিয়ে ক্রমলেখ লেখা হয়েছে, আর ডান পাশের স্তম্ভে তার সমতুল (equivalent) ক্রমলেখ লেখা হয়েছে সংযোজক (connectives) এবং `&&` অথবা `||` না `!` দিয়ে। আমরা আসলে সুবিধামতো কখনো বামপাশের মতো করে লিখি আবার কখনো ডানপাশের মতো করেও লিখি।

```
if (shorto1)
  cout << "kisu ekta";
else if (shorto2)
  cout << "kisu ekta";
else
  cout << "onno kisu";
cout << "koro" << endl;
```

```
if (shorto1 || shorto2)
  cout << "kisu ekta";
else
  cout << "onno kisu";
cout << "koro" << endl;
```

উপরের যদি-নাহলে মইয়ের (if-else ladder) উদাহরণে খেয়াল করো shorto1 সত্য হলেও "kisu ekta" ফলনে (output) যাবে আবার shorto2 সত্য হলেও "kisu ekta" ফলনে (output) যাবে। আর এ দুটোই মিথ্যা হলে ফলনে যাবে "onno kisu"। বাম ও ডান উভয় পাশের ক্রমলেখতেই (program) এই একই ব্যাপার ঘটবে। একটা বিষয় উল্লেখ করা দরকার: shorto1 সত্য হলে বামপাশে দেখো shorto2 পরীক্ষণই দরকার পরে না। ডানপাশেও আসলে একই ঘটনা ঘটবে। অথবা || এর ফলাফল যেহেতু যে কোন একটি উপাদান সত্য হলেই সত্য হয়, সেহেতু shorto1 সত্য হলেই shorto2 এর মুল্যায়ন ছাড়াই || এর ফলাফল সত্য হয়ে যাবে। এই যে ব্যাপারটি এটাকে বলা আংশিক মুল্যায়ন (partial evaluation), এতে অদরকারী কাজ কিছুটা কমে, ক্রমলেখ (program) কিঞ্চিত দ্রুতগতির হয়।

```
if (shorto1)
  if (shorto2)
    cout << "kisu ekta";
  else
    cout << "onno kisu";
else
  cout << "onno kisu";
cout << "koro" << endl;
```

```
if (shorto1 && shorto2)
  cout << "kisu ekta";
else
  cout << "onno kisu";
cout << "koro" << endl;
```

উপরের অন্তান্তি যদি-নাহলের (nested if-else) উদাহরণে খেয়াল করো shorto1 সত্য হলে তারপর shorto2ও সত্য হলে "kisu ekta" ফলনে (output) যাবে। আর শর্তদুটোর যে-কোন একটা মিথ্যা হলেও ফলনে (output) যাবে "onno kisu"। বাম ও ডান উভয় পাশে ক্রম-লেখতেই (program) এই একই ব্যাপার ঘটবে। এখানেও সেই একটা বিষয় উল্লেখ করা দরকার: shorto1 মিথ্যা হলে বামপাশে দেখো shorto2 পরীক্ষণই দরকার পরে না। ডানপাশেও আসলে একই ঘটনা ঘটবে। অথবা && এর ফলাফল যেহেতু যে কোন একটি উপাদান মিথ্যা হলেই মিথ্যা হয়, সেহেতু shorto1 মিথ্যা হলেই shorto2 এর মুল্যায়ন ছাড়াই && এর ফলাফল মিথ্যা হয়ে যাবে। এই যে ব্যাপারটি এটাকে বলা আংশিক মুল্যায়ন (partial evaluation), এতে অদরকারী কাজ কিছুটা কমে, গতি কিছুটা বাড়ে।

```
if (shorto)
  cout << "kisu ekta";
else
  cout << "onno kisu";
cout << "koro" << endl;
```

```
if (!shorto)
  cout << "onno kisu";
else
  cout << "kisu ekta";
cout << "koro" << endl;
```

উপরের উদাহরণে বামপাশে shorto ব্যবহার করা হয়েছে আর ডানপাশে !shorto। ফলে শর্ত সত্য হলে যা করতে হবে আর মিথ্যা হলে যা করতে হবে এই দুটো স্থান বদলাবদলি করেছে।

```
if (shorto1)                    if (shorto1 || !shorto2)
  cout << "kisu ekta";            cout << "kisu ekta";
else if (shorto2)               else
  cout << "onno kisu";            cout << "onno kisu";
else                            cout << "koro" << endl;
  cout << "kisu ekta";
cout << "koro" << endl;
```

উপরের উদাহরণে খেয়াল করে দেখো "kisu ekta" ফলনে (output) যাবে যদি shorto1 সত্য হয় অথবা যদি shorto2 মিথ্যা হয়, অন্য কথায় !shorto2 সত্য হয়। আর shorto1 মিথ্যা হলে তারপর shorto2ও মিথ্যা হলে ফলনে (output) যাবে "onno kisu"। ঠিক এই ব্যাপারটিই উভয়পাশের ক্রমলেখতে (program) প্রতিফলিত হয়েছে।

```
if (shorto1)                    if (!shorto1 || shorto2)
  if (shorto2)                    cout << "kisu ekta";
    cout << "kisu ekta";          else
  else                            cout << "onno kisu";
    cout << "onno kisu";          cout << "koro" << endl;
else
  cout << "kisu ekta";
cout << "koro" << endl;
```

উপরের উদাহরণটি একটু জটিল। বামপাশে খেয়াল করো "kisu ekta" ফলনে (output) যাবে যদি shorto1 মিথ্যা হয় অথবা তা না হলে যদি shorto2 সত্য হয়। কথায় বললে ঠিক তাই-ই ডানপাশেও লিখা হয়েছে। আর একটু বেশী গভীরে বুঝতে চাইলে ধরো বামপাশে "kisu ekta" ফলনে যাবে যদি shorto1 && shorto2 || !shorto1 সত্য হয়। বুলক বীজগণিত দিয়ে সরলীকরণ করলে এটি আসবে !shorto1 || shorto2, তুমি নিজে চেষ্টা করে দেখো।

```
bool shorto = true;             if (shorto1 && shorto2)
if (!shorto1)                     cout << "kisu ekta";
  shorto = false;               else
if (!shorto2)                     cout << "onno kisu";
  shorto = false;               cout << "koro" << endl;
if (shorto)
  cout << "kisu ekta";
else
  cout << "onno kisu";
cout << "koro" << endl;
```

উপরের এই উদাহরণটি খেয়াল করো। প্রথম দুটি if একেবারে আলাদা আলাদা, কেউ কারো অন্তান্তিও (nested) নয়, আবার মইও (ladder) নয়। ডানপাশে যেমন খুব সুন্দর করে সংক্ষেপে আমরা short1 && shorto2 লিখেছি। অনেকসময়ই এটা করা সম্ভব হয় না। কারণ শর্তদুটো আলাদা করে প্রথমে মূল্যায়ন করাটা হয়তো বেশ এক একটা কাজ। তো এইরকম ক্ষেত্রে আমরা বামপাশে যেটি করেছি আলাদা একটা চলক নিয়েছি shorto যেখানে মুলত আমরা && এর ফলাফল চাই। আমরা জানি && ফলাফল যে কোন একটি উপাদান (operand) মিথ্যা হলেই মিথ্যা

হয়। তাই আমরা শুরুতে shorto এর মান নিয়েছি true, এরপর shorto1 মিথ্যা হলে অর্থাৎ !shorto1 সত্য হলে আমরা shorto কে মিথ্যা করে দিয়েছি। একই ভাবে shorto2 মিথ্যা হলে অর্থাৎ !shorto2 সত্য হলেও আমরা shorto কে মিথ্যা করে দিয়েছি। তাহলে দুটো শর্তের যে কোনটি মিথ্যা হলেই shorto মিথ্যা হয়ে যাবে। ঠিক && এর ফলাফলের মতো। শেষের if else এ এবার shorto ব্যবহার করে ফলন দেবার পালা। তবে একটা বিষয় খেয়াল করো ডানপাশে যে-মন shorto1 মিথ্যা হলে আংশিক মূল্যায়নের কারণে (partial evaluation) shorto2 আর পরীক্ষণই করা হবে না, বামপাশে কিন্তু তা হচ্ছে না। তুমি যদি এই উন্নয়ন টুকু করতে চাও তাহলে তোমাকে if (!shorto2) বদলে লিখতে হবে else if (!shorto2)।

```
bool shorto = false;
if (shorto1)
  shorto = true;
if (shorto2)
  shorto = true;
if (shorto)
  cout << "kisu ekta";
else
  cout << "onno kisu";
cout << "koro" << endl;
```

```
if (shorto1 || shorto2)
  cout << "kisu ekta";
else
  cout << "onno kisu";
cout << "koro" << endl;
```

এই উদাহরণটিও ঠিক আগের উদাহরণটি মতো, তবে এখানে || এর জন্য করা হয়েছে। অথ-বার || ক্ষেত্রে যেকোন একটি উপাদান (operand) সত্য হলেই ফলাফল সত্য হয়, আমরা তাই shorto এর আদি মান ধরেছি false। আর তারপর শর্তদুটোর যে কোনটি সত্য হলেই shorto কে সত্য করা হয়েছে। তুমি যদি আংশিক মূল্যায়ন (partial evaluation) এখানেও কাজে লা-গাতে চাও তাহলে বামপাশে if (shorto2) বদলে else if (shorto2) লিখবে।

৭.১৫ যদি-নাহলে অনুকুলায়ন (If-Else Optimisation)

ধরো তোমার ইশকুলে গণিত পরীক্ষায় ৫০ বা বেশী পেলে পাশ, না হলে ফেল। আর ৮০ বা বেশী পেলে তারকা নিয়ে পাশ। তোমার শ্রেণীতে ১০০ জন শিক্ষার্থী আছে, যাদের মধ্যে মোটামুটি ১০ জন ফেল করবে, ২০ জন তারকা সহ পাশ করবে আর বাঁকী ৭০ জন স্রেফ পাশ করবে। তুমি এমন একটি ক্রমলেখ (program) রচনা করো যেটি একজন শিক্ষার্থীর ছাত্রের নম্বর যোগান (input) নিয়ে ফেল, পাশ, বা তারকা সহ পাশ ফলন (output) দিবে। তোমার ক্রমলেখটি ১০০ জন শিক্ষার্থীর জন্য ১০০ বার আলাদা আলাদা করে চালানো (run) হবে। তবে এই ১০০ বার চালানোতে মোট সময় যাতে কম লাগে ক্রমলেখটা সেটা মাথায় রেখে রচনা করতে হবে।

```
if (nombor >= 50)              // যদি পাশের নম্বর
  cout << "pash" << endl;      // পাশ ফলন
else                           // না হলে
  cout << "fell" << endl;      // ফেল ফলন

if (nombor >= 80)              // যদি তারকা নম্বর
  cout << "taroka" << endl;    // তারকা ফলন
```

ধরো উপরের মতো করে তুমি ক্রমলেখ তৈরী করেছো। যে শিক্ষার্থী ফেল করলো বা পাশ করলো বা তারকা সহ পাশ করলো, তার জন্য তো যা ফলন তা দেখাতেই হবে, সেখানে সময় কম লাগা বেশী লাগার ব্যাপার নাই। সময় কম বা বেশী লাগার প্রশ্ন হলো তুমি কতবার শর্ত পরীক্ষা করে কাজটা করতে পারছো সেটাতে। যেমন ধরো একজন ফেল করা শিক্ষার্থীর জন্য উপরের ক্রমলেখতে (nombor >=50) শর্ত পরীক্ষা হবে আবার ক্রমলেখ যে ভাবে লেখা হয়েছে তাতে nombor >= 80 শর্তটিও পরীক্ষা হবে। শর্ত পরীক্ষার ফলাফল সত্য হোক আর মিথ্যা হোক পরীক্ষা তো করতেই হবে। ফলে মোট দুটি শর্ত পরীক্ষা হলো। যে শিক্ষার্থীটি কেবল পাশ করবে খেয়াল করে দেখো তার জন্যেও দুটিই শর্তই পরীক্ষা করতে হবে। একই হবে তারকাসহ পাশের ছাত্রের জন্যেও দুটি শর্তই পরীক্ষা করতে হবে। সুতরাং উপরের ক্রমলেখ দিয়ে এই সমস্যার সমাধান করলে ১০০ জন শিক্ষার্থীর জন্য মোট শর্ত পরীক্ষা হলো ১০০ * ২ = ২০০ বার।

```
if (nombor >= 50)                  //  যদি পাশের নম্বর
{
  cout << "pash" << endl;          //  পাশ ফলন
  if (nombor >= 80)                  //  যদি তারকা নম্বর
    cout << "taroka" << endl;      //  তারকা ফলন
}
else                               //  না হলে
  cout << "fell" << endl;          //  ফেল ফলন
```

এবার একটু ভেবে দেখো পাশ বা ফেল নির্ণয় করার জন্য তো আমাদের একটা শর্ত লাগবেই, কিন্তু যখন আমরা জেনে গেলাম একজন শিক্ষার্থী ফেল করেছে, তখন তার জন্যেও কেন আমরা nombor >= 80 শর্ত পরীক্ষা করবো? সেটা তো অদরকারী কাজ হবে। সুতরাং তারকা দেখানো অংশটুকু যদি আমরা পাশের জন্য যে অংশ সেখানে একটা মহল্লা (block) তৈরী করে সেই মহল্লার ভিতরে নিয়ে যাই, তাহলে nombor >= 80 শর্তটি কেবল পাশ করা শিক্ষার্থীদের জন্য পরীক্ষা হবে। উপরের ক্রমলেখ দেখো। তো এই ক্ষেত্রে পাশ বা ফেল শিক্ষার্থীর জন্য কেবল ১ টা শর্ত পরীক্ষা হলো আর তারকা পাওয়া ছাত্রের জন্য ২টা সুতরাং মোট শর্ত পরীক্ষণ হলো ২০*২ + (৭০+১০)*১ = ১২০ বার মাত্র। নিশ্চিতভাবেই এই ক্রমলেখ আগেরটির চেয়ে তাড়াতাড়ি ১০০ জন শিক্ষার্থীর ফলাফল দেখানোর কাজ শেষ করবে! কেমন মজার বিষয় না!

```
if (nombor >= 80)                  //  যদি তারকা নম্বর
{
  cout << "pash" << endl;          //  পাশ ফলন
  cout << "taroka" << endl;        //  তারকা ফলন
}
else if (nombor >= 50)             //  যদি পাশের নম্বর
  cout << "pash" << endl;          //  পাশ ফলন
else                               //  না হলে
  cout << "fell" << endl;          //  ফেল ফলন
```

তুমি হয়তো ভাবছো দেখি আরেক ভাবে করা যায় কিনা যাতে আরো কম সময় লাগে। যেমন ধরো তুমি প্রথমে ৮০ বা বেশী কিনা পরীক্ষা করবে, তারপর ৫০ এর বেশী কিনা পরীক্ষা করবে, অর্থাৎ উপরের ক্রমলেখয়ের (program) মতো করে। এখানে খেয়াল করো তারকা পাওয়া শিক্ষার্থীদের জন্য শর্ত পরীক্ষা করা লাগবে ১বার সেটি nombor >= 80 আর স্রেফ পাশ বা ফেল করা শিক্ষার্থীদের জন্য ২টি শর্তই পরীক্ষা করা লাগবে। ফলে মোট শর্ত পরীক্ষণ হবে ২০*১ +

(৭০+১০)*২ = ১৮০ বার। সুতরাং উপরের এই তৃতীয় ক্রমলেখ আমাদের লেখা প্রথম ক্রমলেখ-য়ের চেয়ে একটু দ্রুতগতির হলেও দ্বিতীয়টির চেয়ে যথেষ্ট ধীরগতির হবে। তুমি আরো নানান ভাবে চেষ্টা করে দেখতে পারো, তবে আমাদের দ্বিতীয় ক্রমলেখটিই সবচেয়ে দ্রুতগতির হবে, কারণ এতে সবচেয়ে কম সংখ্যক বার শর্ত পরীক্ষা করতে হয়েছে।

আচ্ছা তুমি কী ধরতে পেরেছো কেন দ্বিতীয় ক্রমলেখটিতে সবচেয়ে কম সংখ্যক বার শর্ত পরী-ক্ষা করতে হবে? উত্তরটা কিন্তু খুবই সহজ। আমাদের দেখতে হবে সবচেয়ে বেশী সংখ্যক শিক্ষার্থী কোন ভাগে পড়ে। এক্ষেত্রে স্রেফ পাশ করে সর্বোচ্চ ৭০ জন। আমরা চাইবো এই ৭০ জনের জন্য ফলন (output) যাতে কম সংখ্যক, এক্ষেত্রে মাত্র একটা শর্ত পরীক্ষা করেই দিতে পারি। উল্টা দি-কে যে ভাগে শিক্ষার্থীর সংখ্যা যত কম তার জন্য তত বেশী শর্ত পরীক্ষা করা যেতে পারে। আমাদের তৃতীয় ক্রমলেখতে আমরা আসলে এই নিয়ম ভঙ্গ করেছি। কারণ এটাতে তারকা পাওয়া ২০ জনের ফলন আমরা দেখাই মাত্র ১বার শর্ত পরীক্ষা করে, আর পাশ করা ৭০জনের ফলন দেখাই ২বার শর্ত পরীক্ষা করে। আর সে কারণে এটি দ্বিতীয় ক্রমলেখ থেকে ধীরগতির হবে। তো এখন থেকে যদি-তাহলে নিয়ে কাজ করার সময় শর্তদিয়ে সৃষ্টি হওয়া ডাল-পালাগুলোর কোনটাতে কতগুলো ব্যাপার (case) আসতে পারে সেটা মাথায় রেখে দক্ষ ক্রমলেখ তৈরী করবে, কেমন!

৭.১৬ তিনিক অণুক্রিয়া (Ternary Operator)

সিপিপিতে শর্তালী পরিগণনায় (conditional programming) তিনিক অণুক্রিয়াটি (ternary operator) কী? উদাহরণসহ তিনিক অণুক্রিয়াটির ব্যবহার দেখাও।

সিপিপি ভাষায় ? : এই প্রতীক দুটি বিশেষ ভাবে একসাথে ব্যবহার করে তিনিক অণুক্রি-য়াটি (ternary operator) পাওয়া যায়। তিনিক অণুক্রিয়াটি যদি-তাহলে-নাহলের (if-then-else) কাজ করে, তবে দুটোর মধ্যে তফাৎ হলো তিনিক অণুক্রিয়া একটি রাশির (expression) অংশ হিসাবে থাকে, ফলে এর একটা ফলাফল তৈরী হবে। আর if-else একটা শর্তযুক্ত বিবৃতি (conditional statement) তৈরী করে যার কোন ফলাফল নেই।

```
int prothom, ditiyo;     // চলকদুটির মান যোগান নিতে পারো

int boro = prothom > ditiyo ? prothom : ditiyo;
```

তিনিক অণুক্রিয়া ব্যবহার করে আমরা উপরে দুটো সংখ্যার বড়টি বের করার ক্রমলেখ দেখি-য়েছি। এখানে প্রথমে প্রশ্ন ? চিহ্নের আগে যে শর্ত পরীক্ষা আছে সেটি মূল্যায়ন হবে। শর্ত যদি সত্য হয় তাহলে প্রশ্ন ? আর দোঁটা : চিহ্নের মাঝে যে মানটি আছে সেটি হবে অণুক্রিয়াটির ফলাফল আর শর্ত যদি মিথ্যা হয় তাহলে অণুক্রিয়াটির ফলাফল হবে দোঁটা : চিহ্নের পরে থাকা অংশটুকু। তাহ-লে উপরের ক্রমলেখতে prothom > ditiyo শর্তটি সত্য হলে ফলাফল হবে prothom অর্থাৎ বড়টি আর শর্তটি মিথ্যা হলে ফলাফল হবে ditiyo কারন এটিই তখন বড় অন্যটির চেয়ে। সুতরাং আমরা ফলাফল হিসাবে prothom ও ditiyo চলকদুটির মধ্যে সবসময় বড়টিই পাচ্ছি। তুমি নিশ্চয় এখন দুটো সংখ্যার মধ্যে ছোটটি বের করার ক্রমলেখ এভাবে লিখতে পারবে!

```
int prothom, ditiyo;     // চলকদুটির মান যোগান নিতে পারো
int boro;                // বড় মানটি রাখার জন্য চলক ঘোষণা

prothom > ditiyo ? boro = prothom : boro = ditiyo;
```

তুমি কিন্তু চাইলে দুটো সংখ্যার বড়টি বের করার জন্য উপরের মতো করেও লিখতে পারতে। এইক্ষেত্রে চলক (variable) boro তে মান আরোপণ (assign) আমরা তিনিক অণুক্রিয়ার ভিতরেই করেছি খেয়াল করো। আরোপণ (assign) অণুক্রিয়ার ফলাফল (result) তো আরোপিত মানটিই হয়, সুতরাং এক্ষেত্রেও তিনিক অণুক্রিয়ার ফলাফল হিসাবে আমরা বড়টিই পাবো, যদিও boro চলকে মান আরোপণ আগেই হয়ে গিয়েছে। তুমি জিজ্ঞেস করতে পারো এই ক্ষেত্রে তিনিক অণুক্রিয়াটির যেটা ফলাফল আসবে সেটা আসলে কী কাজে লাগবে। এইখানে আসলে আমরা ফলাফলটি কাজে লাগাচ্ছি না। কিন্তু তুমি চাইলে int fol = prothom > ditiyo ? boro = prothom : boro = ditiyo; লিখতেই পারো। সেক্ষেত্রে বড় মানটি boro চলকের মধ্যে যেমন থাকবে তেমনি fol চলকের মধ্যেও থাকবে। তিনিক অণুক্রিয়ার ব্যবহার এভাবে বেশ সংক্ষিপ্ত।

```
int prothom, ditiyo;      // চলকদুটির মান যোগান নিতে পারো
int boro;                 // বড় মানটি রাখার জন্য চলক ঘোষণা

if (prothom > ditiyo)     // প্রথমটি বড় হলে
  boro = prothom;
else
  boro = ditiyo;          // আর তা না হলে
```

তিনিক অণুক্রিয়ার কাজ তো উপরের মতো করে যদি-নাহলে দিয়েও করা যেতে পারে। তাহলে কখন তুমি তিনিক অণুক্রিয়া ব্যবহার করবে কখন যদি-নাহলে ব্যবহার করবে? অত্যন্ত সংক্ষিপ্ত ধরনের বলে তিনিক অণুক্রিয়া (ternary) আসলে টুকটাক ছোটখাট কিছুর জন্য বেশী ব্যবহার করা হয়। আর যদি-নাহলে হলো একদম সব জায়গায় ব্যবহার করার জন্য, বিশেষ করে শর্ত সত্য বা মিথ্যা হলে যদি একটা মহল্লা (block) নির্বাহ (execute) করতে হয়।

```
int prothom, ditiyo, tritiyo; // মান যোগান নিতে হবে

int boro = prothom > ditiyo ? prothom : ditiyo;
boro = boro > tritiyo ? boro : tritiyo;
```

তুমি কি তিনিক অণুক্রিয়া (ternary operator) ব্যবহার করে তিনটি সংখ্যার মধ্যে সবচেয়ে বড়টি বের করতে পারবে। নিশ্চয় পারবে, এ আর এমন কঠিন কী? উপরের ক্রমলেখয়ের মতো করে প্রথমে দুটোর মধ্যে বড়টি বের করবে। তারপর boro এর সাথে tritiyo টি তুলনা করে যদি boro টিই বড় হয় তাহলে ফলাফল boro আর যদি tritiyo টি বড় হয় তাহলে ফলাফল tritiyo টি। কিন্তু আমরা আসলে এই রকম আলাদা দুটো তিনিক অণুক্রিয়া চাচ্ছি না। আমরা বরং একটা তিনিক অণুক্রিয়াকে আরেকটি তিনিক অণুক্রিয়ার মধ্যে ঢুকিয়ে দিবো, আর যাকে বলব অন্তান্তি (nested) তিনিক অণুক্রিয়া। নীচের ক্রমলেখ খেয়াল করো, আমরা একটু ছাড়ন (indentation) দিয়ে লিখেছি। প্রথমে prothom ও ditiyo তুলনা করা হয়েছে। শর্ত সত্য হওয়া মানে prothom বড় যেটিকে tritiyo এর সাথে তুলনা করা হয়েছে। আর শর্ত মিথ্যা হওয়া মানে ditiyo বড়, কাজেই এটিকে tritiyo এর সাথে তুলনা করা হয়েছে। তিনিক অণুক্রিয়া ব্যবহার করেই আরো নানান ভাবে এটি করা সম্ভব, তুমি নিজে নিজে চেষ্টা করে দেখো।

```
int prothom, ditiyo, tritiyo; // মান যোগান নিতে হবে

int boro = prothom > ditiyo ?
      (prothom > tritiyo ? prothom : tritiyo) :
      (ditiyo > tritiyo ? ditiyo : tritiyo) ;
```

৭.১৭ পল্টি ব্যাপার (Switch Cases)

এমন একটি ক্রমলেখ (program) রচনা করো যেটি একটি পূর্ণক (integer) আর একটি ভগ্নক (fractioner) যোগান (input) নিবে। পূর্ণকটি ১ হলে ক্রমলেখটি পরের সংখ্যাটিকে কোণের পরিমান রেডিয়ানে ধরে নিয়ে তার লম্বানুপাত (sine) ফলন দিবে। আর পূর্ণকটি ২ হলে লগ্নানুপাত (cosine), ৩ হলে স্পর্শানুপাত (tangent) ফলন দিবে। তবে এই তিনটির কোনটিই না হলে বলবে "অসমর্থিত পছন্দ"। এই ক্রমলেখটিতে তুমি পল্টি ব্যাপার (switch case) ব্যবহার করবে আর যদি-নাহলের (if-else) ব্যবহারের সাথে কী তফাৎ হয় সেটাও আলোচনা করবে।

ফিরিস্তি ৭.৫: প্রাপণ্য সহ ত্রিকোণমিতি (Trigonometry with Menu)

```
int onupat;    // কোন অনুপাত sine, cosine, tangent
float kone;    // কোণের পরিমান রেডিয়ানে

// প্রথমে প্রাপণ্য (menu) দেখানো হবে
cout << "onupat 1: lombanupat (sine)" << endl;
cout << "onupat 2: lognanupat (cosine)" << endl;
cout << "onupat 3: sporshanupat (tangent)" << endl
cout << endl;

// তারপর অনুপাত ও কোণ যোগান নেয়া হবে
cout << "onupat: " << endl;    //  যোগান যাচনা
cin >> onupat;                 //  যোগান নেওয়া
cout << "kone: " << endl;    //  যোগান যাচনা
cin >> kone;                 //  যোগান নেওয়া

// পল্টি ব্যাপার ব্যবহার করে ফলন দেখানো হবে
switch(onupat)  // এখানে চলক না হয়ে কোন রাশিও হতে পারে
{
  case 1:      //  লম্বানুপাত (sine) cmath শিরনথি লাগবে
    cout << "lombanupat: " << sin(kone) << endl;
    break;
  case 2:      //  লগ্নানুপাত (cosine) #include <cmath>
    cout << "lognanupat: " << cos(kone) << endl;
    break;
  case 3:      // স্পর্শানুপাত (tangent) cmath শিরনথি লাগবে
    cout << "sporshanupat: " << tan(kone) << endl;
    break;
  default:     // অগত্যা ত্রুটি বার্তা (error)
    cout << "osomorthito posondo" << endl;
    break;
}

cout << "kee chomotkar!" << endl; // পল্টির বাইরে অন্য কিছু
```

উপরের ক্রমলেখ (program) খেয়াল করো। যেমন বলা হয়েছে তেমন করে দুটি চলক নেয়া হয়েছে: কোন অনুপাত তা রাখার জন্য চলক onupat আর কত রেডিয়ান কোন তা রাখার জন্য চলক kone। এরপর একটা প্রাপণ্য (menu) দেখানো হয়েছে, কোন সংখ্যা দিয়ে কোন অনুপাত বুঝানো হচ্ছে সেটা ব্যবহারকারীকে জানানোর জন্য: 1 দিলে লম্বানুপাত (sine), 2 দিলে লগ্নানুপাত (cosine), 3 দিলে স্পর্শানুপাত (tangent)। এরপরে অনুপাত ও কোণ যোগান (input) নেয়ার জন্য প্রথমে যোগান যাচনা (input prompt) করে তারপর যোগান নেওয়া হয়েছে। তারপর মুল অংশ যেখানে পল্টি ব্যাপার (switch case) ব্যবহার করে যে অনুপাত চাওয়া হয়েছে সেটি দেখানো হবে। পল্টি-ব্যাপারের পরে আছে অন্য কিছু ক্রমলেখয়ের বাঁকী অংশ।

আমরা কেবল পল্টি ব্যাপার (switch case) অংশে নজর দেই। যেহেতু onupat চলকটির (variable) মান ওপর নির্ভর করবে আমরা কোন অনুপাত ফলনে (output) দেখাবো, আমরা তাই লিখেছি switch(onupat) আর তারপর আমাদের একটি মহল্লা (block) তৈরী করতে হবে { } বাঁকা বন্ধনী (curly brackets) যুগল দিয়ে। এবার অনুপাতের মান কত হলে কী করতে হবে তার সবকিছু আমরা রাখবো মহল্লার ভিতরে। খেয়াল করো onupat এর মান 1, 2, 3 হওয়ার জন্য আমাদের তিনটি ব্যাপার (case) আছে যেমন case 1: case 2: case 3:; খেয়াল করো প্রথমে case তারপরে onupat চলকটির কোন মান সেটি তারপর একটা : দোঁটা (colon)। প্রতিটি ব্যাপারের (case) পরে দেখো আমরা cout দিয়ে ত্রিকোনমিতির অনুপাত sine, cosine, tangent ব্যবহার করে ফলন দেখিয়েছি। তারপর লিখেছি break; অর্থাৎ এইখানে পল্টি-ব্যাপারের ক্ষান্তি (break) ঘটবে। এই ক্ষান্তিয়ের (break) কাজ আমরা একটু পরেই আলোচনা করছি। তার আগে দেখো case 3: এর ক্ষান্তিয়ের (break) পরে রয়েছে default: যেটি হলো অগত্যা ব্যাপার অর্থাৎ ওপরের কোন case এর সাথেই onupat এর মান না মিললে অগত্যা ব্যাপারটি ঘটবে বলে ধরে নেয়া হবে। তাহলে onupat এর মান যদি 1, 2, 3 ভিন্ন অন্য কিছু হয় তাহলে default: অগত্যা ব্যাপারটি ঘটবে। যথারীতি সেখানে আমরা ত্রুটিবার্তা (error message) দেখিয়েছি। এখানে কিন্তু break; আছে শেষে।

ক্রমলেখ নির্বাহ (program execution) করার সময় ধরে নিতে পারো অদৃশ্য বোতামের মতো একটা ব্যাপার আছে যেটাকে বলা হয় নিয়ন্ত্রণ (control)। এই নিয়ন্ত্রণ বোতামটি ক্রমলেখ নির্বাহের শুরুতে main বিপাতকের একদম প্রথম সারিতে থাকে। বোতামটি যেই সারিতে থাকে সেই সারি নির্বাহিত (executed) হয়। আর তারপর নিয়ন্ত্রণ বোতামটি পরের সারিতে লাফ দেয়, তখন সেই সারিটি নির্বাহিত হয়। এভাবে নিয়ন্ত্রণ বোতামের লাফালাফি ও সেই সাথে সংশ্লিষ্ট সারির নির্বাহ একে একে চলতে থাকে। যদি-নাহলে (if-else) আলোচনা করার সময় আমরা বলেছিলাম শর্ত সত্য হলে কিছু কাজ হয় আবার শর্ত মিথ্যা হলে অন্য কিছু কাজ হয়। ঠিক যেন দুটো শাখা (branch) তৈরী হয়। শর্তের ওপর নির্ভর করে নিয়ন্ত্রণ বোতামটি আসলে হয় এই শাখায় নাহয় ওই শাখায় গিয়ে লাফ দিয়ে বসে। নিয়ন্ত্রণ যে শাখায় বসে সেই শাখা নির্বাহিত হয়, অন্য শাখা নির্বাহিত হয় না। নিয়ন্ত্রণ বোতাম এরপর if-else এর পরের অংশে চলে যায়।

পল্টি-ব্যাপারের (switch-case) ক্ষেত্রে বলতো নিয়ন্ত্রণ switch(onupat) এর পরে লাফ দিয়ে কোন সারিতে গিয়ে বসবে? যদি onupat এর মান হয় 1 তাহলে গিয়ে বসবে case 1: এর সারিতে, 2 হলে গিয়ে বসবে case 2: এর সারিতে, আর 3 হলে বসবে case 3: এর সারিতে, আর তিনটের কোনটাই না হলে গিয়ে বসবে default: এর সারিতে। নিয়ন্ত্রন switch(onupat) হতে লাফ দিয়ে গিয়ে সংশ্লিষ্ট ব্যাপারে (case) বসার পরে সারির পর সারি একে একে যেতে থাকবে যতক্ষণনা একটি break; পাচ্ছে। অর্থাৎ break পাওয়ার আগে পর্যন্ত প্রত্যকটি সারিই একের পর এক নির্বাহিত হতে থাকবে। আর break; পাওয়ার পরেই নিয়ন্ত্রণ আর একটি লাফ দিয়ে পল্টি-ব্যাপারের (switch-case) মহল্লার বাইরে চলে যাবে। ক্ষান্তি না দিলে কী ঘটবে আমরা সেটা পরবর্তীতে আলোচনা করবো। তবে বলে রাখি প্রতিটি ব্যাপারের (case) শেষে আসলে ক্ষান্তি (break) দেয়াটা আসলেই খুব গুরুত্বপূর্ণ, আর আমরা আবার না দেয়ার ভুলটা প্রায়ই করি।

৭.১৮ অন্তান্তি পল্টি ব্যাপার (Nested Switch Cases)

অন্তান্তি পল্টি ব্যাপার (nested switch case) ব্যবহার করে এমন একটি ক্রমলেখ (program) রচনা করো, যেটি প্রথমে প্রাপণ্য (menu) দেখিয়ে জানতে চাবে আমরা বর্গের হিসাব করতে চাই, নাকি বৃত্তের হিসাব করতে চাই। সেটি যোগান (input) নেবার পরে আমাদের পছন্দ বর্গ হলে ক্রমলেখটি যোগান নিবে দৈর্ঘ্য আর কী দেখতে চাই ক্ষেত্রফল নাকি পরিসীমা তা, আর সেই অনুযায়ী ফলন (output) দেখাবে। আর আমাদের পছন্দ বৃত্ত হলে ক্রমলেখটি ব্যাসার্ধ যোগান নিবে আর নিবে ক্ষেত্রফল নাকি পরিধি দেখতে চাই তা, আর সে অনুযায়ী ফলন দিবে।

নীচের ক্রমলেখ (program) খেয়াল করো। প্রথমে আকৃতির প্রাপণ্য (menu) দেখানো হয়েছে। তারপর akriti চলক ঘোষণা (variable declare) করে যোগান যাচনা (input prompt) করে যোগান (input) নেয়া হয়েছে। এরপর akriti চলকের মানের ওপর পল্টি (switch) যাতে তিনটি ব্যাপার (case) আছে। চলক akriti এর মান 1 হলে case 1: বর্গ, 2 হলে case 2: বৃত্ত, আর অন্য কিছু হলে অগত্যা ব্যাপারে default: ত্রুটি বার্তা দেখানো হয়েছে।

ফিরিস্তি ৭.৬: অন্তান্তি পল্টি দিয়ে প্রাপণ্য (Menu with Nested Switch)

```
// আকৃতির প্রাপণ্য (menu)
cout << "akriti 1 borgo" << endl;
cout << "akriti 2 britto" << endl;

int akriti;          // চলক ঘোষণা
cout << "akriti: "; // যোগান যাচনা
cin >> akriti;      // যোগান নেওয়া

// বাইরের পল্টি যার ভিতরে আবার পল্টি থাকবে
switch(akriti)      // আকৃতির পল্টি
{
  case 1:     // বাইরের পল্টি বর্গ হলে
    //  কী পছন্দ তা দেখানো হবে
    cout << "posondo britto" << endl;

    // বর্গের দৈর্ঘ্য যোগান নিতে হবে
    int doirgho;          // চলক ঘোষণা
    cout << "doirgho: ";  // যোগান যাচনা
    cin >> doirgho;       // যোগান নেওয়া

    // কী চাই তার প্রাপণ্য (menu)
    cout << "1 chai khetrofol" << endl;
    cout << "2 chai porishima" << endl;

    int keechai;          // চলক ঘোষণা
    cout << "kee chai: "; // যোগান যাচনা
    cin >> keechai;       // যোগান নেওয়া
```

```
        // ভিতরের পল্টি যেটি আরেকটি পল্টির ভিতরে
        switch(keechai)         // পল্টি কী চাই
        {
          case 1:       // ভিতরের পল্টি ক্ষেত্রফল হলে
            cout << "khetrofol: ";
            cout << doirgho * doirgho;
            cout << endl;
            break;

          case 2:       // ভিতরের পল্টি পরিসীমা হলে
            cout << "porishima: ";
            cout << 4*doirgho;
            cout << endl;
            break;

          default:      // ভিতরের পল্টি অন্য কিছু হলে ত্রুটিবার্তা
            cout << "osomorthito posondo" << endl;
            break;
        }
        //  এটি ভিতরের পল্টি থেকে বাইরে
        cout << "borger hisab shes" << endl;
        break;

      case 2:       // ভিতরের পল্টি বৃত্ত হলে
        //  কী পছন্দ তা দেখানো হবে
        cout << "posondo britto" << endl;

        // বৃত্তের ব্যাসার্ধ যোগান নিতে হবে
        int bashardho;          // চলক ঘোষণা
        cout << "bashardho: ";
        cin >> bashardho;

        // কী চাই প্রাপণ্য
        cout << "1 chai khetrofol" << endl;
        cout << "2 chai poridhi" << endl;

        int chaokee;            // চলক ঘোষণা
        cout << "chao kee: "; // যোগান যাচনা
        cin >> chaokee;         // যোগান নেওয়া

        // ভিতরের পল্টি যেটি আরেকটি পল্টির ভিতরে
        switch(chaokee)         // কী চাই পল্টি
        {
```

```
        case 1:      // ভিতরের পল্টি ক্ষেত্রফল হলে
          cout << "khetrofol: ";
          cout << 3.1416 * bashardho * bashardho;
          cout << endl;
          break;

        case 2:      // ভিতরের পল্টি পরিধি হলে
          cout << "poridhi: ";
          cout << 2 * 3.1416 * bashardho;
          cout << endl;
          break;

        default:     // ভিতরের পল্টি অন্য কিছু হলে ত্রুটিবার্তা
          cout << "osomorthito posondo" << endl;
          break;
      }
      //  এটি ভিতরের পল্টি থেকে বাইরে
      cout << "britter hisab shes" << endl;
      break;

    default:       // বাইরের পল্টি অন্য কিছু হলে ত্রুটিবার্তা
      cout << "osomorthito posondo" << endl;
      break;
  }

  //   বাইরের পল্টিরও বাইরে
  cout << "kee chomotkar!" << endl;
```

যখন akriti এর মান 1 অর্থাৎ বর্গ বেছে নেয়া হয়েছে তখন প্রথমে ফলনে (output) দেখানো হয়েছে যে বর্গ পছন্দ করা হয়েছে। তারপর চলক doirgho ঘোষনা (declare) করে যোগান যাচনা (input prompt) করে যোগান (input) নেওয়া হয়েছে। তারপর বর্গের কী জানতে চাই তার জন্য আরেকটি প্রাপণ্য (menu) দেখানো হয়েছে, যেখানে ক্ষেত্রফল নাকি পরিসীমা চাই সেটা দেখানো হয়েছে। ব্যবহারকারীর পছন্দ যোগান নেয়ার জন্য এখানেও keechai নামে একটি চলক ঘোষনা করে যোগান যাচনা করে মান যোগান নেয়া হয়েছে। তারপর চলক keechai এর মানের ওপর নির্ভর করে আরেকটি পল্টি ব্যাপার (switch case) ব্যবহার করে ক্ষেত্রফল বা পরিসীমা ফলনে (output) দেখানো হয়েছে। এই পল্টি ব্যাপারটি আগের পল্টি-ব্যাপারের ভিতরে, আর তাই এই ভিতরেরটিকে বলা হবে অন্তান্তি পল্টি ব্যাপার (nested switch case)।

যখন akriti এর মান 2 অর্থাৎ বৃত্ত বেছে নেয়া হয়েছে তখন প্রথমে ফলনে (output) দেখানো হয়েছে যে বৃত্ত পছন্দ করা হয়েছে। তারপর চলক bashardho ঘোষনা (declare) করে যোগান যাচনা (input prompt) করে যোগান (input) নেওয়া হয়েছে। তারপর বৃত্তের কী জানতে চাই তার জন্য আরেকটি প্রাপণ্য (menu) দেখানো হয়েছে, যেখানে ক্ষেত্রফল নাকি পরিধি চাই সেটা দেখানো হয়েছে। ব্যবহারকারীর পছন্দ যোগান নেয়ার জন্য এখানেও chaokee নামে একটি চলক ঘোষনা করে যোগান যাচনা করে মান যোগান নেয়া হয়েছে। বর্গের ক্ষেত্রে ব্যবহৃত চলক keechai থেকে ভিন্ন একটি নাম নেয়ার জন্যই মুলত নাম দেওয়া হয়েছে chaokee। এই দুটো

চলকই বাইরের পল্টি-ব্যাপারের যে মহল্লা (block) তার ভিতরে। একই মহল্লায় দুটো চলকের (variable) নাম একই হতে পারে না। আর সে কারণে নামের এই ভিন্নতা, যদিও তাদের উদ্দেশ্য এখানে একই রকম। যাইহোক, চলক chaokee এর মানের ওপর নির্ভর করে এরপর আরেকটি পল্টি ব্যাপার ব্যবহার করে ক্ষেত্রফল বা পরিধি ফলনে (output) দেখানো হয়েছে। এই পল্টি ব্যাপারটি (switch case) বর্গের পল্টি-ব্যাপারের মতোই বাইরের পল্টি ব্যাপারটির ভিতরে, তাই এটিও একটি অন্তান্তি পল্টি ব্যাপার (nested switch case)।

এই পর্যায়ে জিজ্ঞেস করতে পারো, break; পাওয়া মাত্র নিয়ন্ত্রণ (control) সেই পল্টি ব্যাপার (switch case) থেকে বের হয়ে আসে বলে আমরা জানি, তো ভিতরের পল্টি ব্যাপার থেকে break; পেলে কোথায় যাবে? উত্তর হচ্ছে ভিতরের পল্টি ব্যাপার থেকে বের হয়ে যেখানে আসবে সেটা কিন্তু বাইরের পল্টির মহল্লা। ভিতরের পল্টি থেকে বের হয়ে কোথায় আসবে সেটা বুঝার জন্য বর্গের পল্টি ব্যাপারের বাইরে cout << " borger hisab shes" << endl; আর বৃত্তের পল্টি ব্যাপারের বাইরে cout << "britter hisab shes" << endl; লেখা হয়েছে। আর বাইরের পল্টি ব্যাপারের বাইরে লেখা হয়েছে cout << "kee chomotkar!" << endl;।

৭.১৯ পল্টি ব্যাপার ক্ষান্তি (Switch Cases Breaks)

পল্টি ব্যাপারে (switch cases) ক্ষান্তি (break) না দিলে কী ঘটে, আর ক্ষান্তি না দেওয়া কোথায় কাজে লাগতে পারে? যথাযথ উদাহরণ সহ ক্রমলেখ (program) লিখে দেখাও।

ধরো তোমার একজন অতিথি আসবে। সে যদি সকাল ১০ বা ১১টায় আসে তাকে তোমার সকালে নাস্তা, দুপুরের খাবার, আর বিকালের নাস্তা খাওয়াতে হবে। আর সে যদি ১২টায় বা ১৩টায় আসে তবে তাকে কেবল দুপুরের খাবার ও বিকালের নাস্তা খাওয়াতে হবে, আর তিনি যদি ১৪টা বা ১৫টায় আসে তাহলে তাকে কেবল বিকালের নাস্তা খাওয়াতে হবে। এই সময়গুলো ভিন্ন অন্য কোন সময়ে যদি সে আসে তাহলে তাকে কিছুই খাওয়ানোর দরকার নাই।

```
switch(somoy)
{
  case 10:
  case 11:
    cout << "sokaler nasta" << endl;
  case 12:
  case 13:
    cout << "dupurer khabar" << endl;
  case 14:
  case 15:
    cout << "bikaler nasta" << endl;
}
```

উপরের ক্রমলেখতে আমরা ক্ষান্তি (break) ছাড়া পল্টি ব্যাপার (switch case) লিখে ক্রমলেখ (program) তৈরী করেছি। এখানে চলক somoy এ আমরা অতিথির আসার সময় রাখবো, সেটা যোগান (input) নেয়া হয়ে থাকতে পারে, বা কোন ভাবো আরোপিত (assigned) হয়ে থাকতে পারে। সাধারণত পল্টিতে যে ব্যাপারটার সাথে মিলে যায় সেখান থেকে বিবৃতিগুলো (statement) নির্বাহিত হতে শুরু করে আর ক্ষান্তি (break) পাওয়া পর্যন্ত চলে। আর একবার কোন ব্যাপারের সাথে মিলে গেলে পরে আর কোন ব্যাপারের সাথে মিলানোর চেষ্টা করাও হয় না, বরং ক্ষান্তি (break) না পাওয়া পর্যন্ত ক্রমাগত বিবৃতিগুলো নির্বাহিত হতে থাকে।

খেয়াল করো উপরের ক্রমলেখতে (program) সময় যদি ১০টা হয়, ঠিক সেখানে কিছু না থাকলেও পরপর যে বিবৃতিগুলো আছে সেগুলো একে একে নির্বাহিত হবে, ফলে যেমন sokaler nasta, dupurer khabar, bikaler nasta সবগুলো একে একে ফলনে আসতে থাকবে। সময় যদি ১১টা হয় তাহলেও একই ঘটনা ঘটবে। সময় যদি ১২ টা হয়, তাহলে sokaler nasta ফলনে আসবে না, কিন্তু dupurer khabar ও bikaler nasta একে একে আসতে থাকবে। পরের সময়গুলোর জন্যেও একই রকমের কথাবার্তা প্রযোজ্য।

আর একটা বিষয় খেয়াল করো, উপরের পল্টিতে (switch) আমরা অগত্যা ব্যাপার default : দেই নাই। ফলে সময় যদি তালিকায় না থাকে তাহলে সেটি কোন ব্যাপারের (case) সাথেই মিলবে না, আর এতে ফলনে (output) কিছুই আসবে না। আসলে পল্টিতে (switch) অগত্যা (default) ব্যাপার দিতেই হবে এমন কোন কথা নেই, দরকার না লাগলে দিবে না।

```
switch(nombor)
{
  case 4:
  case 0:
  case 2:
    cout << "jor" << endl;
    break;
  case 1:
  case 5:
  case 3:
    cout << "bejor" << endl;
    break;
}
```

এবার কিছু প্রশ্ন: পল্টিতে কী ব্যাপারগুলো মানের ক্রমানুসারেই থাকতে হবে? মানগুলো কী ধারাবাহিকভাবে পরপর সংখ্যা হতে হবে? উভয় প্রশ্নের উত্তর হচ্ছে "না"। কাজেই ঠিক উপরের উদাহরণের মতো তুমি দরকার মতো ব্যাপারগুলো (case) পরপর না হলেও বা উল্টোপাল্টা ক্রমে হলেও লিখতে পারবে। আবার দেখো কিছু ব্যাপারে (case) ক্ষান্তি (break) নাই, আবার কিছু ব্যাপারে আছে। মোট কথা যেখানে ক্ষান্তি দেয়া দরকার সেখানে break; না দরকার হলে নাই।

আরো কিছু প্রশ্ন: পল্টিতে (switch) ব্যাপারগুলো (case) কী পূর্ণক (integer) ছাড়া ভগ্নক (fractioner) হতে পারবে? আর switch() এ চলক (variable) ছাড়া অন্য কিছু ব্যবহার করা যাবে? তুমি কোন ভগ্নক (fractioner) ব্যাপার হিসাবে ব্যবহার করে দেখতে পারো, তাতে সংকলনে (compile) ত্রুটি বার্তা (error message) দেখাবে, তার মানে হলো পারবে না। আর switch(nombor) এখানে switch এ যে কেবল চলক হতে হবে তা নয়, যে কোন রাশি যেটি পূর্ণক ফলাফল দেয় সেটিই তুমি ব্যবহার করতে পারো, যেমন নীচের উদাহরণ দেখো, আমরা ২ দিয়ে ভাগশেষের ওপর পল্টি ব্যবহার করছি। ভাগশেষ ০ হলো জোড়, আর ১ হলে বিজোড়।

```
switch(nombor % 2)
{
  case 0: cout << "jor" << endl; break;
  case 1: cout << "bejor" << endl; break;
}
```

পল্টিতে অবশ্য তুমি একই ব্যাপার দুইবার ব্যবহার করতে পারবে না, যেমন case 1: লিখে একই পল্টির ভিতরে পরে আবার case 1: লিখতে পারবে না। তবে পল্টির ভিতরে অন্তান্তি (nested) পল্টি থাকলে সেখানে case 1: থাকতেই পারে।

৭.২০ পল্টি ব্যাপার যদি-নাহলে (Swtich Cases If Else)

পল্টি ব্যাপার (switch cases) ব্যবহার না করে যদি নাহলে (if else) ব্যবহার করলেই তো হয়। তাহলে পল্টি ব্যাপার কোথায় ব্যবহার করবো, আর কোথায় যদি নাহলে ব্যবহার করবো?

```
switch(nombor)
{
  case -2:
  case -1:
    cout << "rinatok" << endl;
    break;

  case 0:
    cout << "shunyo" << endl;
    break;

  case 1:
  case 2:
    cout << "dhonatok" << endl;
    break;
}
```

উপরের উদাহরণটি খেয়াল করো। এখানে আমরা একটি নম্বর ধনাত্নক (positive), ঋণাত্নক (negative), নাকি শুন্য (zero) নির্ণয় করতে চাই। আমরা যদি আগে থেকে জানি যে নম্বরটি কেবল $-2, -1, 0, 1, 2$ এই পাঁচটি নির্দিষ্ট সংখ্যার একটি হতে পারবে, অন্য আর কিছু নয়, এগুলোর বাইরে নয়, কেবল তাহলেই আমরা উপরের মতো করে পল্টি ব্যাপার (switch case) ব্যবহার করতে পারবো। আবার চাইলে আমরা নীচের মতো করে সমতুল আরেকটি ক্রমলেখও লিখতে পারবো, যেখানে আমরা পল্টি ব্যাপার ব্যবহার না করে যদি নাহলে (if else) ব্যবহার করবো। যদি না হলে ব্যবহার করে অবশ্য আরো নানা ভাবেই এটি করা সম্ভব, এটি কেবল একটা উদাহরণ।

```
if (nombor == -2 || nombor == -1)
    cout << "rinatok" << endl;
else if (nombor == 1 || nombor == 2)
    cout << "dhonatok" << endl;
else // if (nombor == 0)
    cout << "shunyo" << endl;
```

কিন্তু আমাদের নম্বরটি যদি উপরের ওই পাঁচটি সংখ্যার বাইরে অনির্দিষ্ট সংখ্যক নম্বরগুলোর একটি হয়, অথবা অনেক অনেক বেশী সংখ্যকের একটি হয়, তাহলে ঠিক পল্টি ব্যবহার করে আমরা সামলাতে পারবো না। কারণ এ সব ক্ষেত্রে ব্যাপারের সংখ্যা (number of cases) হবে অনেক বেশী বা অসংখ্য। আর একটি ব্যাপার হলো পল্টিতে ব্যাপারগুলো মুলত মান সমান == হলে কী

হবে তার ওপর ভিত্তি করে তৈরী, অন্য কোন ধরনের তুলনা যেমন বড় >, ছোট < ইত্যাদি ব্যবহার করা যায় না। ফলে পল্টি (switch) সাধারণত ব্যবহার করা হয় অল্প কিছু সংখ্যক ও সুনির্দিষ্ট সংখ্যক ব্যাপারের ক্ষেত্রে, আর এ সব ক্ষেত্রে ক্রমলেখ পড়া সহজ হয়ে যায়। অন্যান্য সকল ক্ষেত্রে সাধারণত যদি নাহলে (if else) ব্যবহার করা হয় কারণ যদি নাহলের সাথে যে কোন শর্ত বা সংযোজক (connectives) &&, ||, ! ব্যবহার করে আরো জটিল শর্ত ব্যবহার করা যায়।

```
if (nombor < 0)
    cout << "rinatok" << endl;
else if (nombor > 0)
    cout << "dhonatok" << endl;
else // if (nombor == 0)
    cout << "shunyo" << endl;
```

৭.২১ ব্যাপীয় ও স্থানীয় চলক (Global & Local Variables)

স্থানীয় চলক (local variable) কী? এর বিপরীতে ব্যাপীয় চলকই (global variable) বা কী? শর্তালী পরিগণনায় (conditional programming) স্থানীয় চলকের ব্যবহার দেখাও।

যখন কোন চলক বা ধ্রুবক বাঁকা বন্ধনী যুগলের বাইরে অর্থাৎ যে কোন মহল্লার বাইরে থাকে তখন তাকে ব্যাপীয় চলক (global variable) বা ব্যাপীয় ধ্রুবক (global constant) বলা হয়। নীচের ক্রমলেখতে (program) খেয়াল করো pai আর nimnoshima যে কোন মহল্লার (block) বাইরে, তাই এগুলো যথাক্রমে ব্যাপীয় ধ্রুবক (global constant) ও ব্যাপীয় চলক (global variable)। ব্যাপীয় চলক বা ধ্রুবক ঘোষণা করার পর থেকে ক্রমলেখয়ের যে কোন জায়গায় ব্যবহার করা যায়। যে কোন ধ্রুবকের মান তো ঘোষণার সময় অবশ্যই দিতে হয়, ব্যাপীয় ধ্রুবকের (global constant) মানও ঘোষণার সময়ই দিয়ে দিতে হয়। আর ব্যাপীয় চলকের মান ঘোষণার সময় না দিয়ে দিলে এতে অগত্যা (default) শুন্য থাকে।

ফিরিস্তি ৭.৭: স্থানীয় ও ব্যাপীয় চলকরে ব্যবহার (Using Local & Global Variables)

```
#include <iostream>
#include <cstdlib>

using namespace std;

float const pai = 3.1416; // ব্যাপীয় ধ্রুবক, মান দিতেই হবে
float nimnoshima = 1.00;  // ব্যাপীয় চলক, মান না দিলে শুন্য

int main(void)
{
  float bashardho;        // স্থানীয় চলক
  float const two = 2.0;  // স্থানীয় ধ্রুবক

  cout << "bashardho: ";
  cin >> bashardho;
```

```
  if (bashardho < nimnoshima)
  {
    cout << "nimnoshimar cheyeo kom" << endl;
    return EXIT_FAILURE;
  }

  float poridhi = two * pai * bashardho;  // স্থানীয় চলক
  cout << "poridhi: " << poridhi << endl;

  return EXIT_SUCCESS;
}
```

যখন কোন চলক (variable) বা ধ্রুবক (constant) কোন বাঁকা বন্ধনী যুগল {} বা মহল্লার (block) ভিতরে ঘোষিত হয় তখন তাকে স্থানীয় চলক (local variable) বা স্থানীয় ধ্রুবক (local constant) বলা হয়। উপরের ক্রমলেখতে (program) খেয়াল করো চলক bashardho এবং ধ্রুবক two উভয়ই main বিপাতকের মহল্লার ভিতরে ঘোষিত হয়েছে, কাজেই এ দুটো যথাক্রমে স্থানীয় চলক ও স্থানীয় ধ্রুবক। স্থানীয় চলক বা ধ্রুবক যে কোন মহল্লা (block) বা উপমহল্লার (subblock) ভিতরে ঘোষিত হতে পারে। মহল্লার ভিতরে আবার মহল্লা থাকলে ভিতরের মহল্লাকে উপমহল্লা (subblock) বলা হয় আর বাইরের মহল্লাকে বলা হয় অধিমহল্লা (superblock)। যে কোন ধ্রুবকের মান তো ঘোষণার সময়ই দিয়ে দিতে হয়, স্থানীয় ধ্রুবকের মানও তাই ঘোষণার সময়ই দিয়ে দিতে হবে। আর স্থানীয় চলকের মান দিয়ে না দিলে এটাতে উল্টা পাল্টা একটা মান থাকবে। সুতরাং স্থানীয় চলক ব্যবহারের পূর্বে অবশ্যই এতে মান আরোপণ (assign) করে নিতে হবে। স্থানীয় চলক ও ধ্রুবক ঘোষণা করার পর থেকে ওই মহল্লার ভিতরে যে কোন খানে ব্যবহার করা যায়, এমনকি উপমহল্লা বা উপউপমহল্লার ভিতরেও ব্যবহার করা যায়।

```
int cholok = 2; // ব্যাপীয় চলক

int main()
{
  cout << cholok << endl;    // ব্যাপীয় চলকের মান 2

  int cholok = 3;            // এখন থেকে স্থানীয় চলক
  cout << cholok << endl;    // স্থানীয় চলকের মান 3
  {
    cout << cholok << endl; // অধিমহল্লার চলক মান 3

    int cholok = 5;          // উপমহল্লার স্থানীয় চলক
    cout << cholok << endl; // উপমহল্লার স্থানীয় চলক মান 5
  }
  cout << cholok << endl;    // স্থানীয় চলকের মান 3

  // অন্যান্য কিছু এখানে থাকতে পারে, আমরা লিখছি না
}
int kisuekta = cholok;     // ব্যাপীয় চলকের মান 2
```

অনেক সময় একটি স্থানীয় (local) চলক বা ধ্রুবকে নাম একটি ব্যাপীয় (global) চলক বা ধ্রুবকের নামের সাথে মিলে যেতে পারে। প্রথম কথা প্রতিটি চলক বা ধ্রুবকের নাম পুরো ক্রমলেখ জুড়ে একক (unique) হওয়া উচিৎ, কিন্তু সুবিধার বিচারে অনেক সময় সেটা করা সম্ভব হয় না। এমতাবস্থায় কী করে বুঝবো ব্যবহৃত চলক বা ধ্রুবকটি ব্যাপীয় না স্থানীয়? উপরের ক্রমলেখ (program) খেয়াল করো, সেখানে cholok নাম বারবার ব্যবহার করে অনেগুলো চলক ঘোষণা করা হয়েছে, যার একটি সকল মহল্লার (block) বাইরে তাই ব্যাপীয় (global) আর অন্যগুলো কোন না কোন মহল্লার ভিতরে তাই স্থানীয় চলক। এখন cholok নামের চলককে নানান খানে ফলনে (output) দেখানো হয়েছে। কথা হচ্ছে নাম যেহেতু একই, তো আমরা নামটি দিয়ে কখন কোন চলকটিকে বুঝবো, কখন কোন মানই বা ফলনে দেখতে পাবো?

খেয়াল করে দেখো যেখানে ব্যাপীয় চলকটি ঘোষনা করা হয়েছে আর মান দেওয়া হয়েছে 2 তারপর থেকে এটির কার্যকারীতা বলবৎ আছে, মহল্লার বাইরে তো অবশ্যই আছে যেমন একদম নীচে যেখানে int kisuekta = cholok; লেখা হয়েছে। আবার মহল্লার (block) ভিতরে স্থানীয় চলক ঘোষণার আগে পর্যন্ত এটির কার্যকারীতা রয়েছে ফলে আমরা ব্যাপীয় চলকটির মানটিই অর্থাৎ 2ই দেখতে পাবো। তারপর মহল্লার ভিতরে যখন একই নাম দিয়ে একটি চলক ঘোষনা করা হয়েছে আর মান দেওয়া হয়েছে 3, তখন cholok নামের সাথে স্থানীয় এই চলকটির কার্যকারীতা বলবৎ হয়েছে, আর তা জারি আছে মহল্লা শেষ হওয়া পর্যন্ত, তাছাড়া উপমহল্লার ভিতরে একই নামের আরেকটি চলক ঘোষণার আগে পর্যন্তও তা জারি আছে। ক্রমলেখতে (program) টীকাগুলো (comment) খেয়াল করো। কোথায় কোন মান ফলনে আসবে তা দিয়ে আমরা বুঝার চেষ্টা করছি, কোথায় কোন চলকটির কার্যকারীতা বলবৎ আছে। তাহলে কোন নাম কোন চলকটিকে বুঝাই, সেটার জন্য আমাদের দেখতে হবে একই মহল্লার ভিতরে ওই নামের কোন চলক আছে কিনা? যদি থাকে সেই চলকটি কার্যকর আছে। আর একই মহল্লার ভিতরে যদি না থাকে, তাহলে আমরা ঠিক বাইরের মহল্লাটি দেখবো, সেখানে একই নামে কোন চলক আছে কিনা? যদি থাকে সেটা বলবৎ হবে আর তাও না থাকলে তার ঠিক বাইরে আরো কোন মহল্লা আছে কিনা তা দেখবো।

```
int nombor;
cin >> nombor ;

// নীচের vagshesh হলো স্থানীয় চলক
if (int vagshesh = nombor % 3)
  cout << "nisheshe bivajyo" << endl;
else
  cout << "vagshesh " << vagshesh << endl;
```

সিপিপিতে যদি নাহলে (if else) লেখার সময় যদি { } বাঁকা বন্ধনী যুগল ব্যবহার করে কোন মহল্লা (block) তৈরী করা হয়, তাহলে সেই মহল্লার ভিতরে ঘোষিত যে কোন চলক বা ধ্রুবক তো স্থানীয় চলক বা ধ্রুবক হবে। আমরা সেটা আর আলাদা করে দেখাতে চাই না। তবে উপরের ক্রমলেখ খেয়াল করো if (int vagshesh = nombor % 3) লিখেও আমরা vagshesh নামে একটি চলক ঘোষণা করেছি। এই vagshesh নামের চলকটিও একটি স্থানীয় চলক (local variable) হিসাবে পরিগণিত হয়, আর এটা কার্যকর থাকে কেবল যেখানে লেখা হয়েছে সেখান থেকে শুরু হয়ে ওই if else মই (ladder) বা অন্তান্তি (nesting) যতক্ষণ শেষ না হচ্ছে ততক্ষণ পর্যন্ত, এর বাইরে কোন কার্যকারীতা থাকবে না, ফলে ব্যবহার করলে ত্রুটিবার্তা পাবে।

যদি নাহলে (if else) এর ক্ষেত্রে ঘোষিত স্থানীয় (local) চলকটির মতো আমরা পল্টি ব্যাপারের (switch cases) ক্ষেত্রেও একই ভাবে স্থানীয় চলক ঘোষণা করতে পারি। নীচের ক্রমলেখতে (program) খেয়াল করো switch (int vagshesh = nombor % 3) লিখে আমরা একটি স্থানীয় চলক vagshesh ঘোষণা করেছি। এই চলকটির কার্যকারীতাও কেবল ওই পল্টির মহল্লা-

টির ভিতরেই। বাইরে কোথাও এই চলকটিকে ব্যবহার করবার জো নেই। তুমি কিন্তু পল্টির মহল্লা-টির ভিতরে চাইলে আরো স্থানীয় চলক (local variable) ঘোষণা ও ব্যবহার করতেই পারতে। ধ্রুবকের ক্ষেত্রেও একইরকম আলোচনা প্রযোজ্য।

```
int nombor;
cin >> nombor;

// নীচের সারিতে vagshesh স্থানীয় চলক
switch(int vagshesh = nombor % 3)
{
  case 0:
    cout << "shunya " << vagshesh << endl;
    break;
  case 1:
    cout << "ek " << vagshesh << endl;
    break;
  case 2:
    cout << "dui " << vagshesh << endl;
    break;
}
```

৭.২২ অনুশীলনী সমস্যা (Exercise Problems)

ধারণাগত প্রশ্ন: নীচে কিছু ধারণাগত প্রশ্ন রয়েছে। প্রশ্নগুলোর উত্তর নিজে নিজে বের করবে।

১. শর্তালি পরিগণনা (conditional programming) কী? অল্প কথায় আলোচনা করো।

২. যদি if এর সাথের শর্ত মিথ্যা হলে সংশ্লিষ্ট নাহলেতে else গিয়ে আবারও শর্তের বিপরীত শর্তটি সত্য কিনা পরীক্ষা করা দরকার নেই। ব্যাখ্যা করো।

৩. যদি নাহলে (if else) দিয়ে ক্রমলেখ (program) লিখতে ছাড়ন (indentation) দেয়া গুরুত্বপূর্ণ কেন? কার জন্য গুরুত্বপূর্ণ মানুষের জন্য নাকি গণনির (computer) জন্য?

৪. অন্বয়ী অণুক্রিয়া (relational operators) কী? এগুলো কী ধরনের ফলাফল দেয়? সি-পিপিতে থাকা কয়েকটি অন্বয়ী অণুক্রিয়ার উদাহরণ দাও।

৫. যদি নাহলে মইতে (if else ladder) শর্তগুলো কী ভাবে সাজাবে, যদি চিন্তার সুবিধা বি-বেচনা করো অথবা ক্রমলেখয়ের দক্ষতা বিবেচনা করো?

৬. অন্তান্তি যদি নাহলে (nested if else) ও যদি নাহলে মই (if else ladder) একটা থেকে আরেকটিতে রূপান্তর সম্ভব, উদাহরণ সহ ব্যাখ্যা করো।

৭. ঝুলন্ত নাহলে (dangling else) সমস্যাটি কী? এটির সমাধান কী কী ভাবে করা যেতে পারে, উদাহরণ দিয়ে আলোচনা করো।

৮. শুন্য বিবৃতি (empty statement) কী? কত ভাবে শুন্য বিবৃতি দেওয়া যায়?

৯. বুলক সংযোজকগুলো (Boolean connectives) কী কী, কী ভাবে ফলাফল দেয়?

১০. পূর্ণক (integer) ও ভগ্নক (fractioner) কে সরাসরি বুলক হিসাবে কী ভাবে ব্যবহার করা যায় আলোচনা করো। এতে কী সুবিধা হয়?

১১. বুলক শর্তের (Boolean condition) আংশিক মূল্যায়ন কী ও কী ভাবে কাজ করে?

১২. একাধিক ব্যাপীয় চলক (global variable ও স্থানীয় চলকের (local variable) নাম একই হলে কোনটা কার্যকর তা কী ভাবে নির্ধারিত হয়?

পরিগণনার সমস্যা: নীচে আমরা কিছু পরিগণনার সমস্যা দেখবো। এই সমস্যাগুলো আগে ধৈর্য্য ধরে নিজে নিজে সমাধান করতে চেষ্টা করবে। যখন একেবারেই পারছো না বলে মনে হয় তখনই কেবল সমাধান দেখে নিতে পারো। সমাধানগুলো পরিগণনার প্রশ্নগুলোর শেষে আছে।

১. নীচের ক্রমলেখয়ের (program) ফলন (output) কী তা প্রথমে খাতা কলমে নির্ণয় করো, আর তারপর গণনিতে চালিয়ে তার সাথে মিলাও।

```
int n;  // আদি মান আরোপ করা হয় নি
cout << (n = 4) << endl;
cout << (n == 4) << endl;
cout << (n > 3) << endl;
cout << (n < 4) << endl;
cout << (n = 0) << endl;
cout << (n == 0) << endl;
cout << (n > 0) << endl;
cout << (n && 4) << endl;
cout << (n || 4) << endl;
cout << (!n) << endl;
```

২. নীচের ক্রমলেখতে (program) কিছু গঠনগত (syntactical) ভুল আছে। ভুলটা কোথায় বলে তুমি মনে করো? ভুলটা এমন ভাবে ঠিক করো যাতে এটির ছাড়ন (indentation) দেখে যা করতে চাওয়া হয়েছিল বলে মনে হয়, ক্রমলেখটি (program) অর্থবোধকতায় (semantically) যেন একদম তাই করে।

```
if (x >= y)
  jogfol += x;
  cout << "x boro" << endl;
else
  jogfol += y;
  cout << "y boro" << endl;
```

৩. নীচের ক্রমলেখ (program) চালালে কী ফলন (output) পাওয়া যাবে?

```
int n, k = 5;
n = (100 % k ? k + 1 : k - 1);
cout << "n = " << n << " k = " << k << endl;
```

৪. নীচের ক্রমলেখ (program) চালালে কী ফলন (output) পাওয়া যাবে?

```
int paowagese = 0, gunti = 5;
if (!paowagese || ++ gunti == 0)
  cout << "bipod" << endl;
cout << "gunti = " << gunti << endl;
```

৫. নীচের ক্রমলেখতে, সম্ভবত বলা যায় যে শর্তালি বিবৃতির (conditional statement) একদম প্রথম সারিতেই একটা ভুল আছে। ক্রমলেখটি যে ভাবে লেখা আছে সেরকম অবস্থা-য়ই যদি নির্বাহ (execute) করা হয় তাহলে ফলন (output) কী হবে? আর যেটা করতে চাওয়া হয়েছিল বলে মনে হয় যদি সেটা করা হয় তাহলে ফলন কী হবে?

```
int n = 5;
if (n = 0) // অণুক্রিয়াটি খেয়াল করো
  cout << "n holo shunyo." << endl;
else
  cout << "n shunyo noy" << endl;
cout << "n er borgo " << n * n << endl;
```

৬. নীচের শর্তালি বিবৃতি (conditional statement) তে অনেক অপ্রয়োজনীয় শর্ত আছে। তো সেই অপ্রয়োজনীয় শর্তগুলো বাদ দিয়ে শর্তালি বিবৃতিটি আবার লেখো।

```
float uparjon;
cout << "mashe uparjon koto: ";
cin >> uparjon;

if (uparjon < 0)
  cout << "tomar dena aro barbe." << endl;
else if (uparjon >= 0 && uparjon < 1200)
  cout << "tumi daridro shimar niche." << endl;
else if (uparjon >= 1200 && uparjon < 2500)
  cout << "tumi kinchit socchol aso." << endl;
else if (uparjon >= 2500)
  cout << "tumi jothesto socchol." << endl;
```

৭. যদি ভিন্ন ভিন্ন বার চালানোর সময় ০, ১৫, বা ৭ যোগান (input) দেয়া হয় তাহলে নী-চের ক্রমলেখয়ের (program) ফলন (output) কোন বারে কী হবে । কত যোগান দিলে ফলনে "biroktikor!" আসবে?

```
int n;
cout << "nombor koto: ";
cin >> n;

if (n < 10)
  cout << "10 er choto." << endl;
else if (n > 5)
  cout << "5 er boro." << endl;
else
  cout << "biroktikor!" << endl;
```

৮. নীচের অন্তান্তি যদি নাহলে (nested if else) খেয়াল করো। ছাড়ন (indentation) যে ভাবে দেয়া হয়েছে তাতে মনে হচ্ছে না যা লিখতে চাওয়া হয়েছে তা লেখা হয়েছে।

```
if (n < 10)
  if (n > 0)
    cout << "dhonatok." << endl;
else
  cout << "----------." << endl;
```

যদি n এর মান ৭ বা ১৫ বা -৩ যোগান (input) দেয়া হয় তাহলে ফলন (output) কী হবে? বিবৃতিটির (statement) গঠন (syntax) এমন ভাবে ঠিক করো যাতে ছাড়ন (indentation) দেওয়া থেকে যেমনটি লিখতে চাওয়া হয়েছে বলে মনে হয় ফলনও ঠিক সে রকম আসে। আর সেক্ষেত্রে শুন্যস্থানে কী হবে বলে যৌক্তিক মনে হয় সেটাও ঠিক করো। অন্যদিকে যা লেখা হয়েছে সেটা ঠিকই আছে ধরে নিয়ে কেবল ছাড়নটা (indentation) ঠিক করো, আর তাতে শুন্যস্থানে কী বসানো যথার্থ হবে তাও নির্ণয় করো।

৯. তিনটি সংখ্যা যোগান (input) নিয়ে কোনটি বড়, কোনটি ছোট ফলনে (output) দেখাও।

১০. তিনটি সংখ্যা যোগান (input) নিয়ে তাদের মধ্যে মাঝেরটি ফলনে (output) দেখাও।

১১. তিনটি সংখ্যা যোগান (input) নিয়ে তাদেরকে উর্ধক্রমে সাজিয়ে ফলন (output) দাও।

১২. গণিতে প্রাপ্ত নম্বর যোগান (input) নিয়ে সেটা থেকে বর্ণ মান (letter grade) ফলন দাও। ধরো ৯০ বা বেশী হলে A, ৮০ বা বেশী হলে B, ৭০ বা বেশী হলে C, ৬০ বা বেশী হলে D, ৫০ বা বেশী হলে E, আর তারও কম হলে F বর্ণ মান পাওয়া যায়।

১৩. একটি দ্বিমাত্রিক (two dimensional) বিন্দুর স্থানাঙ্ক দেওয়া আছে, বিন্দুটি চারটি চতু-র্ভাগের (quadrant) ঠিক কোনটিতে পড়বে নির্ণয় করো।

১৪. একটি প্রগমণ ১, ২, ৩, ..., ৯, ১১, ২২, ৩৩, ..., ৯৯ এর ১ম পদ ১, আর ১৮ তম পদ ৯৯। কততম পদ দেখাতে হবে তা যোগান (input) নিয়ে পদটি ফলনে (output) দেখাও।

১৫. তোমাকে -১০০ ও ১০০ এর মধ্যে দুটি সংখ্যা যোগান (input) হিসাবে দেওয়া হবে, তুমি ওই দুটি সংখ্যা সহ তাদের মাঝের সকল সংখ্যার যোগফল ফলনে (output) দেখাও।

১৬. একটি প্রদত্ত বর্ষ অধিবর্ষ কি না তা নির্ণয়ের ক্রমলেখটি (program) তুমি যদি-নাহলে মই (if else ladder) ব্যবহার করে লিখবে। তবে ক্রমলেখটি রচনা করার সময় তোমাকে মনে রাখতে হবে যে এটি ১ থেকে ২০০০ সাল পর্যন্ত প্রতিটি সালের জন্য চালানো হবে। কাজেই তুমি মইয়ের শর্তগুলো এমন ভাবে সাজাবে যাতে ক্রমলেখ দ্রুততম হয়।

১৭. বাংলা বছরের কততম মাস তা যোগান (input) নিয়ে সেই মাসের নাম ও ওই মাসে কত দিন তা ফলনে (output) দেখাও। একাজে পল্টি ব্যাপার (switch cases) ব্যবহার করো।

১৮. কতটা বাজে সেই সময় ঘন্টায় যোগান (input) নিয়ে মাঝরাত (১-২), প্রভাত (৩-৬), সকাল (৭-১১), দুপুর (১২-১৪), বিকাল (১৫-১৭), সন্ধ্যা (১৮-১৯), রাত (২০-২৮) ফলনে দেখাও। একাজে পল্টি ব্যাপার (switch cases) ব্যবহার করো।

১৯. এমন একটি ক্রমলেখ (program) লিখো যেটি ১-৫ পর্যন্ত ক্রম অনুযায়ী পাঁচটা কোমল পানীয়ের (পানি, কোক, স্প্রাইট, ফানটা, পেপসি) নামের তালিকা দেখাবে, তারপর ক্রমিক নম্বর যোগান (input) নিয়ে কোমল পানীয়টির নাম ফলনে (output) দেখাবে। আর ক্রমিক নম্বরটি যদি ১-৫ এর বাইরে হয়, তাহলে সে সংক্রান্ত একটি ত্রুটি বার্তা (error message) দেখাবে। তুমি এই ক্রমলেখটি একবার পল্টি ব্যাপার (switch cases) ব্যবহার করে আবার যদি নাহলে (if else) ব্যবহার করে করো।

২০. একটি সংখ্যার পুরক সংখ্যা নির্ণয় করো। সংখ্যাট এক অঙ্কের হলে তার পুরক সংখ্য ৯ এর সাথে বিয়োগফল, দুই অঙ্কের হলে ৯৯ এর সাথে বিয়োগফল, তিন অঙ্কের হলে ৯৯৯ এর সাথে বিয়োগফল। তিনের চেয়ে বেশী অঙ্কের সংখ্যা যোগান (input) দেওয়া হবে না।

২১. এমন একটি ক্রমলেখ (program) লিখো যেটা ৫ জন লোক যাদের ক্রমিক ১-৫ তাদের কে কতটা করে পরোটা খেয়েছে যোগান (input) নিবে। ক্রমলেখটি তারপর একজনে সর্বোচ্চ কয়টা পরোটা খেয়েছে সেটা ফলনে (output) দেখাবে। আর কোন লোক সর্বোচ্চ সংখ্যক পরোটা খেয়েছে ক্রমলেখটি সেটাও দেখাবে, তবে সর্বোচ্চ পরোটা খাওয়া একাধিক ব্যক্তি থাকলে প্রথমজনের ক্রমিক নম্বর হলেই চলবে, পরের জনদের দরকার নাই।

২২. একজন লোক স্বাভাবিক নিয়ম অনুযায়ী সপ্তাহে ৪০ ঘন্টা কাজ করে, ৪০ ঘন্টার বেশী কাজ করলে অতিরিক্ত সময়টুকুর জন্য স্বাভাবিক নিয়মের চেয়ে ১.৫ গুণ মজুরি পায়। কোন এক সপ্তাহে লোকটি কত ঘন্টা কাজ করেছে আর স্বাভাবিক নিয়মে ঘন্টা প্রতি মজুরি কত তা যোগান (input) নিয়ে ওই সপ্তাহে তার মোট মজুরি কত তা ফলনে (output) দেখাও।

২৩. ধরো তুমি চার টুকরো কাগজ নিয়েছো। তোমার ১ম টুকরোতে লেখা আছে ১, ৩, ৫, ৭, ৯, ১১, ১৩, ২য় টুকরোতে আছে ২, ৩, ৬, ৭, ১০, ১১, ১৪, ১৫, ৩য় টুকরোতে আছে ৪, ৫, ৬, ৭, ১২, ১৩, ১৪, ১৫, ৪র্থ টুকরোতে আছে ৮, ৯, ১০, ১১, ১২, ১৩, ১৪, ১৫। তোমার ক্রমলেখ (program) ব্যবহারকারী মনে মনে একটি সংখ্যা ধরবে, আর সেটি ১ম, ২য়, ৩য়, ৪র্থ টুকরোর কোন কোনটিতে আছে যোগান (input) দিবে, তারপর তোমার ক্রমলেখ ব্যবহারকারী মনে মনে যে সংখ্যাটি ধরেছে সেটি ফলনে (output) দেখাবে। এটি খুব সহজ একটি ব্যাপার। যে যে টুকরোতে সংখ্যাটি আছে ওই টুকরোগুলোর প্রথম সংখ্যাগুলো যোগ করলেই ব্যবহারকারীর সংখ্যাটি পাওয়া যাবে। যেমন ব্যবহারকারীর সংখ্যাটি যদি ১, ৩, ৪ নম্বর টুকরোতে থাকে তাহলে সংখ্যাটি ১ + ৪ + ৮ = ১৩।

পরিগণনা সমাধান: এবার আমরা পরিগণনার সমস্যাগুলোর সমাধান দেখবো। মনে রাখবে সমাধানগুলো দেয়া হয়েছে তুমি যদি একান্তই নিজে নিজে করতে পারছোনা বলে মনে হয়, তখন কেবল একটু সাহায্য যাতে পেতে পারো তাই। কাজেই পারতপক্ষে সমাধান দেখবে না।

১. নীচের ক্রমলেখয়ের (program) ফলন (output) কী তা প্রথমে খাতা কলমে নির্ণয় করো, আর তারপর গণনিতে চালিয়ে তার সাথে মিলাও।

```
int n;  // আদি মান আরোপ করা হয় নি
cout << (n = 4) << endl;
cout << (n == 4) << endl;
cout << (n > 3) << endl;
cout << (n < 4) << endl;
cout << (n = 0) << endl;
cout << (n == 0) << endl;
cout << (n > 0) << endl;
cout << (n && 4) << endl;
cout << (n || 4) << endl;
cout << (!n) << endl;
```

```
4               // আরোপণ হবে 4
1               // মান আসলেই তো 4
1               // কাজেই 3 এর বেশী
0               // 4 এর সমান, কম তো নয়
0               // আরোপন হবে 0
1               // মান 0 এর সমান, সত্য
0               // মান তো 0, বেশী তো নয়
0               // 0 হলো মিথ্যা তাই ফলাফল মিথ্যা
1               // 4 যেহেতু সত্য, তাই ফলাফল সত্য
1               // 0 নিজে মিথ্যা তাই !0 সত্য
```

২. নীচের ক্রমলেখতে (program) কিছু গঠনগত (syntactical) ভুল আছে। ভুলটা কোথায় বলে তুমি মনে করো? ভুলটা এমন ভাবে ঠিক করো যাতে এটির ছাড়ন (indentation) দেখে যা করতে চাওয়া হয়েছিল বলে মনে হয়, ক্রমলেখটি (program) অর্থবোধকতায় (semantically) যেন একদম তাই করে।

```
if (x >= y)
  jogfol += x;
  cout << "x boro" << endl;
else
  jogfol += y;
  cout << "y boro" << endl;
```

উপরের ক্রমলেখতে ছাড়ন দেখে মনে হয় যদির শর্ত সত্য হলে বা মিথ্যা হলে উভয় ক্ষেত্রে ঠিক তাদের পরের দুই সারিতে থাকা বিবৃতিগুলো নির্বাহিত (executed) হবে। কিন্তু একাধিক বিবৃতি যদি নির্বাহ করতে হয় সেক্ষেত্রে আমাদের নীচের ক্রমলেখয়ের মতো করে বাঁকা বন্ধনী দিয়ে যৌগিক বিবৃতি (compound statement) বানিয়ে নিতে হবে। বাঁকা বন্ধনী না দেয়ায় উপরের ক্রমলেখটি সংকলন (compile) করতে গেলে ত্রুটি দেখাবে। ত্রুটি টা

হল সংশ্লিষ্ট if সাপেক্ষে else টা ঠিক জায়গায় নাই। যদি else টা cout << "x boro" << endl; এর আগে থাকে তাহলে গঠনগত (syntactically) ভাবে শুদ্ধ হয়, কিন্তু তাতে অবশ্য আমরা যা করতে চাই তা হতো না।

```
if (x >= y)
{
  jogfol += x;
  cout << "x boro" << endl;
}
else
{
  jogfol += y;
  cout << "y boro" << endl;
}
```

৩. নীচের ক্রমলেখ (program) চালালে কী ফলন (output) পাওয়া যাবে?

```
int n, k = 5;
n = (100 % k ? k + 1 : k - 1);
cout << "n = " << n << " k = " << k << endl;
```

উপরের ক্রমলেখয়ের ফলন নীচে দেখানো হলো। শুরুতে k এর মান আরোপণ (assign) করা হলো 5। তারপর 100 যেহেতু 5 দ্বারা বিভাজ্য তাই 100 % k হবে শুন্য যাহা বুলক (Boolean) হিসাবে ধরলে মিথ্যা, ফলে তিনিক অণুক্রিয়ার (ternerary operator) শেষের অংশ k − 1 অর্থাৎ 4 হবে ফলাফল যা n চলকে (variable) আরোপিত (assign) হবে। সবমিলিয়ে n হলো 4 আর k শুরুতে যা ছিলো তাই অর্থাৎ 5।

```
n = 4 k = 5
```

৪. নীচের ক্রমলেখ (program) চালালে কী ফলন (output) পাওয়া যাবে?

```
int paowagese = 0, gunti = 5;
if (!paowagese || ++gunti == 0)
  cout << "bipod" << endl;
cout << "gunti = " << gunti << endl;
```

উপরের ক্রমলেখয়ের ফলন (output) নীচে দেখানো হলো। চলক paowagese এর মান 0 অর্থ্যাৎ মিথ্যা, ফলে !paowagese হলো সত্য, আর তাই অথবা || এর ফলাফলও সত্য। লক্ষ্য করো এই ফলাফল নির্ধারণে আমাদের কিন্তু || এর পরের অংশ নির্বাহ (execute) করার দরকারই নাই। আংশিক মূল্যায়নের (partial evaluation) কারণে এটি ঘটবে। তাহলে || এর ফলাফল সত্য আসায় ফলনে আসবে "bipod"। আর ++gunti যেহেতু নির্বাহিতই হয় নি, তাই gunti এর মান 5 ই দেখাবো।

```
bipod
gunti = 5
```

৫. নীচের ক্রমলেখতে, সম্ভবত বলা যায় যে শর্তালি বিবৃতির (conditional statement) একদম প্রথম সারিতেই একটা ভুল আছে। ক্রমলেখটি যে ভাবে লেখা আছে সেরকম অবস্থা-য়ই যদি নির্বাহ (execute) করা হয় তাহলে ফলন (output) কী হবে? আর যেটা করতে চাওয়া হয়েছিল বলে মনে হয় যদি সেটা করা হয় তাহলে ফলন কী হবে?

```
int n = 5;
if (n = 0) // অণুক্রিয়াটি খেয়াল করো
  cout << "n holo shunyo." << endl;
else
  cout << "n shunyo noy" << endl;
cout << "n er borgo " << n * n << endl;
```

উপরের ক্রমলেখয়ের ২য় সারিতে আরোপন (assignment) = অণুক্রিয়া ব্যবহার করা হয়েছে, সাধারণতো শর্ত পরীক্ষার জন্য সমান (equal) == অণুক্রিয়া ব্যবহার করা হয়। সুতরাং এটি সম্ভবত একটা ভুল যেটা প্রায়শই আমাদের হয়ে থাকে। যাই হোক ক্রমলেখটি যেমন আছে তেমনি চালালে আরোপণের ফলে n এর মান হবে শুন্য আর আরোপণ অণু-ক্রিয়ার ফলাফলও হবে শুন্য, যা বুলক মান (Boolean value) হিসাবে মিথ্যা। সুতরাং else অংশে থাকা বিবৃতিটুকু নির্বাহিত হবে, আর আমরা ফলনে পাবো n shunyo noy। বিষয়টি কেমন যেন গোলমেলে তাই না, একদিকে n এর মান আসলেই শুন্য, কিন্তু অন্য দিকে ফলন দেখাচ্ছে n শুন্য নয়! যাইহোক n এর মান শুন্য আরোপণের ফলে if else এর পরের সারিতে থাকা cout এর কারণে ফলনে আসবে n er borgo 0। এই ফলনগুলো নীচে বামদিকে দেখানো হলো, আর ডান দিকে রয়েছে আরোপণ (assignment) = না লিখে আমরা যদি সমান (equality) == লিখি তাহলে ফলন (output) কী হবে তা। লক্ষ্য এবারে n এর মান কিন্তু 5ই থাকছে যা আদি আরোপণ করা হয়েছে। ফলে n == 0 মিথ্যা হওয়ায় আগের মতোই n shunyo noy দেখাবে আর পরের সারিতে 5 এর বর্গ হবে 25, কাজেই ফলনে আসবে n er borgo 25।

```
n shunyo noy
n er borgo 0
```

```
n shunyo noy
n er borgo 25
```

৬. নীচের শর্তালি বিবৃতি (conditional statement) তে অনেক অপ্রয়োজনীয় শর্ত আছে। তো সেই অপ্রয়োজনীয় শর্তগুলো বাদ দিয়ে শর্তালি বিবৃতিটি আবার লেখো।

```
float uparjon;
cout << "mashe uparjon koto: ";
cin >> uparjon;

if (uparjon < 0)
  cout << "tomar dena aro barbe." << endl;
else if (uparjon >= 0 && uparjon < 1200)
  cout << "tumi daridro shimar niche." << endl;
else if (uparjon >= 1200 && uparjon < 2500)
  cout << "tumi kinchit socchol aso." << endl;
else if (uparjon >= 2500)
  cout << "tumi jothesto socchol." << endl;
```

অদরকারী শর্তগুলো ছাড়া ক্রমলেখ (program) কেমন হবে তা নীচে দেখানো হলো। যদি uparjon < 0 এই শর্ত মিথ্যা হয়, তাহলে অবশ্যই uparjon >=0 সত্য হবে। কাজেই যদি নাহলে মইতে (if else ladder) নাহলের সাথে যে যদি থাকবে সেখানে uparjon >=0 আবার লেখার কোন দরকার নেই। নিয়ন্ত্রণ (control) ওইখানে যাওয়া মানে ওই শর্ত অবশ্যই সত্য আর এবং (and) && অণুক্রিয়ার (operator) একটি উপাদান (operand) সত্য হলে ফলাফল কেবল দ্বিতীয় উপাদানের ওপর নির্ভর করে। সুতরাং আমরা সরলীকরণ করে কেবল && এর দ্বিতীয় উপাদানটিকেই লিখবো। এই একই ভাবে uparjon >= 1200 আর uparjon >= 2500 লেখার কোন দরকার নাই।

```
float uparjon;
cout << "mashe uparjon koto: ";
cin >> uparjon;

if (uparjon < 0)
  cout << "tomar dena aro barbe." << endl;
else if (uparjon < 1200)  // >= 0 দরকার নেই
  cout << "tumi daridro shimar niche." << endl;
else if (uparjon < 2500)  // >= 1200 দরকার নেই
  cout << "tumi kinchit socchol aso." << endl;
else      // >= 2500 দরকার নেই
  cout << "tumi jothesto socchol." << endl;
```

৭. যদি ভিন্ন ভিন্ন বার চালানোর সময় ০, ১৫, বা ৭ যোগান (input) দেয়া হয় তাহলে নীচের ক্রমলেখয়ের (program) ফলন (output) কোন বারে কী হবে । কত যোগান দিলে ফলনে "biroktikor!" আসবে?

```
int n;
cout << "nombor koto: ";
cin >> n;

if (n < 10)
  cout << "10 er choto." << endl;
else if (n > 5)
  cout << "5 er boro." << endl;
else
  cout << "biroktikor!" << endl;
```

যোগান (input) হিসাবে $0, 15, 7$ দিলে, উপরের ক্রমলেখ কী ফলন (output) দেবে তা নীচে ৩ স্তম্ভে দেখানো হলো। এই ক্রমলেখ n এর কোন মানের জন্যই biroktikor! ফলন দিবে না, নিয়ন্ত্রণ (control) কোন অবস্থাতেই সংশ্লিষ্ট বিবৃতিতে যাবে না। সুতরাং else cout << "biroktikor!" << endl; অংশটুকু পুরোপুরি অদরকারী আর সে কারণে মুছে দেয়া যায়, তাতে ক্রমলেখয়ের বৈশিষ্ট্যে কোন প্রভাব পড়বে না।

```
nombor koto: 0      nombor koto: 15     nombor koto: 7
10 er choto.        5 er boro.          10 er choto.
```

৮. নীচের অন্তান্তি যদি নাহলে (nested if else) খেয়াল করো। ছাড়ন (indentation) যে ভাবে দেয়া হয়েছে তাতে মনে হচ্ছে না যা লিখতে চাওয়া হয়েছে তা লেখা হয়েছে।

```
if (n < 10)
  if (n > 0)
    cout << "dhonatok." << endl;
else
  cout << "_________." << endl;
```

যদি n এর মান ৭ বা ১৫ বা -৩ যোগান (input) দেয়া হয় তাহলে ফলন (output) কী হবে? বিবৃতিটির (statement) গঠন (syntax) এমন ভাবে ঠিক করো যাতে ছাড়ন (indentation) দেওয়া থেকে যেমনটি লিখতে চাওয়া হয়েছে বলে মনে হয় ফলনও ঠিক সে রকম আসে। আর সেক্ষেত্রে শুন্যস্থানে কী হবে বলে যৌক্তিক মনে হয় সেটাও ঠিক করো। অন্যদিকে যা লেখা হয়েছে সেটা ঠিকই আছে ধরে নিয়ে কেবল ছাড়নটা (indentation) ঠিক করো, আর তাতে শুন্যস্থানে কী বসানো যথার্থ হবে তাও নির্ণয় করো।

```
if (n < 10)
  if (n > 0)
    cout << "dhonatok." << endl;
  else
    cout << "_________." << endl;
```

প্রদত্ত ক্রমলেখটি (program) লেখার সময় এমন ভাবে ছাড়ন (indentation) দেয়া হয়েছে যে মনে হচ্ছে else অংশটুকু প্রথম if এর শর্ত n < 10 মিথ্যা হলে কার্যকর হবে। কিন্তু সিপিপি ভাষায় ছাড়ন বা ফাঁকা দেয়া না দেয়া গণনির (computer) জন্য কোন ব্যাপার নয়। আর ঝুলন্ত নাহলের (dangling else) আলোচনা থেকে আমরা জানি এই else টি তার নিকটতম পূর্ববর্তী এমন একটি if এর সাথে সংশ্লিষ্ট যে if এর সাথে আর কোন else জুড়ে দেয়া হয় নি। কাজেই, সেই হিসাবে ঠিক উপরে যেমনটি দেখানো হলো, সেভাবে এই else টি দ্বিতীয় if এর শর্ত n > 0 মিথ্যা হলে কার্যকর হবে।

```
dhonatok.          কোন ফলন নাই          _________.
```

এমতাবস্থায় এই ক্রমলেখ চালালে আমরা ৭, ১৫, বা -৩ যোগান (input) দিয়ে যে ফলন (output) পাবো তা উপরের তিনটি স্তম্ভে দেখানো হয়েছে। লক্ষ্য করো ১৫ যোগান দিলে আমরা কোন ফলন আসলে পাবো না, কারণ n < 10 শর্তের কোন else নেই। আর শুন্য-স্থানটি _________ ফলনে আসে যখন সংখ্যাটি শুন্যের সমান বা কম হয় অর্থাৎ অধনাত্নক হয়। আমরা তাহলে _________ এর স্থানে লিখতে পারি odhonatok।

```
if (n < 10)
{
  if (n > 0)
    cout << "dhonatok." << endl;
}
else
  cout << "_________." << endl;
```

ছাড়ন (indentation) দেয়া দেখে যেমন মনে হয়, ক্রমলেখটি (program) সেই অনুযায়ী সংশোধন করলে ঠিক উপরের মতো করে বাঁকা বন্ধনী ব্যবহার করতে হবে। সেক্ষেত্রে শুন্যস্থান অংশটি ফলনে আসবে যখন n সংখ্যাটি ১০ এর বড় বা সমান, যেমন ধরো ১৫। এক্ষেত্রে আমরা তাই _____ এর বদলে লিখতে পারি "10 ba boro"। এবার খেয়াল করো n এর মান যখন শুন্য এর বেশী কিন্তু ১০ এর কম যেমন ৭, তখন কিন্তু আমরা ফলন পাবো dhonatok, আর শুন্য বা কম হলে কোন ফলনই পাবো না।

৯. তিনটি সংখ্যা যোগান (input) নিয়ে কোনটি বড়, কোনটি ছোট ফলনে (output) দেখাও।

ফিরিস্তি ৭.৮: তিনটি সংখ্যার বড়-ছোট (Small and Big of Three Numbers)

```
int a, b, c;            // চাইলে ভগ্নকও নিতে পারো
cout << "sokhya tinti koto? ";
cin >> a >> b >> c;     // যোগান নাও

int boro, soto;         // চলক ঘোষণা
if (a > b)              // a যদি বড় হয় b এর চেয়ে
  boro = a, soto = b;
else                    // b যদি বড় হয় a এর চেয়ে
  boro = b, soto = a;
if (boro < c)           // c যদি boro এর চেয়ে বড় হয়
  boro =  c;
else if (soto > c)      // c যদি soto এর চেয়ে ছোট হয়
  soto = c;

cout << "boro " << boro << " ";
cout << "soto " << soto << endl;
```

উপরের ক্রমলেখ খেয়াল করো। প্রথম দুটি সংখ্যা a ও b কে তুলনা করে বড় ও ছোট নির্ধারণ করা হয়ছে। তারপর c কে তুলনা করা হয়েছে সেটা আরো বড় কিনা দেখতে, যদি তা না হয় তাহলে সেটা আরো ছোট কিনা সেটা পরীক্ষা করা হয়েছে। লক্ষ্য করো c কে তুলনা করা সময় একটা else লাগানো হয়েছে, কারণ boroর বড় হলে তো আর sotoর ছোট কিনা পরীক্ষা করার দরকার নেই। তুমি কিন্তু চাইলে নীচের মতো করেও ক্রমলেখ লিখতে পারো। প্রথমে ধরে নাও তোমার সংখ্যা একটাই কাজেই aই বড়, আবার aই ছোট। এরপর তাদের সাথে b কে তুলনা করো। আর শেষে তাদের সাথে c কে তুলনা করো।

```
int boro = a, soto = a; // ধরে নেই a–ই বড় ও ছোট
if (boro < b)           // b যদি তার চেয়েও বড় হয়
  boro =  b;
else if (soto > b)      // b যদি তার চেয়েও ছোট হয়
  soto = b;
if (boro < c)           // c যদি তার চেয়েও বড় হয়
  boro =  c;
else if (soto > c)      // c যদি তার চেয়েও ছোট হয়
  soto = c;
```

১০. তিনটি সংখ্যা যোগান (input) নিয়ে তাদের মধ্যে মাঝেরটি ফলনে (output) দেখাও।

ফিরিস্তি ৭.৯: তিনটি সংখ্যার মধ্যক (Median of Three Numbers)

```
// ধরো চলক তিনটি a, b, c ঘোষণা করে যোগান নেয়া হয়েছে

if (a > b)                  //  ক্রম হলো a > b
  if (c > a)                //  ক্রম হলো  c > a > b
    cout << a << endl;
  else if (b > c)           //  ক্রম হলো a > b > c
    cout << b << endl;
  else                      // ক্রম হলো a >= c >= b
    cout << c << endl;
else                        // ক্রম হলো a <= b
  if (c < a)                // ক্রম হলো c < a <= b
    cout << a << endl;
  else if c > b)            // ক্রম হলো a <= b < c
    cout << b << endl;
  else                      // ক্রম হলো a <= c <= b
    cout << c << endl;
```

উপরের ক্রমলেখতে প্রথমে a ও b তুলনা করা হয়েছে। তারপর c তাদের বড়টির চেয়ে বড় কিনা, নাহলে ছোটটির চেয়ে ছোট কিনা পরীক্ষা করা হয়েছে, আর তাও না হলে সেটি উভয়ের মাঝামাঝি। এভাবে তিনটি সংখ্যার ক্রম জানা হয়ে গেলে মাঝেরটি ফলনে (output) দেখানো হয়েছে। এটি অন্তান্তি (nesting) ও মইয়ের (ladder) চমৎকার উদাহরণ।

১১. তিনটি সংখ্যা যোগান (input) নিয়ে তাদেরকে উর্ধক্রমে সাজিয়ে ফলন (output) দাও।

ফিরিস্তি ৭.১০: তিনটি সংখ্যার উর্ধক্রম (Three Numbers in Ascending Order)

```
// ধরো চলক তিনটি a, b, c ঘোষণা করে যোগান নেয়া হয়েছে

if (a > b)                  //  ক্রম হলো a > b
  if (c > a)                //  ক্রম হলো  c > a > b
    cout << b << " " << a << " " << c << endl;
  else if (b > c)           //  ক্রম হলো a > b > c
    cout << c << " " << b << " " << a << endl;
  else                      // ক্রম হলো a >= c >= b
    cout << b << " " << c << " " << a << endl;
else                        // ক্রম হলো a <= b
  if (c < a)                // ক্রম হলো c < a <= b
    cout << c << " " << a << " " << b << endl;
  else if c > b)            // ক্রম হলো a <= b < c
    cout << a << " " << b << " " << c << endl;
  else                      // ক্রম হলো a <= c <= b
    cout << a << " " << c << " " << b << endl;
```

উপরের ক্রমলেখতে প্রথমে a ও b তুলনা করা হয়েছে। তারপর c তাদের বড়টির চেয়ে বড় কিনা, নাহলে ছোটটির চেয়ে ছোট কিনা পরীক্ষা করা হয়েছে, আর তাও না হলে সেটি উভয়ের মাঝামাঝি। এভাবে তিনটি সংখ্যার ক্রম জানা হয়ে গেলে তাদের মানের উর্ধক্রমে (ascending order) ফলনে (output) দেখানো হয়েছে। এটি যদি নাহলের (if else) অন্তান্তি (nesting) ও মইয়ের (ladder) এক সাথে ব্যবহারের চমৎকার উদাহরণ।

১২. গণিতে প্রাপ্ত নম্বর যোগান (input) নিয়ে সেটা থেকে বর্ণ মান (letter grade) ফলন দাও। ধরো ৯০ বা বেশী হলে A, ৮০ বা বেশী হলে B, ৭০ বা বেশী হলে C, ৬০ বা বেশী হলে D, ৫০ বা বেশী হলে E, আর তারও কম হলে F বর্ণ মান পাওয়া যায়।

ফিরিস্তি ৭.১১: নম্বর হতে বর্ণমান (Letter Grades from Numbers)

```
cout << "gonite nombor koto? ";
int nombor; cin >> nombor;

if (nombor >= 90)
  cout << "bornoman A" << endl;
else if (nombor >= 80)
  cout << "bornoman B" << endl;
else if (nombor >= 70)
  cout << "bornoman C" << endl;
else if (nombor >= 60)
  cout << "bornoman D" << endl;
else if (nombor >= 50)
  cout << "bornoman E" << endl;
else // ৫০ এর ছোট
  cout << "bornoman F" << endl;
```

উপরের ক্রমলেখ পল্টি ব্যাপার (switch cases) দিয়ে করা সম্ভব নয়, কারন এখানে >= তুলনা ব্যবহার করতে হবে। পল্টি ব্যাপার কেবল সমান == তুলনায় ব্যবহার করা যায়।

১৩. একটি দ্বিমাত্রিক (two dimensional) বিন্দুর স্থানাঙ্ক দেওয়া আছে, বিন্দুটি চারটি চতুর্ভাগের (quadrant) ঠিক কোনটিতে পড়বে নির্ণয় করো।

এই ক্রমলেখতে (program) আমরা কেবল চতুর্ভাগ (quadrant) বিবেচনা না করে বরং, বিন্দুটি কোন অক্ষের ওপরে কিনা, হলে ধনাত্নক দিকে না ঋণাত্নক দিকে, অথবা স্থানাংকরে মূল বিন্দুতে কিনা তাও বিবেচনা করবো। যে কোন একটি প্রদত্ত বিন্দুর x বা y দুটোই আলাদা আলাদা ভাবে ধনাত্নক বা ঋণাত্নক বা শুন্য এই তিন রকম হতে পারে। কাজেই একসাথে বিবেচনা করলে আমরা মোট নয় রকম সমাবেশ (combination) পাবো।

ফিরিস্তি ৭.১২: বিন্দুর চতুর্ভাগ নির্ণয় (Quadrant of a Point)

```
float x, y;
cout << "bhuj x? ";
cin >> x;
cout << "koti y? ";
cin >> y;
```

```
if (x > 0)
  if (y > 0)
    cout << "prothom choturvag" << endl;
  else if (y < 0)
    cout << "choturtho choturvag" << endl;
  else    // y শুন্য
    cout << "dhonatok x okkher opor" << endl;
else if (x < 0)
  if (y > 0)
    cout << "ditiyo choturvag" << endl;
  else if (y < 0)
    cout << "tritiyo choturvag" << endl;
  else
    cout << "rinatok x okkher opor" << endl;
else    // x শুন্য
  if (y > 0)
    cout << "dhonatok y okkher opor" << endl;
  else if (y < 0)
    cout << "rinatok y okkher opor" << endl;
  else    // y শুন্য
    cout << "sthananker mul bindu" << endl;
```

১৪. একটি প্রগমণ ১, ২, ৩, ..., ৯, ১১, ২২, ৩৩, ..., ৯৯ এর ১ম পদ ১, আর ১৮ তম পদ ৯৯। কততম পদ দেখাতে হবে তা যোগান (input) নিয়ে পদটি ফলনে (output) দেখাও।

```
cout << "kototom pod: " << endl;
int n; cin << n;

if (n < 0)
  cout << "pallar baire" << endl;
else if (n <= 9)  // এক অঙ্কের সংখ্যা
  cout << n << endl;
else if (n <= 18) // দুই অঙ্কের সংখ্যা
  cout << ((n-9) * 11) << endl;
else
  cout << "pallar baire" << endl;
```

১৫. তোমাকে -১০০ ও ১০০ এর মধ্যে দুটি সংখ্যা যোগান (input) হিসাবে দেওয়া হবে, তুমি ওই দুটি সংখ্যা সহ তাদের মাঝের সকল সংখ্যার যোগফল ফলনে (output) দেখাও।

নীচের সংশ্লিষ্ট ক্রমলেখ (program) দেখানো হলো। যে সংখ্যা দুটি যোগান (input) নেয়া হবে, সেগুলো অবশ্যই -১০০ ও ১০০ এর ভিতরে হতে হবে। আমরা তাই আগে পরীক্ষা করে দেখবো। যদি n1 বা n2 যে কোনটি -100 এর ছোট বা 100 এর বড় হয়, তাহলে

ত্রুটি বার্তা (error message) দেখিয়ে বিফল হয়ে নিয়ন্ত্রণ ফেরত যাবে। খেয়াল করো আমাদের কিন্তু শর্তগুলোকে অথবা || দিয়ে যুক্ত করতে হবে।

```
cout << "sonkhya duti koto? ";  // যাচনা
int n1, n2; cin >> n1 >> n2;    // যোগান

// −১০০ ও ১০০ এর মধ্যে কিনা পরীক্ষা করতে হবে
if (n1 < −100 || n1 > 100 ||
        n2 < −100 || n2 > 100)
{
  cout << "sonkhya pallar baire" << endl;
  return EXIT_FAILURE;
}

int s, n;   // (প্রথম পদ + শেষ পদ) আর পদসংখ্যা

s = n1 + n2;  // প্রথম পদ + শেষ পদ।

if (n1 > n2)     // কোনটা ছোট কোনটা বড়
  n = n1 − n2 + 1; // পদসংখ্যা
else
  n = n2 − n1 + 1;  // পদসংখ্যা

cout << "jogfol " << s*n/2; // ফলন
```

এবার আমরা জানি কোন সমান্তর প্রগমণের সংখ্যাগুলোর যোগফল হলো (প্রথম সংখ্যা + শেষ সংখ্যা) * পদসংখ্যা / 2। সংখ্যা দুটো যোগান নেয়ার সময় ব্যবহারকারী যে কোনটিকে আগে যোগান দিতে পারে, মানে কোনটা বড় কোনটা ছোট আমরা নিশ্চিত থাকবো না। (প্রথম সংখ্যা + শেষ সংখ্যা) এই যোগফল s বের করতে এতে কোন সমস্যা হবে না, তবে পদসংখ্যা n বের করতে গেলে আমাদের জানতে হবে কোনটা বড় কোনটা ছোট। ধরো ৭ আর ১৩ নিজেদের সহ তাদের মধ্যে কয়টা সংখ্যা আছে সেটা বের করা যায় ১৩ - ৭ + ১ হিসাব করে, যেখানে ১৩ হলো বড় আর ৭ হলো ছোট। তো n1 আর n2 এর নিজেদের সহ তাদের মাঝে মোট কয়টি সংখ্যা আছে তা বের করতে আমাদের জানতে হবে কোনটি বড়। তো আমরা একটি যদি নাহলে (if else) ব্যবহার করে দেখবো n1 > n2 কিনা, যদি হয় তাহলে পদসংখ্যা n1 − n2 + 1 আর যদি না হয় তাহলে পদসংখ্যা হবে n2 − n1 + 1। সবশেষে যোগফল হলো s * n / 2 আমরা যেটা ফলনে (output) দেখাবো।

১৬. একটি প্রদত্ত বর্ষ অধিবর্ষ কি না তা নির্ণয়ের ক্রমলেখটি (program) তুমি যদি-নাহলে মই (if else ladder) ব্যবহার করে লিখবে। তবে ক্রমলেখটি রচনা করার সময় তোমাকে মনে রাখতে হবে যে এটি ১ থেকে ২০০০ সাল পর্যন্ত প্রতিটি সালের জন্য চালানো হবে। কাজেই তুমি মইয়ের শর্তগুলো এমন ভাবে সাজাবে যাতে ক্রমলেখ দ্রুততম হয়।

যদি ১ থেকে ২০০০ সাল পর্যন্ত প্রতিটি সালের জন্য চালানো হয় তাহলে আমরা প্রথমে প্রত্যেক রকমের সালের হিসাব করি। মোটামুটি প্রতি চারটি সালের তিনটি অধিবর্ষ নয়, একটি অধিবর্ষ। কাজেই সবচেয়ে বেশী সংখ্যক ২০০০ / ৪ * ৩ = ১৫০০ টি সাল আছে যে

গুলো ৪ দিয়ে বিভাজ্য হয় না, এগুলোর কোনটিই অধিবর্ষ নয়। বাঁকী ৫০০ টি সাল ৪ দিয়ে বিভাজ্য। এদের মধ্যে যেগুলো ১০০ দিয়ে বিভাজ্য নয় যেমন ১৯৯৬ এমন ২০টি ছাড়া বাঁকী ৪৮০ টি অধিবর্ষ। আর ওই ২০টি সালের মধ্যে যে ১৫টি ৪০০ দিয়ে বিভাজ্য নয় সেগুলো অধিবর্ষ নয়, আর বাঁকী ৪টি সাল যে গুলো ৪০০ দিয়ে বিভাজ্য সেগুলো অধিবর্ষ।

```
if (bosor % 4 != 0)          // ৪ দিয়ে বিভাজ্য নয়
  cout << "odhiborsho noy" << endl;
else if (bosor % 100 != 0)   // ১০০ দিয়ে বিভাজ্য নয়
  cout << "odhiborsho hoy" << endl;
else if (bosor % 400 != 0)   // ৪০০ দিয়ে বিভাজ্য নয়
  cout << "odhiborsho noy" << endl;
else   //  if (bosor % 400 = 0) ৪০০ দিয়ে বিভাজ্য
  cout << "odhiborsho hoy" << endl;
```

দ্রুততম গতির ক্রমলেখয়ের (program) জন্য যে রকমের সাল সবচেয়ে বেশী সেগুলো নির্ণয় করতে সবচেয়ে কম সংখ্যক শর্ত পরীক্ষণ ব্যবহার করতে হবে। কাজেই আমরা ১৫০০ সাল যেগুলো ৪ দিয়ে বিভাজ্য নয় সেগুলোকে প্রথম শর্ত পরীক্ষা করেই বের করতে চাইবো। উপরের ক্রমলেখ খেয়াল করো, আমরা তাই করেছি। এরপরে রয়েছে যে ৪৮০টি বছর যেগুলো ৪ দিয়ে বিভাজ্য কিন্তু ১০০ দিয়ে বিভাজ্য নয়। আমরা এগুলোকে দুইবার শর্ত পরীক্ষা করে বের করতে চাই। একটা শর্ত হচ্ছে ৪ দিয়ে বিভাজ্য নাহওয়া কাজেই প্রথম শর্তের else হিসাবে থাকবে সেটা, আরেকটি শর্ত হলো ১০০ দিয়ে বিভাজ্য না হওয়া। উপরের ক্রমলেখয়ের যদি নাহলে মইতে (if else ladder) দেখো else if দিয়ে এটা করা হয়েছে। এরপর থাকে ১৫ টি সাল যেগুলো ১০০ দিয়ে বিভাজ্য কিন্তু ৪০০ দিয়ে বিভাজ্য নয় এই ১৫ টি সাল, এগুলো নির্ণয় করা হয়েছে আরেকটি else if লাগিয়ে অর্থাৎ মোট তিনটি শর্ত পরীক্ষণ শেষে। আর সবশেষে ৪০০ দিয়ে বিভাজ্য সেই সালগুলো এসেছে সবশেষের else দিয়ে, এগুলোর জন্য তিনটি শর্ত পরীক্ষণই লেগেছে, কারণ শেষের শর্ত মিথ্যা হলেই তো এগুলো নির্ণীত হবে। তাহলে মোট শর্ত পরীক্ষা লাগলো কতগুলো? ১৫০০ * ১ + ৪৮০ * ২ + ১৫ * ৩ + ৫ * ৩ = ২৫২০ টি। তুমি আরো নানান ভাবে চেষ্টা করে দেখতে পারো, এর চেয়ে কম শর্ত পরীক্ষা করে করতে পারো কিনা! পারবে না!

```
if (bosor % 4 != 0 ||
      (bosor % 100 == 0 && bosor % 400 != 0))
  cout << "odhiborsho noy" << endl;
else
  cout << "odhiborsho hoy" << endl;
```

একই ক্রমলেখ আমরা যদি নাহলে মই (if else ladder) ব্যবহার না করে ঠিক উপরের ক্রমলেখয়ের মতো বুলক সংযোজক (Boolean connective) ব্যবহার করে করতে পারি। বুলক সংযোজকের আংশিক মূল্যায়ন (partial evaluation) মনে আছে? অথবা || ক্ষেত্রে যে কোন একটি উপাদান (operand) সত্যি হলেই অন্যটি মূল্যায়ন ছাড়াই আমরা ফলাফল সত্য বলে ধরে নিতে পারি। আর && এর ক্ষেত্রে যে কোন একটি উপাদান (operand) মিথ্যা হলেই ফলাফল মিথ্যা বলে ধরে নেয়া যায়। কোন সাল অধিবর্ষ নয় যখন সালটি ৪ দ্বারা বিভাজ্য নয় অথবা ১০০ দ্বারা বিভাজ্য হলেও ৪০০ দ্বারা বিভাজ্য নয় তখন। যদির সাথে শর্ত হিসাবে সেটিই লাগানো হয়েছে দেখো। অন্যদিকে কোন সাল অধিবর্ষ হতে গেলে || এর ফলাফল মিথ্যা হতে হবে, তারমানে বাম ও ডানের উভয় উপাদান (operand)

মিথ্যা হতে হবে অর্থাৎ bosor % 4 == 0 এবং (bosor % 100 == 0 && bosor % 400 != 0) মিথ্যা হতে হবে। এখানে bosor % 4 == 0 মিথ্যা হওয়া মানে বছরটি ৪ দ্বারা বিভাজ্য হওয়া আর (bosor % 100 == 0 && bosor % 400 != 0) মিথ্যা হতে গেলে && এর দুপাশের যেকোন একটি মিথ্যা হলেই হবে। তো (bosor % 100 == 0 মিথ্যা হওয়া মানে বছরটি ১০০ দ্বারা বিভাজ্য না হওয়া আর bosor % 400 != 0 মিথ্যা হওয়া মানে ৪০০ দিয়ে বিভাজ্য হওয়া।

১৭. বাংলা বছরের কততম মাস তা যোগান (input) নিয়ে সেই মাসের নাম ও ওই মাসে কত দিন তা ফলনে (output) দেখাও। একাজে পল্টি ব্যাপার (switch cases) ব্যবহার করো।

ফিরিস্তি ৭.১৩: বাংলা মাসের নাম (Bengali Month Names)

```
int mash; cin >> mash;  // চাইলে যাচনা করতে পারো

switch(mash)
{
  case 1: cout << "boishakh 31" << endl; break;
  case 2: cout << "joistho 31" << endl; break;
  case 3: cout << "ashar 31" << endl; break;
  case 4: cout << "shrabon 31" << endl; break;
  case 5: cout << "vadro 31" << endl; break;
  case 6: cout << "arshin 30" << endl; break;
  case 7: cout << "kartik 30" << endl; break;
  case 8: cout << "ogrohayon 30" << endl; break;
  case 9: cout << "poush 30" << endl; break;
  case 10: cout << "magh 30" << endl; break;
  case 11: cout << "falgun 30" << endl; break;
  case 12: cout << "choitro 30" << endl; break;
  default: cout << "ojana mash" << endl; break;
}
```

১৮. কতটা বাজে সেই সময় ঘন্টায় যোগান (input) নিয়ে মাঝরাত (০-২), প্রভাত (৩-৬), সকাল (৭-১১), দুপুর (১২-১৪), বিকাল (১৫-১৭), সন্ধ্যা (১৮-১৯), রাত (২০-২৪) ফলনে দেখাও। একাজে পল্টি ব্যাপার (switch cases) ব্যবহার করো।

```
int somoy; cin >> somoy;  // যাচনা করতে পারো

switch(somoy)
{
  case 0: case 1: case 2:
    cout << "majhrat" << endl; break;
  case 3: case 4: case 5: case 6:
    cout << "provat" << endl; break;
  case 7: case 8: case 9: case 10: case 11:
    cout << "shokal" << endl; break;
```

```
    case 12: case 13: case 14:
      cout << "dupur" << endl; break;
    case 15: case 16: case 17:
      cout << "bikal" << endl; break;
    case 18: case 19:
      cout << "shondhya" << endl; break;
    case 20: case 21: case 22: case 23
      cout << "rat" << endl; break;
    default:
      cout << "ojana somoy" << endl;
  }
```

১৯. এমন একটি ক্রমলেখ (program) লিখো যেটি ১-৫ পর্যন্ত ক্রম অনুযায়ী পাঁচটা কোমল পানীয়ের (পানি, কোক, স্প্রাইট, ফানটা, পেপসি) নামের তালিকা দেখাবে, তারপর ক্রমিক নম্বর যোগান (input) নিয়ে কোমল পানীয়টির নাম ফলনে (output) দেখাবে। আর ক্রমিক নম্বরটি যদি ১-৫ এর বাইরে হয়, তাহলে সে সংক্রান্ত একটি ত্রুটি বার্তা (error message) দেখাবে। তুমি এই ক্রমলেখটি একবার পল্টি ব্যাপার (switch cases) ব্যবহার করে আবার যদি নাহলে (if else) ব্যবহার করে করো।

```
  cout << "talika" << endl;
  cout << "1 pani" << endl;
  cout << "2 coke" << endl;
  cout << "3 sprite" << endl;
  cout << "4 fanta" << endl;
  cout << "5 pepsi" << endl;
  cout << endl;

  cout << "posondo: " << endl;
  int posondo;
  cin >> posondo;

  cout << "posondo ";
  switch(posondo)
  {
    case 1: cout << "pani" << endl; break;
    case 2: cout << "coke" << endl; break;
    case 3: cout << "sprite" << endl; break;
    case 4: cout << "fanta" << endl; break;
    case 5: cout << "pepsi" << endl; break;
    default: cout << "ojana" << endl; break;
  }
```

উপরের ক্রমলেখয়ের (program) পল্টি ব্যাপার (switch case) অংশটি যদি নাহলে (if else) ব্যবহার করে লিখলে নীচের মতো হবে।

```
if (posondo == 1)
  cout << "pani" << endl;
else if (posondo == 2)
  cout << "coke" << endl;
else if (posondo == 3)
  cout << "sprite" << endl;
else if (posondo == 4)
  cout << "fanta" << endl;
else if (posondo == 5)
  cout << "pepsi" << endl;
else
  cout << "ojana" << endl;
```

২০. একটি সংখ্যার পুরক সংখ্যা নির্ণয় করো। সংখ্যাট এক অঙ্কের হলে তার পুরক সংখ্য ৯ এর সাথে বিয়োগফল, দুই অঙ্কের হলে ৯৯ এর সাথে বিয়োগফল, তিন অঙ্কের হলে ৯৯৯ এর সাথে বিয়োগফল। তিনের চেয়ে বেশী অঙ্কের সংখ্যা যোগান (input) দেওয়া হবে না।

```
int nombor, purok;
cin >> nombor;

// ত্রুটি আগেই সামলানো হলো
if (nombor < 0 || nombor > 1000)
{
  cout << "onakankhito" << endl;
  return EXIT_FAILURE;
}

// এবার কেবল বৈধ ব্যাপারগুলো
if (nombor <= 9)          // এক অঙ্ক মানে ৯ বা কম
  purok = 9 - nombor;
else if (nombor <= 99)    // এক অঙ্ক মানে ৯৯ বা কম
  purok = 99 - nombor;
else if (nombor <= 999) // এক অঙ্ক মানে ৯৯৯ বা কম
  purok = 999 - nombor;
```

২১. এমন একটি ক্রমলেখ (program) লিখো যেটা ৫ জন লোক যাদের ক্রমিক ১-৫ তাদের কে কতটা করে পরোটা খেয়েছে যোগান (input) নিবে। ক্রমলেখটি তারপর একজনে সর্বোচ্চ কয়টা পরোটা খেয়েছে সেটা ফলনে (output) দেখাবে। আর কোন লোক সর্বোচ্চ সংখ্যক পরোটা খেয়েছে ক্রমলেখটি সেটাও দেখাবে, তবে সর্বোচ্চ পরোটা খাওয়া একাধিক ব্যক্তি থাকলে প্রথমজনের ক্রমিক নম্বর হলেই চলবে, পরের জনদের দরকার নাই।

আমরা পাঁচজন লোকের জন্য সুবিধার্থে পাঁচটি চলক নিবো p1, p2, p3, p4, p5। তারপর যথাযথ ভাবে যোগান যাচনা (input prompt) করে কোন লোক কতটি পরোটা খেয়েছে সেটা যোগান (input) নিবো। তারপর আমাদের আরো দুটি চলক লাগবে: একটি

হলো sorboccho সর্বোচ্চ কতটি পরোটা খেয়েছে আর একটি হলো kekheyese কে খেয়েছে সর্বোচ্চটি। তারপর আমরা একজন একজন করে লোক বিবেচনা করবো। শুরুতে মাত্র একজন লোক ধরে নিলে সেই সর্বোচ্চ পরোটা খেয়েছে, কাজেই sorboccho = p1, kekheyese = 1 আদিমান হিসাবে আরপণ করা হয়েছে। এর পরের প্রতিটি ব্যক্তির জন্য আমরা পরীক্ষা করে দেখবো সে এ পর্যন্ত sorboccho এর মান যত তার চেয়ে বেশী পরোটা খেয়েছে কিনা। যদি খেয়ে থাকে তাহলে sorboccho এর মান বদলে যাবে আর কে খেয়েছে সেটাও বদলে যাবে। এরকম ক্রমলেখ (program) নীচে দেখো।

ফিরিস্তি ৭.১৪: পাঁচটি সংখ্যার বৃহত্তম (Largest of Five Numbers)

```
int p1; cout << "p1: "; cin >> p1;
int p2; cout << "p2: "; cin >> p2;
int p3; cout << "p3: "; cin >> p3;
int p4; cout << "p4: "; cin >> p4;
int p5; cout << "p5: "; cin >> p5;

int sorboccho = p1, kekheyese = 1;
if (sorboccho < p2)
  { sorboccho = p2; kekheyese = 2; }
if (sorboccho < p3)
  { sorboccho = p3; kekheyese = 3; }
if (sorboccho < p4)
  { sorboccho = p4; kekheyese = 4; }
if (sorboccho < p5)
  { sorboccho = p5; kekheyese = 5; }

cout << "porota " << sorboccho << endl;
cout << "lokta " << kekheyese << endl;
```

খেয়াল করো আমরা > ব্যবহার করেছি >= ব্যবহার করি নাই। এর কারণ এ পর্যন্ত সর্বোচ্চ যতটি খাওয়া হয়েছে তার সমান কেউ যদি পরের কেউ খেয়েও থাকে, আমরা কিন্তু সেই লোকটিকে ফলনে (output) দেখাতে চাইনা, বরং আগের জনকেই দেখাতে চাই। তুমি যদি সর্বোচ্চ পরোটা খেয়েছে এরকম কয়েক জন থাকলে তাদের মধ্যের শেষের জনকে ফলনে (output) দেখাতে চাও, তাহলে < বদলে <= করে দিবে। আর একটি ব্যাপার হলো অনেক সময় sorboccho আর kekheyese চলক দুটির আদিমান ১মজনের পরোটা খাওয়া বিবেচনা করে না দিয়ে নীচের মতো করে বরং আমরা একটা ছোট সংখ্যা ধরে নেই, তারপর ২য়, ৩য়, ৪র্থ, ৫ম লোকের মতো ১ম জনের জন্যও একই রকম যদি নাহলে ব্যবহার করি। এতে সব লোকের জন্য চিন্তা করাটা একই রকম হয়।

```
// সর্বোচ্চ একটা ছোট মান, কে খেয়েছে সেটা জানিনা
int sorboccho = 0, kekheyese = 0;

if (sorboccho < p1)   // ১ম জনের ক্ষেত্রেও একই
  { sorboccho = p1; kekheyese = 1; }
// p2, p3, p4, p5 এর যদি নাহলে ঠিকই থাকবে
```

২২. একজন লোক স্বাভাবিক নিয়ম অনুযায়ী সপ্তাহে ৪০ ঘন্টা কাজ করে, ৪০ ঘন্টার বেশী কাজ করলে অতিরিক্ত সময়টুকুর জন্য স্বাভাবিক নিয়মের চেয়ে ১.৫ গুণ মজুরি পায়। কোন এক সপ্তাহে লোকটি কত ঘন্টা কাজ করেছে আর স্বাভাবিক নিয়মে ঘন্টা প্রতি মজুরি কত তা যোগান (input) নিয়ে ওই সপ্তাহে তার মোট মজুরি কত তা ফলনে (output) দেখাও।

ফিরিস্তি ৭.১৫: সপ্তাহের মজুরি হিসাব (Weekly Wage Calculation)

```
float const shaShima = 40.0;   // স্বাভাবিক সীমা
float const otiHar = 1.5;      // অতিরিক্ত হার

float ghontaProti;  // ঘন্টাপ্রতি কত হার
float kotoGhonta;   // কত ঘন্টা কাজ
float motMojuri;    // মোট মজুরি কত

cout << "ghonta proti har: ";
cin >> ghontaProti;
cout << "koto ghonta kaj: ";
cin >> kotoGhonta;

if (kotoGhonta <= shaShima)
  motMojuri = kotoGhonta * ghontaProti;
else  // অতিরিক্ত সময় কাজ হয়েছে
{
  // স্বাভাবিক মজুরি ৪০ ঘন্টার
  float shaMojuri = shaShima * ghontaProti;

  // অতিরিক্ত ঘন্টা বের করতে হবে
  float otiGhonta = kotoGhonta - shaShima;

  // অতিরিক্ত সময়ের মজুরির হার
  float otiGhontaProti = ghontaProti * otiHar;

  // অতিরিক্ত সময়ে মজুরি
  float otiMojuri = otiGhonta * otiGhontaProti;

  // মোট মজুরি দুটোর যোগফল
  motMojuri = shaMojuri + otiMojuri;
}

cout << "mot mojuri " << motMojuri << endl;
```

২৩. ধরো তুমি চার টুকরো কাগজ নিয়েছো। তোমার ১ম টুকরোতে লেখা আছে ১, ৩, ৫, ৭, ৯, ১১, ১৩, ২য় টুকরোতে আছে ২, ৩, ৬, ৭, ১০, ১১, ১৪, ১৫, ৩য় টুকরোতে আছে ৪, ৫, ৬, ৭, ১২, ১৩, ১৪, ১৫, ৪র্থ টুকরোতে আছে ৮, ৯, ১০, ১১, ১২, ১৩, ১৪, ১৫। তোমার

ক্রমলেখ (program) ব্যবহারকারী মনে মনে একটি সংখ্যা ধরবে, আর সেটি ১ম, ২য়, ৩য়, ৪র্থ টুকরোর কোন কোনটিতে আছে যোগান (input) দিবে, তারপর তোমার ক্রমলেখ ব্যবহারকারী মনে মনে যে সংখ্যাটি ধরেছে সেটি ফলনে (output) দেখাবে। এটি খুব সহজ একটি ব্যাপার। যে যে টুকরোতে সংখ্যাটি আছে ওই টুকরোগুলোর প্রথম সংখ্যাগুলো যোগ করলেই ব্যবহারকারীর সংখ্যাটি পাওয়া যাবে। যেমন ব্যবহারকারীর সংখ্যাটি যদি ১, ৩, ৪ নম্বর টুকরোতে থাকে তাহলে সংখ্যাটি ১ + ৪ + ৮ = ১৩।

সংখ্যা বলার খেলা (Number Finding Game)

```
cout << "ekta sonkhya nao 0 theke 15" << endl;

cout << "tash 1: 1 3 5 7 9 11 13 15" << endl;
cout << "tash 2: 2 3 6 7 10 11 14 15" << endl;
cout << "tash 3: 4 5 6 7 12 13 14 15" << endl;
cout << "tash 4: 8 9 10 11 12 13 14 15" << endl;

cout << "nicher prosno gulor uttor dao" << endl;
cout << "uttor ha hole 1, na hole 0" << endl;

int tash1, tash2, tash3, tash4;

cout << "tash 1 e tomar sonkhya ase? ";
cin >> tash1;
cout << "tash 2 e tomar sonkhya ase? ";
cin >> tash2;
cout << "tash 3 e tomar sonkhya ase? ";
cin >> tash3;
cout << "tash 4 e tomar sonkhya ase? ";
cin >> tash4;

int sonkhya = 0;

if (tash1) sonkhya += 1;
if (tash2) sonkhya += 2;
if (tash3) sonkhya += 3;
if (tash4) sonkhya += 4;

cout << "tomar sonkhya " << sonkhya << endl;
```

উপরের ক্রমলেখ (program) দেখো। প্রথমে কাগজের টুকরো বা তাসগুলোতে কী কী সংখ্যা লেখা আছে তা দেখানো হয়েছে। এরপর বলা হয়েছে পরের দেখানোর প্রশ্নগুলোর উত্তর হ্যা হলে ১ আর না হলে ০ দিয়ে দিতে। আমরা চারটি চলক নিয়েছি। আর উত্তরগুলো ওই চলকগুলোতে আছে। প্রশ্নে যেমন বলা হয়েছে যে তাসগুলোতে ব্যবহারকারীর মনে মনে ধরে নেয়া সংখ্যাটি আছে সেই তাসগুলোর প্রথম সংখ্যাগুলো নিয়ে আমাদের যোগ করতে হবে। আমরা শুরুতে সংখ্যাটি ধরে নিয়েছি শুন্য `int sonkhya = 0;` লিখে। এরপর দেখো

প্রতিটি if পরীক্ষা করছে সংখ্যাটি ওই তাসে আছে কিনা, অর্থাৎ ব্যবহারকারীর দেয়া উত্তর সত্য কিনা, সত্য হলে ওই তাসের প্রথম সংখ্যাটি তো আমরা জানিই, সেটা sonkhya চলকের সাথে যোগ করে দেয়া হয়েছে। পরিশেষে ফলন (output) ব্যবহারকারীর মনে মনে ধরে নেয়া সংখ্যাটি দেখানো হয়েছে।

৭.২৩ গণনা পরিভাষা (Computing Terminologies)

- শর্তালি (conditional)
- যদি (if)
- নাহলে (else)
- অন্বয়ী (relational)
- বুলক (Boolean)
- মই (ladder)
- অন্তান্তি (nested)
- ঝুলন্ত (dangling)
- শুন্য (empty)
- যৌগিক (compound)
- শনাক্তকরণ (detection)
- সংযোজক (connective)
- এবং, ও (and)
- অথবা, বা (or)
- নয়, না (not)
- সহযোজ্য (associative)
- সরল (simplification)
- অগ্রগণ্যতা (precedence)
- ক্রম (order)
- সমতুল (equivalence)
- বন্টন (distribution)
- বিনিময় (commutative)
- শোষণ (absorption)
- অসঙ্গতি (contradiction)
- নঞ মধ্যম (excluded middle)
- সত্যক সারণী (truth table)
- অনুকুলায়ন (optimisation)
- তিনিক (ternary)
- পল্টি (switch)
- ব্যাপার (case)
- নিয়ন্ত্রণ (control)
- ক্ষান্তি (break)
- অগত্যা (default)
- ব্যাপীয় (global)
- স্থানীয় (local)
- মহল্লা (block)
- উপমহল্লা (subblock)
- অধিমহল্লা (superblock)

অধ্যায় ৮

পুনালি পরিগণনা (Iterative Programming)

আমাদের জীবনটা এমন রোমাঞ্চকর নয় যে একদম প্রতিবারই তুমি অভিনব কিছু একটা করবে। খেয়াল করে দেখবে তোমাকে প্রায়শই একই রকম কাজ বার বার করতে হচ্ছে। তুমি হয়তো এতে একঘেঁয়েমি বোধ করছো, কিন্তু কিছুই করার নেই, নাক কান চোখ বন্ধ করে তোমাকে সেই একই কাজ বার বার করে যেতে হবে, যতক্ষণ না সেগুলো শেষ হচ্ছে। পুনালি পরিগণনায় (iterative programing) আমরা শিখবো কী করে বারবার একই কাজ করে যেতে হয়।

৮.১ জন্য ঘূর্ণীর পুনরাবৃত্তি (For Loop Repetition)

সিপিপি ভাষায় এমন একটি ক্রমলেখ (program) রচনা করো যেটি একটি ধণাত্নক পূর্ণক যোগান (input) নিয়ে ওই পূর্ণকের সমান সংখ্যক বার ফলনে (output) "cpp" দেখায়।

```
cout << "cpp cpp cpp cpp cpp " << endl;
```

তোমাকে যদি খুবই অল্প সংখ্যক বার যেমন মাত্র ৫ বার ফলনে (output) সিপিপি দেখাতে বলা হয়, ব্যাপারটা কিন্তু তোমার কাছে খুবই সহজ লাগবে। স্রেফ উপরের মতো করে একটা cout ব্যবহার করেই তুমি ৫ বার সিপিপি ফলনে দেখাতে পারবে। অথবা তুমি চাইলে কিন্তু নীচের মতো করে প্রত্যেকবার আলাদা আলাদা cout দিয়ে একটা করে cpp আলাদা আলাদা করে মোট ৫ বার ফলন দিতে পারো। আর তারপর একটা endl দেখালেই হলো।

```
cout << "cpp ";
cout << "cpp ";
cout << "cpp ";   // cpp এর পরে একটা ফাঁকা
cout << "cpp ";
cout << "cpp ";
cout << endl;
```

উপরের উপায়গুলোতে আমাদের দুটি সমস্যা রয়েছে। প্রথম সমস্যা হচ্ছে মাত্র ৫ বার না হয়ে যদি আমাদের ১০০০ বার বা এই রকম অনেক বেশী বার ফলন (output) দিতে বলা হয়, তাহলে তুমি কী করবে? তুমি কি ক্রমলেখয়ের (program) ভিতরে এইভাবে ১০০০ বার বা ওই রকম

অত বেশী বার আলাদা আলাদা করে cpp লিখবে? নিশ্চয় না! আর দ্বিতীয় সমস্যা হচ্ছে কত বার ফলনে cpp দেখাতে হবে সেটা যদি আমরা ব্যবহারকারীর কাছে থেকে যোগান (input) নিই, তাহলে সংখ্যাটা তো ক্রমলেখ রচনার সময় জানা থাকছে না, সুতরাং ওই ভাবে ঠিক কত বারই বা আমরা ক্রমলেখয়ের ভিতরে cpp লিখবো? ব্যাপারটার তো কোন সুরাহাই নেই!

```
for(int gunti = 1; gunti <= 1000; ++gunti)
  cout << "cpp ";
cout << endl;
```

আমরা আগে প্রথম সমস্যাটির সমাধান দেখি। যদি আমরা ক্রমলেখ (program) লেখার সময়ই জানি কতবার cpp দেখাতে হবে, আর সংখ্যাটি যদি বেশ বড় হয় তাহলে দেখো আমরা অতি সংক্ষেপে ঠিক উপরে দেখানো ক্রমলেখয়ের মতো কত সহজে ১০০০ বার সিপিপি দেখিয়ে ফেলতে পারি! এখন কথা হচ্ছে উপরের মাত্র এই তিনটি সারি কী ভাবে ১০০০ বার cpp লিখবে?

তোমাকে খাতা কলমে লিখতে দিলে তুমি নিজে কী করবে? তুমি হয়তো ১ থেকে ১০০০ পর্যন্ত এক এক করে ধারাবাহিকভাবে গুনতে শুরু করবে। আর যখনই একটা সংখ্যা উচ্চারণ করবে তখনই তুমি একবার cpp লিখবে। তুমি তো বুদ্ধিমান মানুষ তাই তুমি জানো যে তোমাকে ১০০০ এ গিয়ে থামতে হবে। কিন্তু গণনি (computer) যেহেতু বোকা তাই সে যেটা করে প্রতিবার ১ বাড়িয়ে পরের সংখ্যাটা বলে আর তার সাথে সাথেই পরীক্ষা করেও দেখে ১০০০ হলো কী না। আসলে তুমি মানুষ হিসাবে বুদ্ধিমান হলেও মনে মনে তুমি কিন্তু প্রতিবার সেই একই কাজই করো, অর্থাৎ পরীক্ষা করে দেখো ১০০০ হয়ে গেলো নাতো, নাহলে ১ যোগ করে পরের সংখ্যায় চলে যাও! উপরের ক্রমলেখ (program) ঠিক এই কাজটাই করে।

এবার for(int gunti = 1; gunti<=1000; ++gunti) অংশটুকু লক্ষ্য করো। আমরা একটা চলক (variable) নিয়েছি gunti যার আদি মান (initial value) ১, আমরা ১ থেকে গুনতে শুরু করবো, তারপর শর্ত পরীক্ষা (condition check) করে দেখবো gunti এখনো ১০০০ বা ছোট কিনা, আর শর্ত সত্য হলে gunti এর মান এক বাড়াবো। তবে এই শর্ত gunti <= 1000 আর হালায়ন (update) ++gunti এই দুয়ের মধ্যে আসলে for এর নীচে যে বিবৃতি (statement) আছে cout << "cpp "; সেটা নির্বাহ করবো। আর gunti বাড়ানোর পরে আবার শর্ত পরীক্ষা করবো, শর্ত সত্য হলে gunti আবার বাড়াবো, দুইয়ের মাঝে অতি অবশ্যই coutএর বিবৃতিটা নির্বাহ করবো, এইটা বারবার করতে থাকবো যতক্ষণ শর্তটা সত্য আছে। শর্ত মিথ্যা হয়ে গেলে বিবৃতিটাও নির্বাহ করবো না, guntiও বাড়াবো না, শর্ত পরীক্ষার পরপরই নিয়ন্ত্রণ (control) চলে যাবে ঘূর্ণীর বাইরে ধরো এইক্ষেত্রে যেখানে আছে cout << endl; সেখানে।

```
int gunti = 1;     // এটি for এর ১ম অংশ মাত্র একবার হবে

if (gunti <= 1000)           // ২য় অংশ শর্ত বারবার হবে
  { cout << "cpp "; ++gunti; } // বিবৃতি, ৩য় অংশ বারবার হবে
if (gunti <= 1000)           // ঠিক উপরের দুই সারিই আরেকবার
  { cout << "cpp "; ++gunti; }
//এইরকম মোট ১০০০ বার উপরের দুই সারি আছে বলে আমরা ধরে নিতে পারি
if (gunti <= 1000)           // ১০০০ তম বার ওই দুই সারিই
  { cout << "cpp "; ++gunti; }

cout << end;                 // ঘূর্ণীর বাইরে অন্য কিছু
```

কী চমৎকার না মাত্র তিন সারির ক্রমলেখ (program) কী ভাবে বার বার একই কাজ করে কত বড় একটা ক্রমলেখয়ের সমতুল কাজ করে দেয়। ঘূর্ণীর (loop) মুল গুরুত্ব কিন্তু এইখানেই। এই পর্যায়ে তুমি হয়তো বলতে পারো আমি ১ থেকে ১০০০ বার না গুনে যদি ০ থেকে ৯৯৯ বার গুনি তাহলে কি কাজ হবে। নিশ্চয় হবে কারণ ০ থেকে ৯৯৯ পর্যন্ততো মোট ১০০০টা সংখ্যাই হয়। কেউ চাইলে ২ থেকে ১০০১ পর্যন্তও গুনতে পারে, কারণ তাতেও ওই ১০০০টা সংখ্যাই আছে। আসলে এই ক্ষেত্রে তো কত থেকে শুরু আর কত তে গিয়ে শেষ সেটা কোন ব্যাপারই না, মুল ব্যাপার হলো মোট কতটি সংখ্যা আছে তাদের মধ্যে।

```
for(int gunti = 1; gunti <= 1999; gunti += 2)
  cout << "cpp ";
cout << endl;
```

তোমাদের মধ্যে কেউ কেউ আবার বলে বসতে পারো নাহ আমি ১ থেকে এক এক করে বাড়িয়ে ১০০০ পর্যন্ত গুনবো না, আমি গুনবো ১ থেকে দুই দুই করে বাড়িয়ে ১৯৯৯ পর্যন্ত। তাতেও কোন সমস্যা নাই ১, ৩, ৫, ..., ১৯৯৯ পর্যন্ত কিন্তু মোট ১০০০টি সংখ্যাই আছে। আমরা ঠিক উপরে দেখিয়েছি, এটি কী ভাবে করতে হবে। তুমি চাইলে কিন্তু ৩ করেও বাড়াতে পারো, বা অন্য কিছু করেও। একটা ব্যাপার দেখো এক করে বাড়ানোর সময় ++gunti না লিখে আমরা কিন্তু চাইলে gunti++ অথবা gunti += 1ও লিখতে পারতাম, একটু ধীর গতির হলেও কাজ অনুযায়ী সবগুলো একই।

```
for(int gunti = 0; gunti < 1000; ++gunti)
  cout << "cpp ";
cout << endl;
```

এবার একটা গুরুত্বপূর্ণ প্রথা (custom) আলোচনা করি। সিপিপিতে আমরা সাধারণ যে কোন গুনতি শুরু করি ০ থেকে। এটির নানাবিধ কারণ আছে, কারণগুলো আমরা যথা প্রসঙ্গে পরে জানবো। তবে মোদ্দা কথা বলতে পারো সংখ্যা আসলে শুরু হয় শুন্য থেকে। তাহলে আমাদের গুনতি হবে ০, ১, ২, ..., ৯৯৯ পর্যন্ত। কিন্তু এখানে যে ১০০০টি সংখ্যা আছে সেটি বুঝানোর জন্য আমরা শর্তে gunti <= 999 না লিখে বরং লিখবো gunti < 1000 যাতে চোখে দেখে আমরা সহজেই বুঝতে পারি যে ১০০০ বার ঘূর্ণীটা (loop) চলবে। তাহলে ধরে নিতে পারো আমরা সচরাচর ঠিক উপরের মতো করে ০ থেকে ১০০০ এর ছোট পর্যন্ত ঘূর্ণী লিখবো, অন্যভাবে হয়তো নয়।

```
int shuru = 0;     // শুরু হবে ০ থেকে
int barbe = 1;     // বাড়বে ১ করে
int mot = 1000;    // মোট কত বার
for(int gunti = shuru; gunti < mot; gunti += barbe)
  cout << "cpp ";
cout << endl;
```

তুমি কিন্তু চাইলে শুরু কত থেকে হবে, কতবার ঘূর্ণী চলবে, প্রতি পাকে কত করে বৃদ্ধি হবে, এসব বিষয় উপরের মতো করে চলক (variable) ব্যবহার করেও করতে পারো। আর চলকের মান আদি আরোপণ (initial assignment) করতে পারো, কোন ভাবে হিসাব করে আরোপণ করতে পারো, অথবা নীচের মতো করে যোগানও (input) নিতে পারো। চলকের মান যদি যোগান নাও তাহলে বুঝতেই পারছো আমরা আর ক্রমলেখ (program) রচনার সময় আগে থেকে জানিনা ঘূর্ণী (loop) ঠিক কতবার ঘুরবে, সেটা জানা যাবে কেবল ক্রমলেখ নির্বাহ (execute) করার সময়। সুতরাং সেই অর্থে চলক ব্যবহৃত এই ঘূর্ণীগুলোকে ঠিক বিস্তারণ করা যাবে না। খেয়াল

করেছো আমরা কিন্তু এই আলোচনার মাধ্যমে এই পাঠের শুরুর দিকে আলোচিত আমাদের দ্বিতীয় সমস্যাটির সমাধান পেয়ে গেলাম। কেমন চমৎকার ব্যাপার তাই না!

ফিরিস্তি ৮.১: বারবার একই জিনিস দেখানো (Repeatedly Display the Same)

```
int motgunti;
cout << "kotobar? ";
cin >> motgunti;
for(int gunti = 0; gunti < motgunti; ++gunti)
  cout << "cpp ";
cout << endl;
```

তাহলে সব মিলিয়ে আমরা দেখলাম জন্য ঘূর্ণীতে (for loop) চারটি অংশ আছে: আদ্যায়ন (initialisation), শর্ত (condition), হালায়ন (update), বিবৃতি (statement)। এর মধ্যে আদ্যায়ন (initialisation), শর্ত (condition), আর হালায়ন (update) থাকে () গোল বন্ধনী (round brackets) যুগলের মধ্যে, আর ; দির্তি (semicolon) দিয়ে পৃথক করা থাকে। এছাড়া বন্ধনীর সামনে থাকে for আর বন্ধনীর পরে থাকে বিবৃতি (statement)। আদ্যায়ন (initialisation) একবার ঘটে, তারপর শর্ত, বিবৃতি, হালায়ন এই তিনটি এই ক্রমে বার বার ঘটতে থাকে যতক্ষণ শর্ত সত্য হয়, আর শর্ত মিথ্যা হলে ঘূর্ণী (loop) শেষ হয়ে যায়।

```
for(আদ্যায়ন; শর্ত; হালায়ন)
  বিবৃতি
```

৮.২ জন্য ঘূর্ণীর মহল্লা (For Loop Block)

এমন একটি ক্রমলেখ (program) রচনা করো যেটি প্রথমে শ্রেণীতে ছাত্র সংখ্যা যোগান (input) নেবে। তারপর প্রতিটি ছাত্রের গণিতে প্রাপ্ত নম্বর যোগান নিয়ে ফলনে (output) ছাত্রটির ফলাফল পাশ না ফেল তা দেখাবে, যেখানে পাশের মান ৫০ বা বেশী।

ফিরিস্তি ৮.২: শ্রেণীতে গণিতের পাশ ফেল (Pass Fail in Mathematics Class)

```
cout << "chhatro sonkhya koto? ";
int sonkhya; cin >> sonkhya;

cout << endl;
for(int suchok = 0; suchok < sonkhya; ++suchok)
{         // suchok চলকটি মহল্লার ভিতরে একটি স্থানীয় চলক
  cout << "kromik " << suchok + 1;
  cout << " chhater nombor koto? ";
  int nombor; cin >> nombor;

  cout << "folafol: ";
  cout << (nombor >= 50 ? "pass" : "fail");
  cout << endl << endl;
}
```

উপরে ক্রমলেখয়ের (program) মুল অংশ দেখানো হলো, আর নীচে রয়েছে ক্রমলেখটি সংকলন (compile) করে চালালে (run) কেমন যোগান-ফলন (input-output) হতে পারে তার নমুনা। এখানে আমরা কেবল মাত্র তিনজন ছাত্রের জন্য ক্রমলেখটি চালিয়ে ফলন দেখিয়েছি, তুমি চাইলে আরো বেশী জনের জন্যেও চালাতে পারো।

যোগান-ফলন (input-output)

```
chhatro sonkhya koto? 3

kromik 1 chhater nombor koto? 80
folafol: pass

kromik 2 chhater nombor koto? 35
folafol: fail

kromik 3 chhater nombor koto? 50
folafol: pass
```

এবার আমরা ক্রমলেখটি (program) বিশ্লেষণ করি। প্রথমে ছাত্র সংখ্যা কত সেটার জন্য যাচনা বার্তা (prompt message) দেখিয়ে আমরা sonkhya চলক (variable) ঘোষণা করে তাতে ছাত্র সংখ্যা যোগান (input) নিয়েছি। তারপর আমরা জন্য ঘূর্ণী (for loop) লিখেছি যেটা চলবে suchok চলকের মান ০ থেকে sonkhya এর কম পর্যন্ত অর্থাৎ মোট sonkhya সংখ্যক বার। খেয়াল করো প্রতি পাকে ঘূর্ণীটায় কেবল একটা সরল বিবৃতি (simple statement) নির্বাহ (execute) করলেই হবে না, বরং প্রতি পাকে আমাদের দরকার অনেকগুলো বিবৃতি নির্বাহ করা। আমরা তাই {} বাঁকা বন্ধনী (curly brackets) ব্যবহার করে একটা মহল্লা (block) নিবো আর তার ভিতরে দরকার মতো যতগুলো বিবৃতি দরকার তা রাখবো। এখানে বলে রাখা দরকার যে suchok চলকটি জন্য ঘূর্ণীর (for loop) যেখানে ঘোষণা করা হয়েছে তাতে এটি একটি স্থানীয় চলক (local variable) যা কার্যকর থাকবে কেবল ওই মহল্লার (block) ভিতরে।

মহল্লার (block) ভিতরে আমরা যেটা করবো সেটা হল প্রত্যেক ছাত্রের প্রথমে ক্রমিক দেখিয়ে তার গণিতে প্রাপ্ত নম্বর যাচনা (prompt) করবো। লক্ষ্য করো প্রথম পাকে suchok চলকের মান ০ কিন্তু প্রথম ছাত্রটির ক্রমিক হবে ১, দ্বিতীয় পাকের সময় suchok চলকের মান হবে ১ কিন্তু দ্বিতীয় ছাত্রের ক্রমিক হবে ২, এইভাবে চলবে। এতে বুঝায় যায় কোন পাকে suchok চলকের মান যত সেই পাকের ছাত্রটির ক্রমিক হবে তার চেয়ে এক বেশী। যাইহোক যাচনা (prompt) করার পর আমরা নম্বর যোগান নিয়েছি nombor চলকে। তারপর একটি তিনিক অণুক্রিয়া (ternary operator) ব্যবহার করে ৫০ এর বেশী হলে পাশ আর নাহলে ফেল ফলনে দেখিয়েছি। আমরা চাইলে যদি নাহলে (if else) ব্যবহার করেও কাজটি করতে পারতাম।

ঘূর্ণীতে (loop) মহল্লা (block) ব্যবহার করা যখন শিখলাম, তখন বলে রাখা দরকার যে মহল্লার (block) ভিতরে আমরা কিন্তু যে রকম ইচ্ছা বিবৃতি (statement) যতগুলো ইচ্ছা বিবৃতি লিখতে পারি। আমরা আগেই জানি ; দির্তি (semicolon) দিয়ে শেষ হওয়া কোন কিছুই হলো একটা বিবৃতি। তবে যে কোন সরল বিবৃতির (simple statement) পাশাপাশি আমরা যৌগিক বিবৃতিও (compound statement) ব্যবহার করতে পারি। যদি নাহলে (if else), অন্তাঅন্তি যদি নাহলে (nested if else), যদি নাহলে মই (if else ladder), তিনিক অণুক্রিয়া (ternary operator), পল্টি ব্যাপার (switch cases) এগুলো সবগুলোই একক যৌগিক বিবৃতি (unit compound statement)। একক বিবৃতি হওয়ায় এগুলোর যে কোন একটা ব্যবহার করলে

মহল্লা (block) তৈরী করার দরকার নাই, করলেও সমস্যা নাই, কিন্তু এগুলোর একাধিক ব্যবহার করলে অবশ্যই মহল্লা (block) তৈরী করে নিতে হবে। আমরা কিন্তু উপরের মহল্লায় তিনিক অণু-ক্রিয়া (ternery operator) সহ অন্য কয়েকটি সরল বিবৃতি (simple statement) একসাথে দেখিয়েছি। ঘূর্ণীতে (loop) মহল্লা (block) ব্যবহারের সাথে এটাও বলে রাখা দরকার যে মহল্লার ভিতরে আবারও এক বা একাধিক ঘূর্ণী (loop) আমরা চাইলেই লিখতে পারি। তবে আমরা সেই আলোচনা করবো অন্তান্তি ঘূর্ণী (nested loop) প্রসংগের সময়।

৮.৩ পাকের সূচক ও পরম্পরা (Loop Index and Succession)

এমন একটি ক্রমলেখ (program) রচনা করো যেটি একটি সমান্তর ধারার প্রথম পদ a, সাধারণ অন্তর d, ও পদ সংখ্যা n যোগান (input) নিয়ে পদ গুলোকে নীচের মতো করে ফলন (output) দিবে। খেয়াল করো পদগুলো যোগ চিহ্ন দিয়ে আর যোগফল সমান চিহ্ন দিয়ে দেখানো হয়েছে।

```
1 + 3 + 5 + 7 + 9 = 25
```

আমরা জানি সমান্তর ধারার প্রথম পদ a, সাধারণ অন্তর d হলে kতম পদ হল $a+(k-1)d$, আর n পদের যোগফল হলো $n*(2a+(n-1)d)/2$। সুতরাং একটি জন্য ঘূর্ণী (for loop) আমরা 1 থেকে n পর্যন্ত চালিয়ে কাজটি খুব সহজেই করে ফেলতে পারি।

ফিরিস্তি ৮.৩: পাটিগণিতের ধারার সমস্যা (Arithmetic Series Problem)

```
int a, d, n;                    // তিনটি চলক লাগবে
cout << "a d n koto? ";         // যোগান যাচনা করো
cin >> a >> d >> n;             // মান যোগান নাও

if (n <= 0)          // অধনাত্নক পদসংখ্যা গ্রহণযোগ্য নয়
{
  cout << "n er man dhonatok hote hobe" << endl;
  return EXIT_FAILURE;      // এ ক্ষেত্রে ক্রমলেখ ব্যর্থ
}

// ঘূর্ণী চালাও 1 হতে n পর্যন্ত
for (int k = 1; k <= n; ++k)
{
  int t = a + (k - 1) * d;      // k-তম পদটি হিসাব করো
  cout << " + " << t;           // ফাঁকা ও যোগ চিহ্ন সহ ফলন
}

int s = n * (2*a + (n - 1) * d) / 2;  //  যোগফল হিসাব
cout << " = " << s << endl;     // ফাঁকা ও সমান চিহ্ন সহ
```

তবে উপরের ক্রমলেখয়ের (program) ফলনের (output) দিকে তাকালে একটা ছোট সম-স্যা দেখতে পাবে। একটু খেয়াল করলে দেখবে ফলনে একদম প্রথম পদটির সামনেও যোগ চিহ্ন চলে এসেছে, যেটি আসলে চাওয়া হয় নি। এটি সমাধাণের উপায়ও খুব সহজ, একটা যদি (if)

লাগিয়ে যোগ চিহ্নটা k এর মান 1 ছাড়া অন্য যে কোন সময় দেখাবো, আর পদের মানটা তো প্রতিবারই দেখাতে হবে। নীচের ক্রমলেখ দেখো।

```
// ঘূর্ণী চালাও 1 হতে n পর্যন্ত
for (int k = 1; k <= n; ++k)
{
  int t = a + (k - 1) * d;      // k-তম পদটি
  if (k != 1)                   // প্রথম পদ ছাড়া
    cout << " + ";              // ফাঁকা ও যোগ চিহ্ন
  cout << t;                    // পদটি দেখাও
}
```

ঠিক উপরের এই ক্রমলেখটি ঠিক মতো ফলন (output) দিলেও এটা আসলে সর্বোত্তম নয়। কারণ যোগ চিহ্ন না দেওয়ার ব্যাপারটি কেবল প্রথম পদের জন্য, বাঁকী সবগুলোর জন্য যোগ দেখাতে হবে। কিন্তু ওই শর্ত পরীক্ষণটি প্রতিবার প্রতিটি পদের জন্য করতে হচ্ছে, কারণ শর্ত পরীক্ষাটিতো ঘূর্ণীর ভিতরে রয়েছে। আমরা যদি শর্ত পরীক্ষাটা এড়াতে চাই, তাহলে আমাদের বুঝতে হবে যে প্রথম পদ ছাড়া বাঁকী পদগুলো একরকমের, তাদের সবার সামনে যোগ চিহ্ন আছে, কাজেই কেবল তারাই ঘূর্ণীর ভিতরে থাকবে। আর প্রথম পদটিকে সেক্ষেত্রে আলাদা করে ফেলতে হবে। নীচের ক্রমলেখ (program) দেখো শর্ত পরীক্ষণ আর করতে হয় নি, কারণ ঘূর্ণী চলেছে 2 হতে n পর্যন্ত। আর প্রথম পদটি ঘূর্ণীরও আগে দেখানো হয়েছে।

```
cout << a;  //  প্রথম পদটি ঘূর্ণীর বাইরে, এর সামনে যোগ নাই

//  k = 2 হতে n বাঁকী (n-1)টি যোগওয়ালা পদ ঘূর্ণীর ভিতরে
for (int k = 2; k <= n; ++k)
{
  int t = a + (k - 1) * d;      // k-তম পদটি হিসাব করো
  cout << " + " << t;           // ফাঁকা ও যোগ চিহ্ন সহ ফলন
}
```

উপরের এই ক্রমলেখটিতে আরেকটি বিষয় লক্ষ্য করতে পারো। ঘূর্ণীর (loop) প্রতি পাকে আমরা kতম পদ হিসাব করছি $a + (k-1) * d$ রাশিটি (expression) হতে, যেখানে একটি যোগ, একটি বিয়োগ ও একটি গুণ করতে হচ্ছে। আসলে এখানে আমরা প্রতিটি পদ সরাসরি হিসাব করছি কততম পদ সেটা অর্থাৎ k থেকে। কিন্তু তা না করে আমরা অন্য একটা কাজ করতে পারি।

```
//  k = 2 হতে n বাঁকী (n-1)টি যোগওয়ালা পদ ঘূর্ণীর ভিতরে
for (int k = 2, t = a; k <= n; ++k) // আদ্যায়নে t
{
  t += d;                       // আগের মানের সাথে d যোগ
  cout << " + " << t;           // ফাঁকা ও যোগ চিহ্ন সহ ফলন
}
```

যেহেতু সমান্তর ধারার যে কোন দুটো পদের মধ্যে ব্যবধান d, আমরা তাই কেবল আগের পদের সাথে d যোগ করেই পরের পদ পেয়ে যেতে পারি। তাতে প্রতিটি পদ হিসাব করতে কেবল একটা যোগ করলেই চলবে। এক্ষেত্রে আমরা t চলকটিকে ঘূর্ণীর আদ্যায়নে (initialisation) ঘোষণা

করে আদি মান দিতে পারি প্রথমপদটির মানের সমান। আর প্রতি পাকে আগে t এর মান d পরিমান বাড়ানো হবে তারপর ফলন দেওয়া হবে। স্বাভাবিক ভাবে এই ক্রমলেখটি বেশী দক্ষ হয়েছে।

এবার একটু ভিন্ন আলোচনা। জন্য ঘূর্ণীর (loop) আদ্যায়নে (initialisation) আমরা চাইলে , বির্তি (comma) দিয়ে একাধিক চলক ঘোষণা ও আদি মান দিতে পারি। এই চলকগুলো কিন্তু ঘূর্ণীর সাথে যে বিবৃতি (statement) বা মহল্লা (block) কেবল সেখানে স্থানীয় চলক (local variable) হিসাবে ব্যবহার করা যায়। কোন চলককে যদি ঘূর্ণীর ভিতরে ও বাইরে উভয় ক্ষেত্রে কোন কারণে দরকার হয়, সেই চলকটিকে তাহলে ঘূর্ণীর আগেই ঘোষণা (declare) করে ফেলতে হবে। আমরা সে রকম একটি উদাহরণ দেখি নীচে তবে এক্ষেত্রে আমরা যোগফলটিকে আর $n * (2a + (n-1)d)/2$ সুত্র ব্যবহার করে করবো না, বরং এটিকেও ঘূর্ণী দিয়েই করবো, যদিও সেটা দক্ষ হবে না, ঘূর্ণী ছাড়া সুত্র দিয়ে করলেই বরং দক্ষ (efficient) হবে।

```
int t = a, s = a; // প্রথম পদ ও এক পদের যোগফল।

cout << t;  //  প্রথম পদটি ঘূর্ণীর বাইরে, এর সামনে যোগ নাই

for (int k = 2; k <= n; ++k) // ঘূর্ণী 2 হতে n
{
  t += d;                    // আগের মানের সাথে d যোগ
  s += t;                    // যোগফলের সাথে এই পদ যোগ
  cout << " + " << t;        // ফাঁকা ও যোগ চিহ্ন সহ ফলন
}
cout << " = " << s << endl;   // ফাঁকা ও সমান চিহ্ন সহ
```

৮.৪ ঘূর্ণীতে ক্ষান্তির ব্যবহার (Using Breaks in Loops)

সিপিপি ভাষায় এমন একটি ক্রমলেখ (program) লিখো যেটি তোমার মাধ্যমিক পরীক্ষায় ১০ টি বিষয়ের নম্বর যোগান (input) নিবে। তুমি যদি প্রতিটি বিষয়ে ৫০ এর বেশী করে পেয়ে থাকো তাহলে তোমার গড় নম্বর দেখাবে, আর যদি যে কোন একটি বিষয়েও ফেল করে থাকো, তাহলে বাঁকী বিষয়গুলোর নম্বর যোগান (input) নেয়া বাদ দিয়েই সরাসরি ফলন (output) দেখাবে অকৃতকার্য। এবার ভিন্ন একটা অবস্থা চিন্তা করো যেখানে তোমাকে সবগুলো বিষয়ের নম্বর যোগান নিতেই হবে, একটাতে ফেল করলে তুমি ফলনে অকৃতকার্যও দেখাবে, তবে গড় নম্বর দেখাবে কেবল পাশ করা বিষয়গুলোর নম্বর নিয়ে। এই নতুন ক্রমলেখটি (program) কেমন হবে?

ফিরিস্তি ৮.৪: দশ বিষয়ের পাশ ফেল নির্ণয় (Pass Fail in Ten Subjects)

```
int motnombor = 0, suchok = 0;  // আদিমান শুন্য

for(; suchok < 10; ++suchok)
{
  cout << suchok << "tom bishoyer nombor koto? ";
  int nombor; cin >> nombor;

  if (nombor < 50) break; // ঘূর্ণী থেকে বের হয়ে যাও!
```

```
    motnombor += nombor;
  }

  if (suchok < 10)   // ঘূর্ণী থেকে আগেই বের হয়ে এসেছে
    cout << "okritokarjo" << endl;
  else          // ঘূর্ণী শেষ করে তারপর এখানে এসেছে
  {
    float gornombor = motnombor / 10.0;
    cout << "gor nombor " << gornombor << endl;
  }
```

প্রদত্ত সমস্যা দুটির প্রথমটির সমাধান উপরে দেখানো হয়েছে। তেমন কঠিন কিছু নয়। একটা জন্য ঘূর্ণী (for loop) ব্যবহার করা হয়েছে যদি suchok নামক চলকটির (variable) মান ০ থেকে ১ করে বাড়িয়ে বাড়িয়ে ১০ এর কম পর্যন্ত চলবে। ঘূর্ণীর প্রতি পাকে বাঁকা বন্ধনীর (curly brackets) ভিতরে প্রথমে কততম বিষয়ের নম্বর যোগান (input) নিতে চাই সেটা যোগান যাচনা (input prompt) করবো, তারপর nombor নামক চলকে (variable) সেটি যোগান নিবো। লক্ষ্য করো suchok এর মান ০ হলে আমরা কিন্তু ০তম বিষয়ের নম্বর কত তা জানতে চেয়েছি। অর্থাৎ আমরা গুনতে শুরু করেছি ০ থেকে। যাইহোক nombor চলকে মান যোগান নেওয়ার পরে আমাদের সেটা motnombor চলকের সাথে যোগ করার কথা, কারণ সবশেষে আমরা গড় বের করতে চাই। তবে প্রশ্নে বলা হয়েছে ৫০ বা বেশী হলে গড় বের করতে হবে, ৫০ এর কম হলে আর যোগান (input) না নিয়ে অকৃতকার্য দেখাতে হবে। এই অংশটুকু করতে আমরা যেটি করেছি তা হলো if (nombor < 50) break; এই অংশটুকু লিখেছি।

আমরা পল্টি-ব্যাপার (switch-case) আলোচনা করার সময় ক্ষান্তি (break) এর ব্যবহার দেখেছিলাম। এখানেও ক্ষান্তির (break) এর কাজ প্রায় একই। ক্ষান্তি (break) পাওয়া মাত্রই ওই ক্ষান্তি যে ঘূর্ণীর (loop) ভিতরে, নিয়ন্ত্রণ (control) সেই ঘূর্ণী থেকে বের হয়ে তারপরের অংশে চলে যায়। উপরের ক্রমলেখতে (program) ৫০ এর কম নম্বর যোগান (input) হলেই নিয়ন্ত্রণ ঘূর্ণীর (loop) বাইরে থাকা যদি নাহলেতে (if else) চলে যাবে, আমরা যা চাইছিলাম। এখন দেখো আমরা এই যদি নাহলেতে (if else) suchok চলকের (variable) মান পরীক্ষা করছি ১০ এর কম কিনা। যদি ক্ষান্তি (break) হয়ে নিয়ন্ত্রণ ঘূর্ণী থেকে বের হয়ে এসে থাকে তাহলে suchok চলকের মান অবশ্যই ১০ এর কম হবে, সর্বোচ্চ ৯ হবে। আর যদি ঘূর্ণীর সবগুলো পাক শেষ করে এসে থাকে তাহলে অবশ্যই suchok এর মান ১০ বা বেশী, কারণ ঘূর্ণী (loop) চলার শর্তই তো হলো suchok চলকের মান ১০ এর কম হতে হবে। তো if else এ suchok < 10 হলে আমরা অকৃতকার্য দেখিয়েছি, আর নাহলে motnombor কে প্রথমে ১০ দিয়ে ভাগ করে গড় বের করে তারপর সেটা ফলনে (output) দেখিয়েছি।

একটা বিষয় খেয়াল করো এখানে গড় নম্বরটি ভগ্নক (fractioner) হবে তাই সেটা ধারণ করার জন্য আমরা float ধরণের চলক gornombor নিয়েছি। আর motnomor কে স্রেফ ১০ দিয়ে ভাগ করলে আমরা আসলে কোন ভগ্নক পাবো না, বরং কারণ দুটো পূর্ণকের (integer) ভাগফলও পূর্ণক আমরা জানি। কিন্তু একটি পূর্ণক ও একটি ভগ্নকের ভাগফল আবার একটি ভগ্নক। তাই কৌশল হিসাবে আমরা 10 না লিখে লিখেছি 10.0 যাতে ভাগফল আসে ভগ্নক (fractioner) হিসাবে। আর একটি বিষয় খেয়াল করো, suchok চলকটিকে আমরা কিন্তু for(int suchok = 0; suchok < 10; ++suchok) লিখে ঘূর্ণীর (loop) ভিতরে স্থানীয় চলক (local variable) হিসাবে ঘোষণা করি নাই। আমরা বরং suchok ঘোষণা করেছি ঘূর্ণীর (loop) বাইরে আর আদি

মানও (initial value) দিয়েছি ঘূর্ণীর বাইরে। এর কারণ হলো suchok চলকটিকে যেহেতু আমরা জন্য ঘূর্ণীর (for loop) বাইরে যদি নাহলেতে (if else) ব্যবহার করতে চাই, তাই এটিকে ঘূর্ণীর ভিতরে ঘোষণা করা যাবে না, সেক্ষেত্রে ঘূর্ণীর বাইরে সেটি আর কার্যকর থাকবে না বলে। তুমি কিন্তু চাইলে কেবল ঘোষণাটা ঘূর্ণীর বাইরে করে আদিমান দেওয়াটা ঘূর্ণীতেই করতে পারতে। আর একটা বিষয়ও তাহলে ফাঁক তালে আমরা জানলাম সেটা হলো জন্য ঘূর্ণীর (for loop) এর আদ্যায়ন (initialisation) অংশে কিছু না থাকলেও কোন সমস্যা নেই।

```
int motnombor = 0, motbishoy = 0, suchok;
bool fellhoise = false; // আদি মান

for(suchok = 0; suchok < 10; ++suchok)
{
  cout << suchok << "tom bishoyer nombor koto? ";
  int nombor; cin >> nombor;

  if (nombor < 50) fellhoise = true;
  else
  {
    motnombor += nombor;
    motbishoy += 1;
  }
}

if (fellhoise)
  cout << "okritokarjo" << endl;
else
{
  float gornombor = motnombor / motbishoy;
  cout << "gor nombor " << gornombor << endl;
}
```

আমাদের যদি সবগুলো বিষয়ের নম্বর যোগান (input) নিতেই হয়, তাহলে আমরা উপরের ক্রমলেখয়ের মতো করে ক্ষান্তি (break) ছাড়া লিখবো। তবে কোন একটা বিষয়ের নম্বর ৫০ এর কম ছিলো কিনা সেটা মনে রাখার জন্য আমরা এখানে বুলক (boolean) ধরণের একটি চলক fellhoise নিয়েছি। এই fellhoise চলকটির মান আদিতে false, কারণ তখনও আমরা একটা বিষয়ও পরীক্ষা করি নাই। তারপর ঘূর্ণীর ভিতরে খেয়াল করো যদি নাহলেতে (if else) নম্বর ৫০ এর কম হলে আমরা fellhoise চলকের মান করে দিয়েছি true। একাধিক বিষয়ের মান ৫০ এর কম হলে সেগুলোর প্রতিবারেই fellhoise চলকের মান true হবে, কিন্তু ঘূর্ণীর (loop) ভিতরে fellhoise চলকের মান false হওয়ার কোন পথ নাই। কাজেই ঘূর্ণীর শেষে fellhoise চলকের মান true মানে হলো এক বা একাধিক বিষয়ের নম্বর ৫০ এর কম, আর fellhoise চলকের মান তখনও false থাকা মানে হলো কোন বিষয়ের নম্বরই ৫০ এর কম ছিলো না। fellhoise ধরনের বুলক চলক (boolean variable) যেগুলো আমরা ঘূর্ণীর ভিতরে কোন প্রদত্ত শর্ত কখনো সত্য হয়েছিলো কিনা মনে রাখতে ব্যবহার করি সেগুলোকে বলা হয় পতাকা চলক (flag variable)।

উপরের ক্রমলেখতে (program) আমরা motnombor হিসাব করেছি কেবল nombor চলকের মান ৫০ বা বেশী হলে, আর এরকম বিষয় কয়টি সেটাও মনে রাখার জন্য motbishoy নামের আরেকটি চলক নিয়ে সেটার মান প্রতিবার ১ করে বাড়িয়েছি। খেয়াল করো motnombor ও motbishoy চলক দুটির ঘোষণা ও আদিমান শুন্য দেয়া হয়েছে ঘূর্ণীর (loop) আগে। এখানে suchok চলকটি আমরা ঘোষণা করেছি ঘূর্ণীর বাইরে, কিন্তু এবার আদি মান দিয়েছি ঘূর্ণীর (loop) আদ্যায়ন (initialisation) অংশে। তোমার ইচ্ছামতো ও দরকারমতো তুমি নানাভাবেই এগুলো করতে পারো। এই পাঠ শেষ করি আরেকটি বিষয় দিয়ে। লক্ষ্য করো motbishoy দিয়ে আমরা যেখানে motnombor কে ভাগ করেছি, সেখানে motbishoy শুন্য কিনা পরীক্ষা করি নাই। তুমি জানো শুন্য দিয়ে ভাগ করলে divide by zero বা শুন্য দিয়ে ভাগ বলে একটি ত্রুটি (error) দেখা দেয়। আমাদের motbishoy এর মান শুন্য পরীক্ষা করা দরকার হয় নাই, কারণ সেটা হওয়া সম্ভব যদি ১০টা বিষয়ের সবগুলোতেই নম্বর ৫০ এর কম হয়। কিন্তু একটা বিষয়েও যদি নম্বর ৫০ এর কম হয় তাহলে তো fellhoise সত্য হয়ে যাবে আর অকৃতকার্য ফলনে (outout) আসবে, গড় নম্বর নয়। কাজেই কোন ত্রুটি আসার সুযোগই তৈরী হবে না এখানে।

৮.৫ ঘূর্ণীতে পাক ডিঙানো (Continue in Loops)

এক ব্যবসায়ী তার খরিদ্দারদের হিসাব সংরক্ষণের জন্যে একটা করে খাতা নম্বর দিয়ে দেয়। তবে কুসংস্কার জনিত কারণে সে মনে করে কোন খরিদ্দারের খাতা নম্বর যদি ১৩ বা এর গুণিতক হয়, তাহলে সেই খরিদ্দার তার জন্যে ক্ষতির কারণ হবে, হয়তো বাঁকী নিবে অথবা বাঁকী নিয়ে পরিশোধ করবে না। এখন তোমাকে এমন একটি ক্রমলেখ (program) লিখতে হবে যেটা শুরুর নম্বর আর শেষের নম্বর যোগান (input) নিয়ে ১৩ দিয়ে বিভাজ্য নম্বরগুলো বাদ দিয়ে অন্য নম্বরগুলো ফলন (output) দিবে, যাতে ওই ব্যবসায়ী নম্বরগুলো খরিদ্দারদের খাতার ওপর লাগাতে পারে।

ফিরিস্তি ৮.৫: দুর্ভাগ্যের সংখ্যা উপেক্ষা (Ignoring Unlucky Numbers)

```
int shuru, shesh; //  চলকগুলোর মান যোগান নিতে হবে

for(int nombor = shuru; nombor <= shesh; ++nombor)
{
  if (nombor % 13 == 0)    // ১৩ দিয়ে বিভাজ্য হলে
    continue;              // পরেরটাতে চলে যাও

  cout << nombor << endl; // অন্যগুলোর জন্য
}
```

এই ক্রমলেখ (program) লেখা খুবই সহজ। তোমার দুটো চলক (variable) shuru আর shesh নিতে হবে। এই দুটোর মান তুমি ব্যবসায়ীর কাছে থেকে যোগান (input) নিবে। তারপর তুমি একটি জন্য ঘূর্ণী (for loop) চালাবে যেটি nombor চলকের মান shuru থেকে shesh পর্যন্ত এক এক করে বাড়িয়ে বাড়িয়ে যাবে। আর ঘূর্ণীর (loop) ভিতরে তুমি যদি (if) ব্যবহার করে পরীক্ষা করে দেখবে nombor চলকের (variable) মান ১৩ দ্বারা বিভাজ্য কিনা। যদি nombor % 13 == 0 শর্ত সত্য হয় তাহলে আমরা সেখানে ওই নম্বর ডিঙিয়ে (continue) চলে যাবো মানে আর কিছু না করে ঘূর্ণীর পরের পাকে চলে যাবো, আর শর্ত মিথ্যা হলে ওই পাক না ডিঙি-য়ে পরে যেখানে ফলন (output) দেওয়া হয়েছে সেটা করবো। জন্য ঘূর্ণীর (for loop) পাক

ডিঙিয়ে (continue) পরের পাকে যাওয়া মানে ঠিক যেখানে continue; লেখা হয়েছে সেখান থেকে নিয়ন্ত্রণ (control) ঘূর্ণীর হালায়ন (update) অংশে ++nombor করতে চলে যাবে, তার ফলে nombor এর মান ১ বাড়বে, আর তারপর নিয়মানুযায়ী এর পরে শর্ত পরীক্ষণ (condition checking), পরের পাকের বিবৃতি নির্বাহ (statement execution), এই ভাবে চলতে থাকবে। জন্য ঘূর্ণীতে (for loop) ডিঙানোর (continue) এর কাজ মূলত এতটুকুই।

```
int shuru, shesh; //  চলকগুলোর মান যোগান নিতে হবে

for(int nombor = shuru; nombor <= shesh; ++nombor)
{
  // এই খানে এক গাদা কাজ থাকতে পারে, তাই মহল্লা নেয়া হয়েছে

  if (nombor % 13 != 0)    // ১৩ দিয়ে বিভাজ্য না হলে
  {
    cout << nombor << endl;

    // এই খানে আরো এক গাদা বিবৃতি থাকতে পারে।
    // এক গাদা বিবৃতি না থাকলে মহল্লা {} লাগবে না।
  }
}
```

উপরের ক্রমলেখ (program) দেখো। যে কাজ ডিঙানো (continue) ব্যবহার করে আমরা সম্পন্ন করেছিলাম, সেটা ডিঙানো (continue) ছাড়াই করা হয়েছে। যেহেতু ১৩ দিয়ে বিভাজ্য না হলে আমাদের নম্বরগুলো ফলনে (output) দেখাতে হবে, আমরা তাই শর্তটা উল্টে দিয়ে nombor % 13 != 0 লিখেই কাজটি করে ফেলতে পারি। ১৩ দ্বারা বিভাজ্য হওয়ার ক্ষেত্রে তো আমাদের তাহলে আর কিছু করার নাই, সুতরাং নিয়ন্ত্রণ (control) আপনা আপনিই হালায়ন (update) অংশে চলে যাবে। তারমানে বলতে গেলে ডিঙানো (continue) একটা অদরকারী বিষয়। তাহলে এইটা আছেই বা কেন? একটা বিষয় খেয়াল করো ১৩ দ্বারা বিভাজ্য না হলে এই ক্ষেত্রে আমাদের কেবল একটাই কাজ, নম্বরটা ফলনে দেখানো। আমাদের এইটাকে কোন মহল্লার (block) ভিতরে রাখার দরকার নাই, যদিও আমরা উপরে আমরা সেটা রেখেছি। মহল্লা (block) মূলত দরকার যদি ওইখানে আমাদের এক গাদা বিবৃতি নির্বাহ (execute) করতে হয়। তো এই বিশাল এক গাদা বিবৃতি কে একটা যদি (if) এর মহল্লার ভিতরে ঢুকিয়ে দেওয়ার চেয়ে ডিঙানো (continue) ব্যবহার করে ক্রমলেখ (program) লিখলে বুঝতে সুবিধা হয়।

৮.৬ জন্য ঘূর্ণীতে হ্রাসের ব্যবহার (For Loop and Decrement)

তুমি দশতলা দালানের ১ তলা থেকে উত্তোলকে (elevator) করে দশ তলায় উঠতে ও নামতে চাও। তো উত্তোলককে এক তলা হতে আরেক তলায় যাওয়ার জন্য প্রতিবার আলাদা করে নির্দেশ দিতে হয়। এবার এমন একটি ক্রমলেখ (program) রচনা করো যেটি তোমার হয়ে উত্তোলককে একের পর এক উঠার ও তারপর নামার নির্দেশ দিবে।

এই ক্রমলেখয়ের (program) উঠার অংশ তো খুবই সহজ। নীচের ক্রমলেখ (program) দেখো, মূলত জন্য ঘূর্ণী (for loop) দেখবে। ঘূর্ণী tola চলকের (variable) মান ১ হতে ৯ (বা ১০ এর কম) পর্যন্ত ১ বাড়িয়ে বাড়িয়ে চলবে আর প্রতিবারে ফলনে (output) দেখাবে tola হতে

tola + 1 এ উঠতে হবে অর্থাৎ কোন পাকে tola চলকের মান ৪ হলে দেখাবে ৪ হতে ৫। তলা ৯ হতে ১০ এ উঠে আর উঠতে হবে না তাই ঘূর্ণী কিন্তু tola = 10 এর জন্য ঘুরবে না।

ফিরিস্তি ৮.৬: দশতলায় উঠা-নামা (Ten Floor Up Down)

```
// উঠার অংশ
cout << "tola 1" << endl;
cout << "utha shuru" << endl;
for(int tola = 1; tola < 10; ++tola)
  cout << tola << " hote " << tola+1 << endl;
cout << "utha shesh" << endl;

cout << "tola 10" << endl;

// নামার অংশ
cout << "nama shuru" << endl;
for (int tola = 10; tola > 1; --tola)
  cout << tola << " hote " << tola-1 << endl;
cout << "nama shesh" << endl;

cout << "tola 1" << endl;
```

এবার আসা যাক ক্রমলেখয়ের পরের অংশে। এখানে ১০ থেকে নামা শুরু, প্রথমে ১০ থেকে ৯ এ, তারপর ৯ থেকে ৮ এ, এইভাবে ২ থেকে ১ এ গিয়ে শেষ, ১ থেকে আর নামার ব্যাপার নাই। কাজেই এইখানেও আমরা একটা ঘূর্ণী (loop) ব্যবহার করবো। এই ঘূর্ণী চলবে tola চলকের মান ১০ হতে ২ (বা ১ এর বেশী) পর্যন্ত, আর প্রতিবারে tola চলকের মান এক কমবে, অর্থাৎ জন্য ঘূর্ণীর (for loop) হালায়ন অংশে ++tola না লিখে লিখবো --tola। ব্যস হয়ে গেলো।

এবার একটা খুচরা বিষয়। এখানে tola চলকটি (variable) আমরা দুইবার ঘোষণা করেছি, দুই ঘূর্ণীতে (loop) দুইবার। যেহেতু জন্য ঘূর্ণীর (for loop) অংশে চলকগুলো স্থানীয় চলক (local variable) হিসাবে ঘোষণা (declare) করা হয়েছে, সেহেতু চলক দুটির কার্যকারীতা কিন্তু সংশ্লিষ্ট ঘুর্ণীর ভিতরেই শেষ। কাজেই প্রতিবার ঘোষিত চলক আসলে নাম একই হলেও আলাদা আলাদা চলক। তুমি যদি কেবল একবার চলক ঘোষণা করে কাজ সারতে চাও সেটাও করতে পারবে। প্রথম ঘূর্ণীরও (loop) আগে int tola; লিখে চলক ঘোষণা একবারই করে ফেলো আর ঘূর্ণী দুটোর আদ্যায়ন (initialisation) অংশে int tola = 1; না লিখে স্রেফ tola = 1; করে দাও। তাহলে একই tola চলক উভয় ঘূর্ণীতে ব্যবহৃত হলো।

৮.৭ জন্য ঘূর্ণীতে ফাঁকা শর্ত (For Loop Empty Condition)

জন্য ঘূর্ণীতে (for loop) শর্ত ফাঁকা রাখলে কী ঘটে? জন্য ঘূর্ণীতে শর্ত ফাঁকা রেখে এমন একটি ক্রমলেখ (program) রচনা করো যেটি দশটি ধনাত্মক পূর্ণক (positive integer) যোগান (input) নিবে। যদি শুন্য বা ঋণাত্মক (negative) পূর্ণক যোগান দেয়া হয়, সেটা উপেক্ষা করবে। ক্রমলেখটি এরপর ধনাত্মক সংখ্যা দশটির যোগফল ফলনে (output) দেখাবে।

নীচের ক্রমলেখ (program) দেখো। এতে শর্ত ফাঁকা রাখা হয়েছে। আমরা gunti চলকটিকে (variable) চাইলে জন্য ঘূর্ণীতে (for loop) ঘোষণা (declare) করতে পারতাম, কিন্তু jogfol

৮.৭. জন্য ঘূর্ণীতে ফাঁকা শর্ত (For Loop Empty Condition)

চলকটিকে অবশ্যই ঘূর্ণীর বাইরেই ঘোষণা করতে হবে। ঘূর্ণীর (loop) ভিতরে দেখো যোগান যাচনা (input prompt) করে নম্বর যোগান নেয়া হয়েছে। যোগান নেয়া নম্বরটি যদি শুন্য বা ঋণাত্নক হয় তাহলে পাক ডিঙাতে হবে তাই continue; দেয়া হয়েছে, আর ধনাত্নক হলে যোগফল বাড়বে, আর গুনতিও বৃদ্ধি (increment) পাবে। গুনতির মান বাড়ানোর পরেই আমরা একটি যদি নাহলে (if else) লাগিয়ে পরীক্ষা করে দেখতে পারি gunti চলকের মান ১০ হলো কিনা। যদি হয়ে থাকে তাহলে আমাদের আর ঘূর্ণী (loop) চালিয়ে যাওয়া উচিৎ হবে না। আমরা তাই break ; লাগিয়ে ঘূর্ণীতে ক্ষান্তি দিবো। আর gunti চলকের (variable) মান ১০ না হয়ে থাকলে পরে নিয়ন্ত্রণ (control) জন্য ঘূর্ণীর (for loop) হালায়ন (update) অংশে যাবে, সেখানে তো কিছু হবে না, কারণ সেটি ফাঁকা। তারপর শর্ত (condition) অংশে যাবে, সেখানেও ফাঁকা। কিন্তু একটা বিষয় মনে রাখবে ফাঁকা শর্ত মানে সব সময় সত্য অর্থাৎ এই ক্ষেত্রে for (; ;) আর for (; true ;) একই কথা। যাই হোক শর্তা সত্য হওয়ায় পরের পাক যথারীতি শুরু হবে। তুমি যদি কোন কারণে if (gunti >= 0) break; এই শর্ত যুক্ত ক্ষান্তি এই জন্য ঘূর্ণীতে না দাও তাহলে কিন্তু ঘূর্ণী থেকে বের হয়ে যাওয়ার আর কোন পথ রইলো না। ওদিকে ফাঁকা শর্ত তো সবসময় সত্য রয়েছে। এমতাবস্থায় এই ঘূর্ণী (loop) অসীম সংখ্যক বার ঘূরতে থাকবে।

```
int gunti = 0, jogfol = 0;

for( ; ; )       // শর্ত ফাঁকা
{
  cout << "nombor koto? ";
  int nombor; cin << mombor;

  if (nombor <= 0)  // ধনাত্নক না হলে
    continue;       // পাক ডিঙাও

  jogfol += nombor; // যোগফল

  ++gunti;          // হালায়ন

  // নীচের শর্ত যুক্ত ক্ষান্তি না দিলে অসীম ঘূর্ণী হবে
  if (gunti >= 10)  // শর্ত এখানে
    break;          // ঘূর্ণীতে ক্ষান্তি
}

cout << "jogfol " << jogfol << endl;
```

যে ঘূর্ণী (loop) অসীম সংখ্যক বার ঘুরে, আর ঘূর্ণী থেকে বের হওয়ার কোন সুযোগ নাই, এ রকম ঘূর্ণীকে বলা হয় অসীম ঘূর্ণী (infinite loop)। ফাঁকা শর্তা ছাড়াও অসীম ঘূর্ণী তৈরী হতে পারে, যদি তোমার শর্ত এমন হয় যে সেটা সবসময় সত্য, যেমন ধরো gunti == gunti এই শর্তটিও সর্বদা সত্য, কাজেই এটাও অসীম ঘূর্ণী তৈরী করবে। অসীম ঘূর্ণী তৈরী হওয়া মানে এই ক্রমলেখ কোন দিনই থামবে না। ক্রমলেখতে (program) ঘূর্ণী (loop) তৈরী করলেই আমাদের তাই অতিরিক্ত সতর্ক থাকতে হয় যাতে সেটা কোন ভাবেই অসীম ঘূর্ণী না হয়ে যায়।

৮.৮ জন্য ঘূর্ণীতে ফাঁকা হালায়ন (For Loop Empty Update)

এমন একটি ক্রমলেখ (program) রচনা করো যেটি দশটি ধনাত্মক পূর্ণক (positive integer) যোগান (input) নিবে। যদি শুন্য বা ঋণাত্মক (negative) পূর্ণক যোগান দেয়া হয়, সেটা উপেক্ষা করবে। ক্রমলেখটি এরপর ধনাত্মক সংখ্যা দশটির যোগফল ফলনে (output) দেখাবে।

এই ক্রমলেখটি লেখা একদমই সোজা। এখানে মূলত আমরা দেখতে চাই যে জন্য ঘূর্ণীতে (for loop) আদ্যায়ন (initialisation) অংশের পাশাপাশি হালায়ন (update) অংশও ফাঁকা রাখা যায়। নীচের ক্রমলেখ (program) দেখো। এখান আমরা দুটো চলক (variable) gunti আর jogfol নিয়েছি। দুটোরই আদি মান (initial value) শুন্য, কারণ এখন একটা সংখ্যাও যোগান (input) নেয়া হয় নি, আর তাই যোগফলও এই অবস্থায় শুন্য। চলক gunti কে তুমি চাইলে অবশ্য আদ্যায়ন (initialisation) অংশেও ঘোষণা করে আদি মান দিতে পারতে যেমন for (int gunti = 0; gunti < 10;) কিন্তু চলক jogfol কে অবশ্যই ঘূর্ণীর (loop) বাইরে ঘোষণা করতে হবে, কারণ আমরা ফলন (output) দেখাবো তো ঘূর্ণীর বাইরে।

```
int gunti = 0, jogfol = 0;

for( ; gunti < 10; )   // ফাঁকা হালায়ন
{
  cout << "nombor koto? ";
  int nombor; cin << mombor;

  if (nombor <= 0)     // ধনাত্মক না হলে
    continue;          // পাক ডিঙাও

  jogfol += nombor;
  ++gunti;             // হালায়ন
}

cout << "jogfol " << jogfol << endl;
```

উপরের ক্রমলেখয়ের জন্য ঘূর্ণীতে (for loop) আমরা কেবল শর্তের (condition) অংশটি রেখেছি, মোট নম্বর ১০ টি হলো কিনা তা পরীক্ষা করতে। ঘূর্ণীর (loop) ভিতরে যোগান যাচনা (input prompt) করে নম্বরটি যোগান (input) নেয়া হয়েছে। তারপর nombor চলকের মান যদি শুন্য বা কম হয় তাহলে continue; দিয়ে পাক ডিঙাতে বলা হয়েছে। ধনাত্মক সংখ্যা ছাড়া অন্য রকমের সংখ্যা আসলে আমরা উপেক্ষা করতে চাই, এ কারণে এ ব্যবস্থা।

আমরা জানি জন্য ঘূর্ণীতে (for loop) continue; করলে নিয়ন্ত্রণ (control) সরাসরি হালায়ন (update) অংশে চলে যায়। তো আমাদের এই ঘূর্ণীতে হালায়ন অংশতো ফাঁকা রেখেছি, কাজেই gunti চলকের মান যা ছিলো তাই থাকলো। ফলে আবার যোগান যাচনা করে নম্বর যোগান নেওয়া হবে। যতক্ষণ শুন্য বা তার কম কোন সংখ্যা যোগান (input) দেয়া হচ্ছে ততক্ষণ এইভাবে চলতে থাকবে gunti চলকের মান বাড়বে না। এবার ধরো ধনাত্মক নম্বরটি যোগান দেয়া হলো, তাহলে nombor <= 0 এই শর্তটি মিথ্যা হবে, ফলে নিয়ন্ত্রণ (control) আর পাক ডিঙাবে না, পরের সারিতে গিয়ে নম্বরটিকে jogfol এর সাথে যোগ করবে, আর gunti চলকের (variable) মানও এক বাড়বে। এই ভাবে দশটি ধনাত্মক সংখ্যা হলেই কেবল gunti চলকের মান বেড়ে দশ হওয়া সম্ভর আর তাতে ঘূর্ণী (loop) থেকে বের হয়ে যাওয়া সম্ভর, ঋনাত্মক সংখ্যা বা শুন্য দিয়ে

শর্ত মিথ্যা করা সম্ভব হবে না। তাহলে আমরা দেখলাম ঘূর্ণীর (loop) হালায়ন অংশকে ফাঁকা রেখে শর্ত সাপেক্ষে হালায়ন করতে চাইলে সেটা আমরা বিবৃতি (statement) অংশে নিতে পারি।

তবে একটা ব্যাপার এখানে স্মরণ করিয়ে দেয়া দরকার। ধরো তর্কে খাতিরে দুষ্টামি করে আমরা কখনোই ধনাত্মক সংখ্যা যোগান (input) না দিয়ে কেবলই শুন্য বা ঋণাত্মক সংখ্যা যোগান দিতে থাকলাম। এই অবস্থায় কী ঘটবে? তাহলে তো gunti চলকের (variable) মান কখনো বাড়বে না, ফলে gunti < 10 শর্তটি মিথ্যা হওয়ার কোন সম্ভাবনা থাকছে না। এই অবস্থায় কিন্তু জন্য ঘূর্ণীটি (for loop) অসীম সংখ্যক বার ঘুরতে থাকবে। যে ঘূর্ণী (loop) অসীম সংখ্যক বার ঘুরে, আর ঘূর্ণী থেকে বের হওয়ার কোন সুযোগ নাই, এ রকম ঘূর্ণীকে বলা হয় অসীম ঘূর্ণী (infinite loop)। অসীম ঘূর্ণী তৈরী হওয়া মানে এই ক্রমলেখ কোন দিনই থামবে না। ঘূর্ণী (loop) তৈরী করলেই আমাদের তাই সতর্ক থাকতে হয় যাতে সেটা কোন ভাবেই অসীম ঘূর্ণী না হয়ে যায়।

যদিও এই ক্রমলেখতে (program) দশটি ধনাত্মক সংখ্যা যোগান (input) দিয়ে দিয়ে ঘূর্ণী থেকে বের হওয়ার আমাদের সুযোগ আছে, তবে সেটা কেবল সম্ভব যদি যোগান দাতার (user who is giving the input) সদিচ্ছা থাকে আর আমরা তার ওপরে আস্থা রাখতে পারি। তুমি যদি যোগান দাতার ওপরে আস্থাশীল না হও তাহলে একটা কাজ করতে পারো। সেটা হলো সর্বোচ্চ কত বার তুমি যোগান (input) চাইবে সেটা নির্দিষ্ট করে দিতে পারো। যেমন ১০ টি সংখ্যা যোগান নেয়ার জন্য ধরো আমরা ধরে নিলাম যে সর্বোচ্চ ১৫ বার যোগান দেয়া আমরা মেনে নেবো। নিতান্ত যদি ভুল করে ঋণাত্মক বা শুন্য কেউ দেয়, সেই রকম ভুল আমরা এক্ষেত্রে ৫ বারের বেশী হতে দিবো না। তাহলে আমরা নীচের মতো করে ক্রমলেখ (program) লিখতে পারি।

```
int chesta = 0, gunti = 0, jogfol = 0;

for( ; chesta < 15 && gunti < 10; )
{
  cout << "nombor koto? ";
  int nombor; cin << mombor;

  ++chesta;

  if (nombor <= 0)      // ধনাত্মক না হলে
    continue;           // পাক ডিঙাও

  jogfol += nombor;
  ++gunti;              // হালায়ন
}

if (gunti == 10)
  cout << "jogfol " << jogfol << endl;
else
  cout << "sorboccho chesta shesh" << endl;
```

উপরের এই ক্রমলেখতে আমরা chesta নামের আরেকটি চলক নিয়েছি যেটি দিয়ে সর্বমোট কয়বার যোগান (input) দেয়া হলো সেটা হিসাব রাখবো। প্রতিবার নম্বর যোগান দেওয়া মাত্রই chesta চলকের (variable) মান এক বাড়বে, নম্বরটি ধনাত্মক, ঋণাত্মক, শুন্য যাই হোক, এ কারণে এটি কিন্তু if (nombor <= 0) continue; এর আগে দেয়া হয়েছে। আর chesta

চলকের মান যাতে ১৫ হওয়া পর্যন্ত ঘূর্ণী ঘুরে তাই আমরা এবার ঘূর্ণীর (loop) শর্ত অংশটি বদলে লিখেছি chesta < 15 && gunti < 10। এর মানে হলো যে কোন একটি শর্ত ভঙ্গ হলেই ঘূর্ণী আর ঘুরবে না, কাজেই ১৫ বারের বেশী চেষ্টা করা সম্ভব হবে না, আবার ১০ টির বেশী ধনাত্মক নম্বরও যোগান (input) দেয়া সম্ভব হবে না। তাহলে একটা ব্যাপার আমরা দেখলাম, ঘূর্ণীর শর্ত (condition) অংশে আমরা চাইলে একাধিক শর্ত (condition) বুলক সংযোজক (boolean connectives) যেমন এবং &&, অথবা ||, নয় ! দিয়ে সংযুক্ত করে দিতে পারি। সবশেষে দেখো ঘূর্ণীর (loop) বাইরে আমরা যোগফল ফলন (output) দিয়েছি যদি gunti চলকের মান ১০ হয়ে থাকে। আর না হয়ে থাকলে মানে ১৫ বারের চেষ্টায়ও ১০ টি ধনাত্মক সংখ্যা নেয়া সম্ভব হয় নাই সেক্ষেত্রে একটা বার্তা (message) দেখানো হয়েছে যে সর্বোচ্চ চেষ্টা শেষ হয়ে গেছে।

৮.৯ জন্য ঘূর্ণীতে ফাঁকা বিবৃতি (For Loop Empty Statement)

এমন একটি ক্রমলেখ (program) লিখো যেটি শুন্য থেকে প্রতি পাকে ৩ করে বাড়িয়ে সর্বোচ্চ ১০ ধাপ সামনে যাবে, আর এই ভাবে যদি এমন কোন সংখ্যা পায় যেটি ৭ দ্বারা বিভাজ্য তাহলে থেমে যাবে। আমরা এই রূপে পাওয়া সর্বশেষ সংখ্যাটি ফলনে জানতে চাই।

```
int nombor = 0; // গুনতি দিয়ে ঘূর্ণী
for(int gunti = 0; gunti < 10; ++gunti)
{
  if (nombor % 7 == 0)  // নম্বর দিয়ে ক্ষান্তি
    break;
  nombor += 3;
}
cout << nombor << endl;
```

ক্ষান্তি (break) ব্যবহার করে এই ক্রমলেখটি লেখা বেশ সহজ। ঘূর্ণী (loop) gunti চলকের (variable) মান শুন্য থেকে দশের কম পর্যন্ত চালাও আর প্রতি পাকে nombor চলকের মান ৭ দ্বারা বিভাজ্য হলো কিনা পরীক্ষা করে দেখো। ৭ দ্বারা বিভাজ্য হলে ঘূর্ণীতে ক্ষান্তি (break) দাও। আর না হলে nomborএর মান ৩ বাড়াও। তুমি কিন্তু চাইলে এই একই কাজ নীচের মতো করেও করতে পারো যেখানে আমরা ঘূর্ণী (loop) চালিয়েছি nombor চলক ব্যবহার করে। লক্ষ্য করো nombor চলকের মান শুন্য থেকে শুরু করে প্রতিবারে ৩ করে বাড়বে, আর ঘূর্ণী চলবে যতক্ষণ nombor ৭ দ্বারা বিভাজ্য নয় ততক্ষণ, ৭ দ্বারা বিভাজ্য হওয়া মাত্র ঘূর্ণী শেষ হয়ে যাবে। বিবৃতি অংশে দেখো আমরা gunti চলকের মান ১০ বা বেশী হলে ঘূর্ণীতে ক্ষান্তি (break) দিয়েছি, আর না হলে gunti চলকের মান এক বাড়বে।

```
int nombor, gunti = 0;  // নম্বর দিয়ে ঘূর্ণী
for(nombor = 0;  nombor % 7 != 0; nombor += 3)
{
  if (gunti >= 10)       // গুনতি দিয়ে ক্ষান্তি
    break;
  ++gunti;
}
cout << nombor << endl;
```

উপরের দুটি ক্রমলেখতে আমরা একবার একটা চলককে (variable) ঘূর্ণীতে (loop) আর অন্য চলকটিকে ক্ষান্তি (break) এ ব্যবহার করেছি, আর আরেকবার ঠিক উল্টোটা করেছি। আমরা কি চাইলে উভয় চলককে ঘূর্ণীতে ব্যবহার করতে পারি না। অবশ্যই পারি। নীচের ক্রমলেখ (program) খেয়াল করো। এখনো nombor ও gunti দুটো চলককেই ঘূর্ণীতে ব্যবহার করছি। আদ্যায়ন (initialisation) অংশে দুটোর মানই শুরু হয়েছে শুন্য থেকে, আমরা বির্তি (comma) , ব্যবহার করেছি এখানে। শর্ত (condition) অংশে আমরা দুটো শর্ত দিয়েছি এবং && দিয়ে জোড়া দিয়ে যাতে ঘূর্ণী ততক্ষণ চলে যতক্ষণ উভয় শর্ত সত্য হয়। যে কোন একটি শর্ত মিথ্যা হলেই ঘূর্ণী শেষ হয়ে যাবে। আর শর্ত দুটি হলো gunti < 10 ও nombor % 7 == 0, প্রথম শর্তটি সর্বোচ্চ দশ ধাপের জন্য আর দ্বিতীয় শর্তটি ৭ দ্বারা বিভাজ্য না হওয়া পর্যন্ত। আর ঘূর্ণীর (loop) হালায়ন (update) অংশে দেখো আমরা gunti আর nombor দুটোকেই বাড়িয়েছি, একটাকে এক করে, আরেকটাকে তিন করে। তো এ সবের ফলে আমাদের বিবৃতি (statement) অংশে কিন্তু আর কিছুই করার থাকছে না। আমরা তাই বিবৃতি অংশে একটা ফাঁকা বিবৃতি (empty statement) দিয়েছি কেবল একটা দির্তি (semicolon) ব্যবহার করে। তুমি চাইলে {} ফাঁকা মহল্লা (block) দিয়েও সেটা করতে পারো।

```
int nombor, gunti;
for(gunti = 0, nombor = 0;            // আদ্যায়ন
    gunti < 10 && nombor % 7 != 0;    // শর্ত
    ++gunti, nombor += 3)             // হালায়ন
  ;   //  ফাঁকা  বিবৃতি
cout << nombor << endl;
```

তাহলে আমরা দেখলাম, জন্য ঘূর্ণীর (for loop) বিবৃতিও (statement) ফাঁকা থাকতে পারে। আর আদ্যায়ন (initialisation) ও হালায়ন (update) অংশে একাধিক চলক ব্যবহার করা যাবে, উপরোন্ত বৃদ্ধি বা হ্রাস যে কেবল এক করে করতে হবে তাও না, বরং অন্য যে কোন পরিমানে করা যাবে। এছাড়া শর্ত (condition) অংশেও দরকার মতো একাধিক শর্ত বুলক সংযোজক (boolean connectives) যেমন এবং && অথবা || দিয়ে জোড়া দেওয়া যাবে।

৮.১০ বিবৃতি হালায়ন মিথস্ক্রিয়া (Statement and Update)

সাধারণত জন্য ঘূর্ণীর (for loop) চলকটির (variable) মান আমরা বিবৃতিতে (statement) অথবা হালায়নে (update) পরিবর্তন করি। কিন্তু উভয় অংশে পরিবর্তন করলে অবস্থা কী দাঁড়ায়?

```
for(int gunti = 0; gunti < 10; ++gunti)
{
  cout << gunti << " ";
  if (gunti == 5)
    ++gunti;
}
cout << endl;
```

উপরের ক্রমলেখতে (program) আমরা জন্য ঘূর্ণীর (for loop) চলক gunti এর মান বিবৃতি (statement) ও হালায়ন (update) উভয়খানে পরিবর্তন করেছি। হালায়নে নিয়মিত

পরিরর্তন হিসাবে এক করে বাড়বে প্রতিবার, আর বিবৃতিতে যদি চলতি মান (current value) হয় তাহলে এক বাড়বে। এর ফলে কী হবে? খেয়াল করো gunti এর মান ৫ ছাড়া অন্য কিছু হলে কেবল হালায়ন অংশে এক বাড়বে, কিন্তু gunti এর মান ৫ হলে, তখন বিবৃতিতে if (gunti == 5) শর্ত সত্য হওয়ায় সেখানে gunti এর মান এক বাড়বে, আবার হালায়নে তো আরো এক বাড়বেই। ফলে পরের পাকে যখন gunti চলকের মান ফলনে (output) আসবে তখন সেটা ৬ না হয়ে ৭ হবে। কাজেই নীচে যেমন দেখানো হলো আমরা ফলনে ৬ দেখতে পাবো না।

```
0 1 2 3 4 5 7 8 9
```

জন্য ঘূর্ণীর (for loop) হালায়ন (update) অংশে বৃদ্ধিটা ওরকমই থাকুক, কিন্তু বিবৃতি (statement) অংশে আমরা বৃদ্ধি না করে যদি হ্রাস (decrement) করি, তাহলে অবস্থা কী দাঁড়াবে? ধরা যাক নীচের ক্রমলেখয়ের (program) মতো বিবৃতিতে gunti চলকের মান ৫ হলে এক না বাড়িয়ে আমরা বরং এক কমালাম। খেয়াল করো এই ক্ষেত্রে gunti চলকের মান ৫ হওয়ার পরে যদির শর্ত সত্য হওয়ায় এক কমে ৪ হবে, কিন্তু হালায়নে গিয়ে আবার এক বেড়ে হয়ে যাবে ৫, ফলনে (output) ৫ দেখানোর পরে আবার যদির শর্ত সত্য হওয়ায় এক কমে হবে ৪, হালায়নে গিয়ে হবে ৫, ফলে আবার ফলনে ৫ দেখাবে। কাজেই 0 1 2 3 4 5 করে একবার ৫ দেখানোর পরে এই জন্য ঘূর্ণী (for loop) তারপরের প্রতি পাকেই কেবল ৫ই দেখিয়ে যাবে। আর এই ঘূর্ণী-তে (loop) বের হওয়ার কোন উপায় নাই। কাজেই এটি একটি অসীম ঘূর্ণীতে (infinite loop) পরিণত হবে। তুমি জানো আমরা সবসময় চাই অসীম ঘূর্ণী এড়িয়ে যেতে।

```
for(int gunti = 0; gunti < 10; ++gunti)
{
  cout << gunti << " ";
  if (gunti == 5)
    --gunti;
}
cout << endl;
```

৮.১১ অদরকারী জন্য ঘূর্ণী (Unnecessary For Loop)

এমন একটি ক্রমলেখ (program) রচনা করো যেটি ০ থেকে ৯ পর্যন্ত প্রতিটি অঙ্ককে কথায় লিখ-বে। আর অঙ্কগুলোকে পরপর অঙ্কেই লিখলে যে ক্রমলেখ হতো তার সাথে তফাৎটা কেমন দাঁড়ায়?

```
0 1 2 3 4 5 6 7 8 9
```

অঙ্কগুলোকে উপরের মতো করে পরপর অঙ্কেই লিখলে ব্যাপারটা তো খুবই সহজ। এই রকম ক্রমলেখ (program) আমরা নীচে দেখালাম। একটা জন্য ঘূর্ণী (for loop) চলক (variable) onkoএর মান ০ থেকে ৯ পর্যন্ত এক করে বাড়িয়ে যাবে, আর প্রতি পাকে আমরা cout ব্যবহার করে onkoটাকে দেখাবো। এখানে cout জানে কোন অঙ্ককে কীভাবে লিখতে হয়!

```
for(int onko = 0; onko <= 9; ++okno)
  cout << onko << " ";
cout << endl;
```

৮.১১. অদরকারী জন্য ঘূর্ণী (Unnecessary For Loop)

এবার অঙ্কগুলোকে যদি অঙ্কে না লিখে আমরা কথায় লিখি, তাহলে কী করবো? এখানে যেহেতু আমরা ০ থেকে ৯ পর্যন্ত অনেকবার কথায় লিখছি, অনেকে তাই এখানে জন্য ঘূর্ণী (for loop) ব্যবহার করতে উদ্বুদ্ধ হয়। ফলে তারা নীচের মতো করে ক্রমলেখ তৈরী করে। এখানে cout কিন্তু কোন অঙ্ককে কী ভাবে লিখতে হবে তা জানেনা, আমাদের তাই একটি পল্টি ব্যাপার (switch case) ব্যবহার করে প্রতিটি অঙ্ককে আলাদা আলাদা করে বলে দিতে হয়েছে।

```
for(int onko = 0; onko <= 9; ++okno)
{
  swtich(onko)
  {
    case 0: cout << "shunyo "; break;
    case 1: cout << "ek "; break;
    case 2: cout << "dui "; break;
    case 3: cout << "tin "; break;
    case 4: cout << "char "; break;
    case 5: cout << "panch "; break;
    case 6: cout << "soy "; break;
    case 7: cout << "shat "; break;
    case 8: cout << "aat "; break;
    case 9: cout << "noy "; break;
  }
}
cout << endl;
```

কিন্তু উপরের ক্রমলেখতে (program) জন্য ঘূর্ণী (for loop) ব্যবহার আসলে অদরকারী। কারণ ঘূর্ণীর (loop) প্রতি পাকে আসলে পল্টি ব্যাপারের (switch case) আলাদা আলাদা ব্যাপার নির্বাহিত (execute) হচ্ছে। আমাদেরকে তো প্রতিটি পাকের জন্যে সেই আলাদা আলাদা কাজই করতে হলো, তাহলে আর ঘূর্ণী কেন ব্যবহার করবো, কাজগুলো নীচের মতো করে সরাসরি একের পর এক লিখলেই তো হয়ে যায়। মনে রাখবে আমরা ঘূর্ণী তখনই ব্যবহার করবো যখন প্রতি পাকের জন্য এই রকম আলাদা আলাদা কিছু করতে হয় না। যেমন কথায় না দেখিয়ে অঙ্কে দেখালে কিন্তু আমাদের জন্য ঘূর্ণী ব্যবহার করলেই চলতো কারণ cout ব্যবহারের কারণে প্রতিটি অঙ্ক দেখানোতে যে ভিন্নতা সেটা আমাদের সরাসরি সামলাতে হয় না।

```
cout << "shunyo ";
cout << "ek ";
cout << "dui ";
cout << "tin ";
cout << "char ";
cout << "panch ";
cout << "soy ";
cout << "shat ";
cout << "aat ";
cout << "noy ";
cout << endl;
```

৮.১২ জন্য ঘূর্ণীর সাধারণ ব্যবহার (General Purpose For Loop)

জন্য ঘূর্ণী (for loop) আমরা এ পর্যন্ত কেবল ক্রম (order), প্রগমণ (progression), ধারা (series) ইত্যাদির ক্ষেত্রে ব্যবহার করেছি। জন্য ঘূর্ণী (for loop) কী এ সব ছাড়া যে কোন শর্ত পরীক্ষার মাধ্যমে সাধারণ ভাবে একটি ঘূর্ণী তৈরীতে ব্যবহার করা যায়?

সিপিপি ছাড়া অন্যান্য অনেক ভাষায় জন্য ঘূর্ণী (for loop) কেবল ক্রম (order), প্রগমণ (progression), ধারা (series) এসবেই ব্যবহার করা যায়। তবে সিপিপির জন্য ঘূর্ণী আসলে অনেক শক্তিশালী, এটাকে যে কোন রকম ঘূর্ণী তৈরীতে ব্যবহার করা যায়। বস্তুত সিপিপিতে জন্য ঘূর্ণী (for loop) দিয়েই সকল রকমের ঘূর্ণী তৈরী করা যায়, কাজেই অন্য কোন ঘূর্ণী দরকার হয় না, যদিও সিপিপিতে আরো দুটি ঘূর্ণী (loop) আছে, যে গুলো আমরা পরের পাঠগুলোতে দেখবো।

```
int nombor = 1, gunti = 0, jogfol = 0;

for( ; nombor != 0; )    // আদ্যায়ন হালায়ন ফাঁকা
{
     cin >> nombor;      // নম্বর যোগান নাও
     if (nombor)         // নম্বর শুন্য না হলে
     {
       gunti += 1;       // গুনতি এক বাড়বে
       jogfol += nombor; // যোগফলেও যোগ হবে
     }
}

cout << "gunti " << gunti << " ";
cout << "jogfol " << jogfol << endl;
```

উপরের ক্রমলেখ (program) খেয়াল করো। এখানে আমরা কিছু নম্বর যোগান (input) নিয়ে তাদের যোগফল বের করতে চাই। তবে কয়টি নম্বর যোগান নিবো আমরা সেটা আগে থেকে জানিনা । ব্যবহারকারী যতগুলো ইচ্ছে নম্বর যোগান দিতে থাকবে, যখন সে আর কোন নম্বর যোগান দিতে চায় না তখন সে একটা শুন্য যোগান দিয়ে সেটা জানাবে। আমরা তারপর কয়টি নম্বর যোগান নিয়েছি আর তাদের যোগফল কত সেটা ফলনে (output) দেখাবো।

তো এই ক্রমলেখটি (program) খুবই সহজ। আমরা একটি জন্য ঘূর্ণী (for loop) নিয়েছি, কিন্তু এটাকে আমরা কোন চলকের (variable) মান বাড়িয়ে বা কমিয়ে ঘুরাবো না। আমরা মূলত শর্তটা ব্যবহার করবো ঘূর্ণী (loop) তৈরীতে। তিনটি চলক (variable) nombor, gunti আর jogfol নেয়া হয়েছে যাদের আদিমান (initial value) দেওয়া হয়েছে ঘূর্ণীর বাইরে। ঘূর্ণীর পরে যেহেতু gunti ও jogfol ফলনে (output) দেখানো হবে, তাই ওগুলো অবশ্যই ঘূর্ণীর (loop) বাইরে ঘোষণা (declare) করতে হবে। কিন্তু nombor চলকটি ঘূর্ণীর ভিতরে ঘোষণা ও আদ্যায়ন (initialisation) করা যেতে পারতো। যাইহোক ঘূর্ণীর ভিতরে নম্বরটি যোগান নিয়ে যদি শুন্য না হয় তাহলে গুনতি এক বাড়িয়ে যোগফলের সাথে নম্বরটি যোগ করা হয়েছে।

একটা বিষয় খেয়াল করো আমাদের ঘূর্ণীতে (loop) শর্ত nombor != 0 অর্থাৎ nombor এর মান শুন্য ছাড়া অন্য কিছু হলে কেবল ঘূর্ণীর বিবৃতি (statement) নির্বাহিত (execute) হবে। তো প্রথমবার আমরা তো অবশ্যই ঘূর্ণীর ভিতরে ঢুকতে চাই, কিন্তু nombor তো তখন পর্যন্ত একটাও যোগান (input) নেয়া হয় নাই। ঘূর্ণীর ভিতরে যদি আমাদের ঢুকতেই হয়, আমাদের

সেক্ষেত্রে কোন ভাবে শর্ত সত্য করে দিতে হবে, nombor চলকরে আদিমান শুন্য ছাড়া একটা কিছু দিয়ে রাখতে হবে। আমরা nombor এর আদি মান দিয়েছি 1, তুমি চাইলে শুন্য ছাড়া অন্য যে কোন কিছু দিতে পারতে। চলক gunti আর jogfol এর আদিমান তো শুন্যই দিতে হবে, সেটা বুঝতেই পারছো, যেহেতু তখনও আমাদের একটাও নম্বর যোগান নেওয়া হয় নাই।

আসলে জন্য ঘূর্ণীতে (for loop) শর্ত পরীক্ষণ হয় সাধারণত বিবৃতিতে (statement) ঢুকার আগে, অথচ আমরা এখানে শর্ত পরীক্ষা করতে চাই বিবৃতি অংশের পরে, কারণ বিবৃতিতে আমরা যে নম্বরটি যোগান নিবো সেটা আমরা পরীক্ষা করতে চাই পরের পাকে ঢুকার আগে। পরের পাকের আগে পরীক্ষণ মানে আগের পাকের পরে আর কী! আর সে কারণে জোর করে nombor চলকের আদি মান 1 দিয়ে প্রথমবার শর্ত সত্য বানিয়ে আমরা খানিকটা চালাকি করেছি!

৮.১৩ জন্য ঘূর্ণীর নানান বাহার (For Loop Variations)

জন্য ঘূর্ণী (for loop) কত ভাবে লেখা যায়? এর মধ্যে নিশ্চয় বুঝে ফেলেছো for (;;) জন্য ঘূর্ণীর বন্ধনীর (brackets) ভিতরের দির্তি (semicolon) ; দুটো কেবল আবশ্যিক। তাহলে আদ্যায়ন (initialisation), শর্ত (condition), হালায়ন (update) কত ভাবে বিন্যাস করা সম্ভব?

```
for (আদ্যায়ন; শর্ত; হালায়ন)          আদ্যায়ন
  বিবৃতি                                 for ( ; শর্ত; হালায়ন)
                                           বিবৃতি
```

উপরে বামপাশে জন্য ঘূর্ণীর (for loop) সাধারণ অবস্থা দেখানো হয়েছে, আর ডান পাশে দেখানো হয়েছে যে চাইলেই আদ্যায়ন (initialisation) অংশটি ঘূর্ণীর (loop) বাইরে নিয়ে যা-ওয়া যায়। তাতে ফলাফল একই থাকবে। তবে তফাৎ অল্প একটুই আছে সেটা হলো বামপাশের আদ্যায়নে যদি চলক ঘোষিত হয় সেটি কেবল ঘূর্ণীর জন্য স্থানীয় চলক (local variable), তাই ঘূর্ণীর পরে আর ব্যবহার করা যায় না। কিন্তু ডানপাশের আদ্যায়নে যদি চলক ঘোষিত (variable declaration) হয় সেটা ঘূর্ণীর (loop) পরেও ব্যবহার করা যাবে। আমরা এর পরে থেকে আদ্যা-য়ন (initialisation) ঘূর্ণীর (loop) ভিতরেই হয়তো লিখবো, তবে আরেকবার বলেই দিচ্ছি চলক ঘোষণার ব্যাপারটিতে কোন সমস্যা না থাকলে তুমি চাইলেই সেটি ঘূর্ণীর আগেই লিখতে পারবে। সুতরাং আদ্যায়ন কোথায় থাকলো সে বিষয়ে ভিন্নতা এখানের পরে আর দেখাবো না।

```
for (আদ্যায়ন; শর্ত; হালায়ন)          for (আদ্যায়ন; শর্ত; )
  বিবৃতি                                 {
                                           বিবৃতি
                                           হালায়ন
                                         }
```

এরপর উপরে দেখো আমরা কী ভাবে হালায়নের (update) অংশটকু যথাস্থানে না লিখে বি-বৃতি (statement) সাথে দিয়ে দিয়েছি। এতে ঘূর্ণীর (loop) ফলাফল একই থাকবে, কারণ বি-বৃতির পরপরই তো হালায়ন নির্বাহিত (execute) হতো, এখনো তাই হচ্ছে। কাজেই তুমি চাইলে যে কোন সময় এইটা করতে পারো। আমরা এরপরে হালায়ন (update) যথাস্থানেই রাখবো, এ সংক্রান্ত ভেদনগুলো (variation) আর দেখাবো না।

```
for (আদ্যায়ন; ; হালায়ন)          for (আদ্যায়ন; true; হালায়ন)
{                                    {
```

```
    if (!শর্ত) break;
    বিবৃতি
  }
```

```
    if (!শর্ত) break;
    বিবৃতি
  }
```

এবার আমরা শর্ত (condition) অংশের ভিন্নতা দেখবো। উপরে দেখো আমরা শর্তটিকে বিবৃতির (statement) অংশে নিয়ে গিয়েছি। ফলে শর্ত অংশে বাম পাশের মতো হয় ফাঁকা রাখা হবে, না হয় ডানপাশের মতো true লিখে দেওয়া হবে। দুটো মূলত একই কথা কারণ ফাঁকা শর্ত মানে সত্য। শর্ত যখন বিবৃতি অংশে গেছে তখন দেখো আমরা ! লাগিয়ে উল্টো শর্ত দিয়ে ঘূর্ণীতে (loop) ক্ষান্তি (break) দিয়েছি যাতে শর্ত মিথ্যা হলেই ঘূর্ণী থেকে নিয়ন্ত্রণ (control) বের হয়ে যায়। সুতরাং এই ভেদনগুলো (variation) জন্য ঘূর্ণীর (for loop) সাধারণ অবস্থার সমার্থক।

```
for (আদ্যায়ন; শর্ত১; হালায়ন)
{
  if (শর্ত২)
    বিবৃতি
}

for (আদ্যায়ন; ; হালায়ন)
{
  if (!শর্ত১) break;
  if (শর্ত২) বিবৃতি
}
```

```
for (আদ্যায়ন; শর্ত১; হালায়ন)
{
   if (!শর্ত২) continue;
   বিবৃতি
}

for (আদ্যায়ন; ; হালায়ন)
{
 if (!শর্ত১) break;
   if (!শর্ত২) continue;
  বিবৃতি
}
```

উপরে দেখো বিবৃতি অংশে আমরা আরেকটি শর্ত ব্যবহার করে বিবৃতি নির্বাহ করেছি। তো আমরা চাইলে এখানে উপরের ডান পাশের মতো করে শর্ত২ এর বিপরীত শর্ত দিয়ে পাক ডিঙাতে (continue) পারি। তাতে বিবৃতি আর যদি (if) এর অধীনে থাকছে না। উদাহরণগুলোর শর্ত১ কে যদি আমরা বিবৃতি অংশে নামিয়ে দেই তাহলে আমরা যা পাবো তাও উপরে দেখানো হয়েছে।

```
for (আদ্যায়ন; ; )
{
  if (!শর্ত১) break;
  if (শর্ত২) বিবৃতি
  হালায়ন
}
```

```
for (আদ্যায়ন; ; )
{
 if (!শর্ত১) break;
   if (!শর্ত২)
   {
     হালায়ন
     continue;
   }
   বিবৃতি
   হালায়ন
}
```

সবশেষে আমরা উপরে একটু দেখি হালায়নটুকু (update) যদি বিবৃতি (statement) অংশে ঢুকে যায় তাহলে ক্ষান্তি (break) আর ডিঙানোর (continue) সাথে কী মিথস্ক্রিয়া ঘটে। বিশেষ করে ডানপাশে দেখো শর্ত২ সত্য না হলে আমাদের প্রথমে হালায়নের (update) কাজটুকু করতে হবে, তারপর আমরা ডিঙাতে (continue) পারবো। এর কারণ for(আদ্যায়ন;;) এখানে যেহেতু

হালায়ন অংশটুকু ফাঁকা আর continue করলে স্বাভাবিক ভাবে জন্য ঘূর্ণীতে (for loop) হালায়ন অংশটুকুতেই নিয়ন্ত্রণ (control) চলে যায়, সেহেতু হালায়নের কাজটুকু আমাদের continue; এর আগেই সেরে ফেলতে হবে। এ ছাড়া খেয়াল করো হালায়নের কাজটুকু আমরা বিবৃতির পরেও করেছি, সেটাতো স্বাভাবিক ভাবেই হওয়ার কথা।

৮.১৪ পূর্ব শর্তের ক্ষণ ঘূর্ণী (Precondition in While Loop)

সিপিপিতে একটি ক্রমলেখ (program) লিখো যেটি ব্যবহারকারীর (user) কাছে থেকে দুটি ধনাত্মক পূর্ণক (positive integer) যোগান (input) নিয়ে তাদের গরিষ্ঠ সাধারণ গুণনীয়ক বা গসাগু নির্ণয় করবে। এই ক্রমলেখটি তুমি ক্ষণ ঘূর্ণী (while loop) ব্যবহার করে লিখবে।

ফিরিস্তি ৮.৭: দুটি সংখ্যার গসাগু (HCF of Two Numbers)

```
int purnok1, purnok2;    // চলক ঘোষণা
cout << "duto dhonatok purnork: ";  // যোগান যাচনা
cin >> purnok1 >> purnok2;     // যোগান নেয়া

if (purnok1 <= 0 || purnok2 <= 0) // ধনাত্মক না হলে
{
  cout << "shunyo ba rinatok purnok!" << endl;
  return EXIT_FAILURE;
}

// গসাগু নির্ণয়ের মুল অংশ
int vagshesh = purnok1 % purnok2; // ভাগশেষ নির্ণয়
while(vagshesh) // ভাগশেষ শুন্য না হলে
{
  purnok1 = purnok2;     // ভাজকই হবে নতুন ভাজ্য
  purnok2 = vagshesh;    // ভাগশেষ হবে নতুন ভাজক
  vagshesh = purnok1 % purnok2; // আবার ভাগশেষ
}

cout << "gosagu: " << purnok2 << endl;  // ভাজকই গসাগু

return EXIT_SUCCESS;
```

উপরে দেখানো ক্রমলেখ (program) খেয়াল করো। আমরা প্রথমে চলক ঘোষণা (variable declare) করে যোগান যাচনা (input prompt) করে পূর্ণক (integer) দুটি যোগান (input) নিয়েছি। এরপর আমরা যদি (if) লাগিয়ে শর্ত পরীক্ষা করেছি, দেখেছি পূর্ণকদুটির যে কোনটি শুন্য বা তার কম কিনা। কারণ শুন্য বা ঋণাত্মক সংখ্যার জন্য আমরা গসাগু নির্ণয় করবো না, সেক্ষেত্রে বরং আমরা ত্রুটি বার্তা (error message) দেখিয়ে ক্রমলেখ (program) ব্যর্থতার (failure) সাথে শেষ করবো। এরপরে রয়েছে ঘূর্ণী (loop) দিয়ে আমাদের গসাগু নির্ণয়ের মুল অংশটুকু।

গসাগু নির্ণয় করতে গেলে আমাদের প্রথমে প্রদত্ত সংখ্যা দুটির একটিকে দিয়ে আরেকটিকে ভাগকরে ভাগশেষ বের করতে হয়। ভাগশেষ শুন্য হওয়া মানে আমাদের আর ভাগ করতে হবে

না, আর সেক্ষেত্রে ভাজক যে পূর্ণকটি সেটিই হলো আমাদের গসাগু। তো খেয়াল করো আমরা কিন্তু purnok1 কে purnok2 দিয়ে ভাগ করে ভাগশেষ নিয়েছি vagshesh চলকে। এক্ষেত্রে আমরা সিপিপির % অনুক্রিয়া ব্যবহার করে ভাগশেষ নির্ণয় করেছি। আমরা এখানে একটি নতুন ধরণের ঘূর্ণী (loop) ব্যবহার করেছি যেটি হলো ক্ষণ ঘূর্ণী (while loop)। এই ক্ষণ ঘূর্ণীর শর্ত দেয়া হয়েছে vagshesh অর্থাৎ ভাগশেষ যতক্ষণ সত্য (বা শুন্য নয়) ততক্ষণ ঘূর্ণী (loop) চলবে। তুমি কিন্তু চাইলে শর্ত হিসাবে (vagshesh) না লিখে (vagshesh != 0) লিখতে পারতে, সেটা একই কথা হতো। যাইহোক শর্তটি সত্য না হলে অর্থাৎ vagshesh শুন্য হলে ঘূর্ণীর বাইরে মহল্লার (block) {} পরে দেখো আমরা purnok2কে গসাগু হিসাবে ফলন দেখিয়েছি।

এখন কথা হচ্ছে ভাগশেষ শুন্য না হলে আমাদের কী করতে হবে? সেটা আমরা মহল্লার (block) ভিতরে লিখেছি। আমাদের আগের ভাজকটি হবে নতুন ভাজ্য, তাই আমরা প্রথমে লিখেছি purnok1 = purnok2;, এতে কিন্তু আগের ভাজ্যটি হারিয়ে গেলো, আমাদের আসলে সেটি আর দরকার নাই। তারপর দেখো purnok2 = vagshesh; লিখে আমরা ভাগশেষটিকে নতুন ভাজক হিসাবে নিয়ে নিলাম। তুমি নিশ্চয় বুঝতে পারছো কেন purnok1 = purnok2; আগে আর purnok2 = vagshesh; পরে লিখতে হয়েছে। যদি উল্টোটা করা হতো তাহলে কিন্তু purnok2 = vagshesh; এর কারণে purnok2 যেটি কিনা আমাদের নতুন ভাজ্য হবে সেটির মান হারিয়ে যেতো, ফলে ঠিক পরপরই purnok1 = purnok2; করলে আমরা যে ভাজ্যটি পেতাম সেটা আসলে vagsheshএরই মান। নতুন ভাজ্য ও নতুন ভাজক ঠিক করার পরে দেখো এবার আমরা আবার ভাগশেষ নির্ণয় করেছি vagshesh = purnok1 % purnok2 লিখে। এই ভাগশেষটি শুন্য হলে আমাদের ঘূর্ণী (loop) থেকে বের হয়ে যেতে হবে, আর শুন্য না হলে আবারও ঘূর্ণীর (loop) মহল্লার (block) ভিতরে যা আছে তা করতে হবে। তো এই হলো আমাদের ক্রমলেখ।

```
for(int vagshesh = purnok1 % purnok2; // আদ্যায়ন
    vagshesh != 0;                    // শর্ত
    vagshesh = purnok1 % purnok2)     // হালায়ন
{
  purnok1 = purnok2;     // ভাজকই হবে নতুন ভাজ্য
  purnok2 = vagshesh;    // ভাগশেষ হবে নতুন ভাজক
}
```

তুমি হয়তো এবার প্রশ্ন করতে পারো, আচ্ছা আমরা কি এই ক্রমলেখ জন্য ঘূর্ণী (for loop) দিয়ে লিখতে পারতাম। কেন নয়? উপরে দেখো আমরা একই ক্রমলেখ জন্য ঘূর্ণী (for loop) ব্যবহার করে লিখেছি, মূলত ক্ষণ ঘূর্ণীর (while loop) বদলে জন্য ঘূর্ণীটা (for loop) এখানে দেখানো হয়েছে। আদ্যায়ন (initialisation) অংশে আছে প্রথমবার ভাগশেষ নির্ণয়ের ব্যাপারটা, শর্ত অংশে (condition) আছে ভাগশেষ শুন্য না হওয়ার শর্ত যেটা কিনা এখানে vagshesh != 0 লিখা হয়েছে কিন্তু কেবল vagshesh লিখলেও চলতো। আর হালায়ন (update) অংশে আবার ভাগশেষ নির্ণয়ের অংশটুকু দিয়েছি। জন্য ঘূর্ণীর (for loop) নানান বাহার আমরা যে গুলো দেখেছিলাম তুমি সেগুলো এখানেও নিজে নিজে প্রয়োগ করতে পারো।

এবার তাহলে প্রশ্ন করতে পারো, জন্য ঘূর্ণী (for loop) দিয়েই যদি এতো সুন্দর কাজ হয় তাহলে ক্ষণ ঘূর্ণীর (while loop) দরকার কী? সত্যি বলতে আসলে দরকার নাই। সিপিপিতে জন্য ঘূর্ণী (for loop) এতটাই শক্তিশালী যে আর কোন ঘূর্ণী দরকার নাই। তবে জন্য ঘূর্ণীটা তবুও বেশীর ভাগ ক্ষেত্রে ক্রম, প্রগমন, ধারা ইত্যাদির ক্ষেত্রে বেশী ব্যবহার করা হয়, আর ক্ষণ ঘূর্ণী ব্যবহার করা সাধারণ ক্ষেত্রে। ক্ষণ ঘূর্ণীতে (while loop) ঘূর্ণীর অংশ হিসাবে আদ্যায়ন (initialisation) নাই, তাই সেরকম কিছু দরকার হলে ওই অংশটুকুকে রাখতে হবে ঘূর্ণীরও আগে। ক্ষণ ঘূর্ণীতে

(while loop) হালায়ন (update) অংশও আলাদা করে নাই, কাজেই সেটি চলে যাবে বিবৃতির অংশ হিসাবে। এর মানে তোমাকে অবশ্যই বিবৃতির ভিতরে হালায়নের কাজ করে দিতে হবে যাতে ঘূর্ণীটা অসীম ঘূর্ণীতে পরিণত না হয়। একটা বিষয় খেয়াল করো শর্ত প্রথমেই মিথ্যা হলে ক্ষণ ঘূর্ণী (while loop) একবারও না ঘুরতে পারে, এটা অবশ্য জন্য ঘূর্ণীর (for loop) জন্যেও সত্য। এই উভয় ঘূর্ণী প্রথমে শর্ত পরীক্ষা করে, শর্ত সত্য হলে তারপর ঘুরতে যায়। তোমাকে যদি নুন্যতম একবার কাজ করতেই হয় সেটা তাহলে ঘূর্ণীর আগে বা পরে করে ফেলতে হবে। গসাগু নির্ণয়ের ক্ষেত্রে আমাদের যেমন কমপক্ষে একবার ভাগশেষ করতেই হবে, যেটি আমরা ক্ষণ ঘূর্ণীতে (while loop) ঘূর্ণীর আগেই আর জন্য ঘূর্ণীতে (for loop) আদ্যায়নে (initilisation) করে ফেলেছি।

```
for(আদ্যায়ন; শর্ত; হালায়ন)          আদ্যায়ন
    বিবৃতি                           while(শর্ত)
                                        বিবৃতি+হালায়ন
```

৮.১৫ উত্তর শর্তের কর ঘূর্ণী (Post-condition in Do Loops)

দুটো ধনাত্নক পূর্ণ সংখ্যার গসাগু নির্ণয়ের ক্রমলেখটি (program) তুমি আরেকবার লিখো, কিন্তু এবার তুমি ক্ষণ ঘূর্ণী (while loop) বা জন্য ঘূর্ণী (for loop) ব্যবহার না করে তার বদলে ব্যবহার করবে কর ঘূর্ণী (do loop)। কর ঘূর্ণী হলো সিপিপিতে তৃতীয় ও শেষ প্রকারের ঘূর্ণী।

আমরা আগের পাঠেই এই ক্রমলেখটি (program) আলোচনা করেছি ক্ষণ ঘূর্ণী (while loop) ব্যবহার করে, এখানে আগে সেটি আরেকবার একটু দেখে নেই। purnok1, purnok2 চলক দুটো তুমি ঘোষণা করে সেগুলোতে ধনাত্নক সংখ্যা যোগান (input) নিবে। দরকার হলে আগের পাঠ থেকে যোগান (input) নেয়ার ও তারপর পূর্ণক দুটি ধনাত্নক কিনা পরীক্ষা করার ব্যাপারটি দেখে নিতে পারো। তাহলে আমরা এবার গসাগু নির্ণয়ের মুল অংশটায় যেতে পারি।

```
int purnok1, purnok2; // ধনাত্নক মান তুমি যোগান নিবে
int vagshesh = purnok1 % purnok2; // ভাগশেষ নির্ণয়
while(vagshesh) // ভাগশেষ শুন্য না হলে
{
  purnok1 = purnok2;     // ভাজকই হবে নতুন ভাজ্য
  purnok2 = vagshesh;    // ভাগশেষ হবে নতুন ভাজক
  vagshesh = purnok1 % purnok2; // আবার ভাগশেষ
}

cout << "gosagu: " << purnok2 << endl;  // ভাজকই গসাগু
```

উপরের এই ক্রমলেখটির (program) ঘূর্ণীটাকে (loop) যদি আমরা বিস্তার করি, মানে ঘূর্ণী না লিখে প্রত্যেক পাকে যা হতো সেগুলো যদি বার বার লিখি তাহলে কেমন হতো সেটা আমরা নীচে দেখালাম। খেয়াল করো প্রথম ভাগশেষ নির্ণয়টা কিন্তু উপরের ঘূর্ণীর বাইরে ছিলো, আর তারপর নতুন ভাজ্য, নতুন ভাজক, আর আবার ভাগশেষ নির্ণয়ের বিবৃতিগুলো (statement) ছিলো ঘূর্ণীর ভিতরে, তাই ওগুলো নীচের বিস্তারণে বারবার এসেছে।

```
vagshesh = purnok1 % purnok2; // ভাগশেষ নির্ণয়
```

```
purnok1 = purnok2;     // ভাজকই হবে নতুন ভাজ্য
purnok2 = vagshesh;    // ভাগশেষ হবে নতুন ভাজক
vagshesh = purnok1 % purnok2; // আবার ভাগশেষ
purnok1 = purnok2;     // ভাজকই হবে নতুন ভাজ্য
purnok2 = vagshesh;    // ভাগশেষ হবে নতুন ভাজক
vagshesh = purnok1 % purnok2; // আবার ভাগশেষ
purnok1 = purnok2;     // ভাজকই হবে নতুন ভাজ্য
purnok2 = vagshesh;    // ভাগশেষ হবে নতুন ভাজক
vagshesh = purnok1 % purnok2; // আবার ভাগশেষ
.....................................
```

এখন উপরের এই বিস্তারণ দেখে কারো মনে কিন্তু অন্য রকম করে ঘূর্ণী লেখার সাধ জাগতে পারে। কেউ হয়তো বলতে পারে ঘূর্ণীটা (loop) কেন প্রথম ভাগশেষ নির্ণয়কে বাদ দিয়ে শুরু হয়েছে। ঘূর্ণীটাতো বরং প্রথম ভাগশেষ নির্ণয় থেকেই শুরু হতে পারতো। কথা সত্য, আর তাইতো আমাদের নতুন ধরনের একটি ঘূর্ণীর (loop) উদ্ভব হয়েছে, যেটি হলো কর ঘূর্ণী (do loop)।

```
int purnok1, purnok2; // ধনাত্মক মান তুমি যোগান নিবে
int vagshesh;   // এই চলক বাইরেই ঘোষণা করতে হবে!
do
{
  vagshesh = purnok1 % purnok2; // ভাগশেষ নির্ণয়
  purnok1 = purnok2;     // ভাজকই হবে নতুন ভাজ্য
  purnok2 = vagshesh;    // ভাগশেষ হবে নতুন ভাজক
}
while (vagshesh);

// সতর্কতা: এখানে গসাগু কিন্তু purnok1, purnok2 নয়
cout << "gosagu: " << purnok1 << endl;  // ভাজকই গসাগু
```

উপরের ক্রমলেখ (program) খেয়াল করো আমরা গসাগু নির্ণয় করেছি কর ঘূর্ণী (do loop) ব্যবহার করে। এখানে প্রথমবার ভাগশেষ নির্ণয় করা হয়েছে ঘূর্ণীর (loop) ভিতরেই। আর তার-পর নতুন ভাজ্য ও নতুন ভাজক নির্ধারণ করা হয়েছে। ঘূর্ণীর শর্ত পরীক্ষণ তারও পরে while (vagshesh); যেখানে লেখা হয়েছে সেখানে। শর্ত যদি সত্য হয় তাহলে ঘূর্ণীর পরের পাক শুরু হবে, অর্থাৎ নিয়ন্ত্রণ (control) লাফ দিয়ে doএর পরে যে মহল্লা (block) {} শুরু হয়েছে সেখা-নে চলে যাবে। তবে একটা গুরুত্বপূর্ণ বিষয় এখানে উল্লেখ করতে হবে এখানে সেটা হলো ফলনে (output) কিন্তু এখন গসাগু purnok2 হবে না, বরং গসাগু হবে purnok1। এর কারণ হলো মহল্লার ভিতরে vagshesh চলকের মান শুন্য হোক বা না হোক আমরা কিন্তু ভাজকটাকে নতুন ভাজ্য হিসাবে ধরে নিয়েছি, ফলে purnok2 এর মান এখন purnok1এ আছে।

তাহলে ক্ষণ ঘূর্ণী (while loop) আর কর ঘূর্ণীর (do loop) তফাৎ হলো আগেরটিতে শর্ত পরীক্ষা পাকে ঢুকার আগে হয়, শর্ত সত্য হলে পাকে ঢুকে, আর পরেরটিতে শর্ত পরীক্ষা পাক শেষ করে হয়, শর্ত সত্য হলে পরের পাকে ঢুকে। এর মানে কর ঘূর্ণীর (do loop) প্রথম পাক বাদ দিলে ওইটা ক্ষণ ঘূর্ণী (while loop) হয়ে যেতে পারে অথবা উল্টোটা।

৮.১৬ আবার ক্ষান্তি ও ডিঙানো (Break and Continue Again)

এমন একটি ক্রমলেখ (program) তৈরী করো যেটি ব্যবহারকারীকে ক্রমিক নম্বর অনুযায়ী একটি প্রাপণ্য (menu) দেখাবে যোগ, বিয়োগ, গুণ, ভাগফল, বা ভাগশেষ কলন (calculate) করার জন্য। ব্যবহারকারী যত নম্বরের কলন করতে চাইবে তার জন্য দুটি পূর্ণক (integer) যোগান (input) নিয়ে হিসাব করে ফলন (output) দেখাবে। ব্যবহারকারী যতক্ষণ একের পর এক কলন করে যেতে চায় তুমি ততক্ষণ প্রাপণ্য দেখিয়ে, যোগান নিয়ে, কলন করে যাবে।

ফিরিস্তি ৮.৮: অনুন্নত খেলনা কলনি (Rudimentary Toy Calculator)

```
// একটা অসীম ঘূর্ণী তৈরী করবো
while(true) // অথবা for (; ;)
{
  // প্রাপণ্য দেখাও
  cout << "khelna koloni" << endl;
  cout << "0. ber hou" << endl;
  cout << "1. jog + " << endl;
  cout << "2. biyog −" << endl;
  cout << "3. gun ∗" << endl;
  cout << "4. vagfol /" << endl;
  cout << "5. vagshesh %" << endl;

  int posondo;       // পছন্দ চলক
  cin >> posondo;    // যোগান নাও

  if (posondo == 0) // পছন্দ শুন্য হলে
    break;          // ঘূর্ণী থেকে ক্ষান্তি

  // উল্টাপাল্টা পছন্দ হলে পাক ডিঙিয়ে যাও
  if (posondo < 1 || posondo > 5)
    continue;

  // পূর্ণক দুটি যাচনা দিয়ে যোগান নাও
  cout << "purnok duto: ";
  int purnok1, purnok2, folafol;
  cin >> purnok1 >> purnok2;

  // পছন্দ অনুযায়ী ফলাফল কলন করো
  switch(posondo)
  {
    case 1: folafol = purnok1 + purnok2; break;
    case 2: folafol = purnok1 − purnok2; break;
    case 3: folafol = purnok1 ∗ purnok2; break;
    case 4: folafol = purnok1 / purnok2; break;
```

```
      case 5: folafol = purnok1 % purnok2; break;
    }

    // ফলন দেখাও
    cout << "folafol: " << folafol << endl;
  }

  // বিদায় সম্ভাষণ
  cout << "amake abar chalaben" << endl;
```

উপরের ক্রমলেখ (program) দেখো, আমরা একটা অসীম ঘূর্ণী (infinite loop) নিয়েছি while (true) { } লিখে তুমি চাইলে কিন্তু for (; ;) { } লিখে এমন কি do { } while (true); লিখেও অসীম ঘূর্ণী তৈরী করতে পারতে। অসীম ঘূর্ণী তৈরী করলেই আমাদের অবশ্যই ঘূর্ণীর (loop) ভিতরে কোন ভাবে ঘূর্ণী থেকে ক্ষান্তি (break) দেওয়ারও ব্যবস্থা রাখতে হবে।

যাইহোক ঘূর্ণীর (loop) ভিতরে দেখো আমরা প্রথমে কত ক্রমিক নম্বর যোগান (input) দিলে কী করা হবে সেটি দেখিয়েছি, যেখানে ১ হলে যোগ, ২ হলে বিয়োগ, ৩ হলে গুণ, ৪ হলে ভাগ-ফল, ৫ হলে ভাগশেষ, আর ০ হলে বের হয়ে যাওয়া আছে। ব্যবহারকারীর (user) পছন্দ যাচনা (prompt) করে posondo চলকে নেয়া হয়েছে। এবার দেখো posondo চলকের মান ০ হলে নিয়ন্ত্রণ (control) ঘূর্ণী থেকে break দিয়ে ক্ষান্তি দিবে, তাহলে অসীম ঘূর্ণী (infinite loop) আর হচ্ছে না। আর posondo এর মান ০ না হলে তারপর আমরা পরীক্ষা করে দেখেছি সেটি ১ এর কম বা ৫ এর বেশী কিনা। যদি সেরকম হয় তাহলে এইরকম উল্টাপাল্টা পছন্দের জন্য আসলে আমাদের কিছু করার নেই, আমরা কলন করবো কেবল ১ হতে ৫ পর্যন্ত ক্রমিক নম্বরের জন্য। তাহলে এই রকম ক্ষেত্রে আমাদের কলন করা বাদ দিয়ে সরাসরি পরের পাকে চলে যেতে হবে, অর্থাৎ আবার প্রাপণ্য (menu) দেখিয়ে পছন্দ যোগান নিতে হবে। আমরা এই কাজটি করেছি continue ব্যবহার করে পাক ডিঙিয়ে। এরপরে দেখে আমরা একটি পল্টি-ব্যাপার (switch-case) ব্যবহার করে পছন্দ অনুযায়ী ফলাফল কলন করেছি, তারপর ফলন (output) দিয়েছি।

ক্ষণ ঘূর্ণী (while loop) বা কর ঘূর্ণীতে (do loop) ক্ষান্তি (break) দেয়া ঠিক জন্য ঘূর্ণীতে (for loop) ক্ষান্তি দেয়ার মতোই। নিয়ন্ত্রণ (control) ঘূর্ণী (loop) থেকে বের হয়ে ঘূর্ণীর বাইরে যা আছে সেখানে চলে যাবে। তবে ক্ষণ ঘূর্ণী (while loop) বা কর ঘূর্ণীতে (do loop) পাক ডিঙানোর (continue) সাথে জন্য ঘূর্ণীতে (for loop) পাক ডিঙানোর (continue) কিঞ্চিত তফাৎ আছে। তফাৎটা হলো continue এর পরে জন্য ঘূর্ণীতে (for loop) নিয়ন্ত্রণ (control) হালায়ন (update) অংশে চলে যায়। কিন্তু ক্ষণ ঘূর্ণী (while loop) বা কর ঘূর্ণীতে (do loop) এ হালায়ন অংশতো আলাদা করে নাই। হালায়ন সাধারণত বিবৃতির অংশ হিসাবেই করা হয় যেমন প্রতিবার এখানে posondo এর মান যোগান নেয়া হয়েছে।, কাজেই নিয়ন্ত্রণ ক্ষণ ঘূর্ণী বা কর ঘূর্ণীর ক্ষেত্রে সরাসরি চলে যায় শর্ত পরীক্ষা (condition) অংশে।

৮.১৭ ঘূর্ণী যদি মিথস্ক্রিয়া (Loop and If Interaction)

এমন একটি ক্রমলেখ (program) লিখো যেটি একটি ধনাত্মক পূর্ণক (positive integer) যোগান (input) নিয়ে ১ থেকে সেই পূর্ণক (integer) পর্যন্ত জোড় সংখ্যাগুলো একদিকে আর বিজোড় সংখ্যাগুলো আরেকদিকে যোগ করবে। এই ক্রমলেখতে (program) মূলত আমরা ঘূর্ণীর (loop) ভিতরে যদি-নাহলে (if-else) ব্যবহার না করতে চেষ্টা করবো।

```
cout << "dhonatok purnok ";
int purnok; cin >> purnok;

int jorJogfol = 0, bijorJogfol = 0;
for(int suchok = 1; suchok <= purnok; ++suchok)
{
  if (suchok % 2 != 0)  // বিজোড়
    bijorJogfol += suchok;
  else     // জোড়
    jorJogfol += suchok;
}
cout << jorJogfol << " " << bijorJogfol << endl;
```

উপরের ক্রমলেখতে (program) আমরা প্রথমে একটি ধনাত্মক পূর্ণক (positive integer) যোগান (input) নিয়েছি, তারপর জোড় সংখ্যাগুলোর যোগফলের জন্য jorJogfol আর বিজোড় সংখ্যাগুলোর যোগফলের জন্য bijorJogfol চলক (variable) নিয়েছি। তারপর একটি জন্য ঘূর্ণী (for loop) চলেছে চলক (variable) suchok এর মান ১ থেকে পূর্ণকের মান পর্যন্ত। ঘূর্ণীর ভিতরে সূচক বিজোড় হলে বা (suchok % 2 != 0) শর্ত সত্য হলে সূচকের মান bijorJogfol এর সাথে যোগ হবে আর শর্ত মিথ্যা হলে jorJogfolএর সাথে যোগ হবে।

উপরের ওই ক্রমলেখ (program) আমাদের সঠিক ফলাফল দিবে তবে একটা বিষয় খেয়াল করো সূচকের মান জোড় নাকি বিজোড় এইটা কেন আমাদের ঘূর্ণীর (loop) প্রতি পাকে (lap) পরীক্ষা করতে হবে? এইটা তো আমরা আসলে আগে থেকে জানিই কোন পাকে সূচকের মান জোড় কোন পাকে সেটা বিজোড়। কাজেই ঘূর্ণীর (loop) ভিতরে যে শর্ত পরীক্ষণ সেটা আসলে আমাদের অতিরিক্ত হয়েছে বলে মনে হচ্ছে। এই রকমের ক্ষেত্রে আমরা আসলে একটা ঘূর্ণীর (loop) বদলে দুটো ঘূর্ণী লিখে ফেলতে পারি। নীচে দেখো আমরা তাই করেছি। প্রথম ঘূর্ণীতে ১ থেকে শুরু করে প্রতিপাকে ২ করে বাড়বে, ফলে কেবল বিজোড় সংখ্যাগুলোই হবে সূচকের মান, আর দ্বিতীয় ঘূর্ণীতে ২ থেকে শুরু করে প্রতিপাকে দুই করে বাড়বে, ফলে কেবল জোড় সংখ্যাগুলোই হবে সূচকের মান। প্রথম ঘূর্ণীতে সূচকের মান bijorJogfolএর সাথে আর দ্বিতীয় ঘূর্ণীতে সূচকের মান jorJogfol এর সাথে যোগ করা হয়েছে। তো এইরূপ বিভাজনের ফলে আমাদের আর কোন ঘূর্ণীতেই (loop) শর্ত পরীক্ষা করতে হলো না, অথচ একই ফলাফল পাওয়া গেলো।

```
cout << "dhonatok purnok ";
int purnok; cin >> purnok;

int jorJogfol = 0, bijorJogfol = 0;
for(int suchok = 1; suchok <= purnok; suchok += 2)
  bijorJogfol += suchok;

for(int suchok = 2; suchok <= purnok; suchok += 2)
  jorJogfol += suchok;

cout << jorJogfol << " " << bijorJogfol << endl;
```

এবার একই রকমের আরেকটি ব্যাপার দেখো নীচের ক্রমলেখতে (program)। এটিতে সূচকের মান জোড় নাকি বিজোড় সেটার ওপর ভিত্তি করে যোগ না করে, ঘূর্ণীর (loop) প্রতি পাকে একটা করে নম্বর যোগান (input) নেয়া হয়েছে। যোগান নেয়া নম্বরটি যদি জোড় হয় তাহলে jorJogfolএর সাথে আর বিজোড় হলে bijorJogfolএর সাথে যোগ করা হয়েছে। এই ক্রমলেখয়ের (program) ঘূর্ণীর (loop) ভিতরে থাকা যদি-নাহলেকে (if else) চাইলেও দুটো ঘূর্ণীতে বিভাজন করা সম্ভব না। কারণ এখানে আমাদের আগে থেকে বুঝার উপায় নেই যোগান নেয়া নম্বরটি জোড় হবে নাকি বিজোড় হবে! কাজেই যদি-নাহলে ঘূর্ণীর ভিতরেই থাকবে।

```
cout << "dhonatok purnok ";
int purnok; cin >> purnok;

int jorJogfol = 0, bijorJogfol = 0;
for(int suchok = 1; suchok <= purnok; ++suchok)
{
  int nombor; cin >> nombor;  // যোগান
  if (nombor % 2 != 0)  // বিজোড়
    bijorJogfol += nombor;
  else     // জোড়
    jorJogfol += nombor;
}

cout << jorJogfol << " " << bijorJogfol << endl;
```

তারপর আরো একটি একইরকম ব্যাপার দেখা যাক। এখানে আমাদেরকে purnok চলকের মান ১০এর কম হলে আমরা সূচকের মানগুলো sotoJogfolএ যোগ করতে চাই, আর purnok চলকের মান ১০ বা বেশী হলে সূচকের মানগুলো boroJogfol পেতে চাই। তো নীচের ক্রমলেখতে (program) আমরা একটা ঘূর্ণী (loop) ব্যবহার করে লিখেছি, আর ঘূর্ণীর ভিতরে রয়েছে শর্ত পরীক্ষা (purnok < 10)। শর্ত সত্য হলে sotoJogfolএ যোগ আর শর্ত মিথ্যা হলে boroJogfolএ যোগ। এখানেও আমাদের একই রকমের সমস্যা, শর্ত পরীক্ষণ কি ঘূর্ণীর (loop) ভিতরে দরকার আছে? নাকি এটাকে ঘূর্ণীর বাইরে নেয়া সম্ভব?

```
int purnok;  cin >> purnok; // চলকের মান যোগান
int sotoJogfol = 0, boroJogfol = 0;

for (int suchok = 1; suchok <= purnok; ++suchok)
  if (purnok  < 10)
    sotoJogfol += suchok;
  else
    boroJogfol += suchok;

cout << sotoJogfol << " " << boroJogfol << endl;
```

একটু খেয়াল করলেই একটা বিষয় নজরে আসে সেটা হলো purnok < 10 শর্তটি আসলে কোন ভাবেই সূচকের মানের সাথে সম্পর্কিত নয়, ফলে এই শর্ত পরীক্ষণ আসলে পাকের ওপর নির্ভর করে না। সুতরাং আমরা চাইলে এই শর্তটিকে ঘূর্ণীর (loop) বাইরে নিয়ে যেতে পারি।

নীচের ক্রমলেখ (program) খেয়াল করো আমরা তাই করেছি। ঘূর্ণীতে যাওয়ার আগেই আমরা শর্ত পরীক্ষা করেছি। শর্ত (purnok < 10) সত্য হলে আমরা একটা ঘূর্ণীতে sotoJogfol নির্ণয় করেছি, আর শর্ত মিথ্যা হলে আরেকটি ঘূর্ণীতে (loop) boroJogfol নির্ণয় করেছি।

```
int purnok;  cin >> purnok; // চলকের মান যোগান
int sotoJogfol = 0, boroJogfol = 0;

if (purnok  < 10)
  for (int suchok = 1; suchok <= purnok; ++suchok)
    sotoJogfol += suchok;
else
  for (int suchok = 1; suchok <= purnok; ++suchok)
    boroJogfol += suchok;

cout << sotoJogfol << " " << boroJogfol << endl;
```

উপরের এই ক্রমলেখতে (program) শর্ত মাত্র একবার পরীক্ষা হলো, প্রতিপাকে একই শর্ত বারবার পরীক্ষা করার ব্যাপার আর রইলো না। কাজেই আমাদের ক্রমলেখ খানিকটা দক্ষ হয়ে গেলো, এটা চালাতে সময় কম লাগবে, ঠিক যেটা আমরা করতে চেয়েছিলাম।

৮.১৮ অন্তান্তি স্বাধীন ঘূর্ণী (Nested Independent Loops)

এমন একটি ক্রমলেখ (program) লিখো যেটি একটি দুই-মাত্রার ছকের বর্গগুলোর ক্রমিক নম্বর নীচের মতো করে লিখবে। খেয়াল করো আড়ির (row) ক্রম ১ থেকে ৪, কিন্তু খাড়ির (column) ক্রম উল্টো দিকে ৪ থেকে ১। এই ক্রমলেখ (program) তুমি দুটো অন্তান্তি স্বাধীন ঘূর্ণী (nested independent loop) ব্যবহার করে অর্থাৎ ঘূর্ণীর ভিতরে ঘূর্ণী ব্যবহার করে লিখবে। চাইলে জন্য ঘূর্ণী (for loop), ক্ষণ ঘূর্ণী (while loop), কর ঘূর্ণী (do loop) ব্যবহার করতে পারো।

```
(1,4) (1,3) (1,2) (1,1)
(2,4) (2,3) (2,2) (2,1)
(3,4) (3,3) (3,2) (3,1)
(4,4) (4,3) (4,2) (4,1)
```

তো চলো আমরা প্রথমে এই ক্রমলেখটি (program) জন্য ঘূর্ণী (for loop) দিয়ে লিখি। প্রথমে চলো আমরা প্রথম আড়ির (row) দিকে নজর দেই (1,4) (1,3) (1,2) (1,1)। এই আড়িতে চারটি ক্রমিক দেখানো হয়েছে, প্রতিটি ক্রমিকে দুটি করে সংখ্যা আছে, প্রথমটি আড়ি নম্বর, আর দ্বিতীয়টি খাড়ি (column) নম্বর। তো সবগুলো ক্রমিকের আড়ি নম্বরই 1, কেবল খাড়ি নম্বর বদলে গেছে 4 থেকে শুরু করে 1 পর্যন্ত, প্রতিবার 1 করে কমবে। কাজেই আমরা ঘূর্ণী (loop) চালাবো কেবল খাড়ির জন্য আর আড়িনম্বরটি প্রত্যেক ক্ষেত্রে সরাসরি ফলনে (output) দেখাবো।

```
for(int khari = 4; khari >= 1; --khari)
    cout << "(" << 1 << "," << khari << ") ";
```

উপরের এই ক্রমলেখয়ের ফলন যদি দেখো, তাহলে নীচের মতো লাগবে।

```
(1,4) (1,3) (1,2) (1,1)
```

আমাদের যে ফলন (output) দিতে বলা হয়েছে সেখানে যেহেতু চারটি আড়ি (row) আছে, সেহেতু উপরের ক্রমলেখয়ের (program) মতো ঘূর্ণী (loop) আমরা চারবার লিখলেই কাঙ্খিত ফলন (output) পেয়ে যাবো। তবে প্রতিটা ঘূর্ণীতে কেবল আড়ি নম্বরের জায়গায় নীচের মতো করে 1এর বদলে 2, 3, 4 লিখে নিতে হবে। আর প্রতিটি আড়ির পরে পরের আড়িতে ফলন (output) যাওয়ার জন্য আমাদের cout << endl; লিখতে হবে।

```
for(int khari = 4; khari >= 1; --khari)
    cout << "(" << 1 << "," << khari << ") ";
cout << endl;
for(int khari = 4; khari >= 1; --khari)
    cout << "(" << 2 << "," << khari << ") ";
cout << endl;
for(int khari = 4; khari >= 1; --khari)
    cout << "(" << 3 << "," << khari << ") ";
cout << endl;
for(int khari = 4; khari >= 1; --khari)
    cout << "(" << 4 << "," << khari << ") ";
cout << endl;
```

চারটি আড়ির (row) জন্য না হয় প্রায় একই রকম সংকেত (code) চারবার লিখলাম, কিন্তু আরো বেশী সংখ্যক আড়ির জন্য নিশ্চয় অতবার লিখবো না। আমরা একই কাজ বার বার করার জন্যেই তো ঘূর্ণী (loop) ব্যবহার করা শিখেছিলাম, সেটা কি আড়ির জন্যেও ব্যবহার করতে পারি না? নিশ্চয় পারি। নীচে দেখো আমরা তাই করলাম। আমরা প্রথমে আড়ির জন্য একটি ঘূর্ণী নিয়েছি যেটি আড়ি ১ থেকে ৪ পর্যন্ত প্রত্যেকবার ১ করে বাড়িয়ে চলবে। আর এই ঘূর্ণীর (loop) মহল্লার (block) ভিতরে থাকবে আমাদের আগের লেখা ঘূর্ণীটা যেটা খাড়ির জন্য ঘুরে। মহল্লা (block) ব্যবহার করতে হলো কারণ একটা ঘূর্ণী আর একটা cout মোট দুটো বিবৃতি (statement) নির্বাহ করতে হবে প্রতিটি আড়ির জন্য। আর খাড়ির জন্যে লেখা ভিতরের ঘূর্ণীটাতে যেখানে আড়ির নম্বর 1, 2, 3, 4 সরাসরি লিখে দিয়েছিলাম, এবার সেখানে ari চলক লিখে দিলেই হয়ে গেলো।

```
for(int ari = 1; ari <= 4; ++ari)
{
  for (int khari = 4; khari >= 1; --khari)
    cout << "(" << ari << "," << khari << ") ";
  cout << endl;
}
```

এবার চলো এই ক্রমলেখটিই (program) জন্য ঘূর্ণী (for loop) ব্যবহার না করে আমরা ক্ষণ ঘূর্ণী (while loop) ব্যবহার করে লিখে ফেলি। তবে আড়ি ও খাড়ি সংখ্যা নির্দিষ্ট করে ৪ ধরে না নিয়ে আমরা এখানে ariSonkhya ও khariSonkhya নামে দুটো চলক (variable) ব্যবহার করবো, যার মান তুমি চাইলে যোগান (input) নিতে পারো। জন্য ঘূর্ণী (for loop) থেকে ক্ষণ ঘূর্ণী (while loop) লেখা তো তেমন কঠিন কিছু নয়। জন্য ঘূর্ণীর (for loop) আদ্যায়ন (initialisation) অংশটাকে ক্ষণ ঘূর্ণীর (while loop) আগে লিখে ফেলো, আর জন্য ঘূর্ণীর হালায়ন (update) অংশটাকে ক্ষণ ঘূর্ণীর বিবৃতি (statement) অংশের শেষে দিয়ে দাও। তুমি চাইলে একটা ঘূর্ণীকে জন্য ঘূর্ণী রেখে আরেকটাকে ক্ষণ ঘূর্ণী করে দিতে পারো। তা ছাড়া তুমি চাইলে জন্য বা ক্ষণ ঘূর্ণী বাদ দিয়ে কর ঘূর্ণীও (do loop) ব্যবহার করতে পারো।

```
int ariSonkhya = 4, khariSonkhya = 4;

int ari = 1;
while (ari <= ariSonkhya)
{
  int khari = khariSokhya;
  while (khari >= 1)
  {
    cout << "(" << ari << "," << khari << ") ";
    —khari;
  }
  cout << endl;
  ++ari;
}
```

৮.১৯ অন্তান্তি নির্ভরশীল ঘূর্ণী (Nested Dependent Loop)

ঘূর্ণীর (loop) ভিতরে ঘূর্ণী অর্থাৎ অন্তান্তি ঘূর্ণী (nested loop) ব্যবহার করে এমন একটি ক্রম-লেখ (program) লিখো যেটি নীচের মতো ফলন (output) দিবে। এইক্ষেত্রে ভিতরের ঘূর্ণীটি বাইরের ঘূর্ণীর ওপরে নির্ভরশীল হবে: ভিতরের ঘূর্ণীর সূচকের মান বাইরেরটির সূচকের মানের সাথে সম্পর্কিত হবে। তুমি তিন রকম ঘূর্ণীর (loop) যে কোনটিই ব্যবহার করতে পারো।

```
(1,1)
(2,2) (2,1)
(3,3) (3,2) (3,1)
(4,4) (4,3) (4,3) (4,1)
```

আমরা প্রথমে নীচের মতো করে দুটো অন্তান্তি স্বাধীন ঘূর্ণী (nested independent loop) লিখে ফেলতে পারি যেটা আমরা ঠিক আগের পাঠেই শিখেছি। আলোচনার ধারাবাহিকতা বুঝার জন্য তুমি চাইলে আগের পাঠটি একটু দেখে নিতে পারো। এখানে বাইরের ঘূর্ণী (loop) ৪ বার চলবে, আর তার প্রতি পাকের জন্য ভিতরের ঘূর্ণীটাও ৪ বার চলবে।

```
for(int ari = 1; ari <= 4; ++ari)
{
  for (int khari = 4; khari >= 1; —khari)
    cout << "(" << ari << "," << khari << ") ";
  cout << endl;
}
```

দুটো অন্তান্তি স্বাধীন ঘূর্ণী (nested independent loop) লেখার ফলে আমরা যে ফলন (output) পাবো তা নীচের বাম পাশের মতো। খেয়াল করে দেখো ছকের প্রতিটি আড়িতে (row) প্রতিটি খাড়িতেই (column) সংশ্লিষ্ট বর্গের ক্রমিক নম্বর লেখা হয়েছে। এখন এটার সাথে আমাদের এই পাঠে যে ফলন (output) চাওয়া হয়েছে (ডান পাশেরটি) তা মিলাও।

```
(1,4) (1,3) (1,2) (1,1)                         (1,1)
(2,4) (2,3) (2,2) (2,1)                   (2,2) (2,1)
(3,4) (3,3) (3,2) (3,1)             (3,3) (3,2) (3,1)
(4,4) (4,3) (4,2) (4,1)       (4,4) (4,3) (4,2) (4,1)
```

এবার একটা ব্যাপার খেয়াল করো যে ডান পাশের যে ফলন (output) চাওয়া হয়েছে সেখানে প্রত্যক আড়িতে (row) এমন একটা ঘর থেকে লেখা শুরু হয়েছে যেখানে আড়ি (row) আর খাড়ি (column) সমান, যেমন (1,1), (2,2), (3,3), (4,4)। খাড়ির ক্রমিক আড়ির ক্রমিক থেকে বড় হলে সেই ঘরে কিছু দেখানো হয় নাই। তো এটার জন্য আমরা আমাদের ক্রমলেখতে (program) কেবল ভিতরের ঘূর্ণীটি (loop) কিঞ্চিত বদলে নিবো। ভিতরের ঘূর্ণীটি আগে ছিলো `for (int khari = 4; khari >= 1; --khari)`, এখন সেখানে নীচের মতো করে আদিমান 1 এর বদলে `ari` লিখে দিবো। এর ফলে ভিতরের ঘূর্ণীটি আর স্বাধীন থাকলো না, কারণ এটি কতবার ঘুরবে সেটা নির্ভর করবে বাইরের ঘূর্ণীতে `ari` এর মান কতো তার ওপর।

```
for(int ari = 1; ari <= 4; ++ari)
{
  for (int khari = ari; khari >= 1; --khari)
    cout << "(" << ari << "," << khari << ") ";
  cout << endl;
}
```

উপরের এই ক্রমলেখটির (program) ফলে আমরা যে রকম ফলন পাবো সেটি নীচের বাম পাশের মতো, কিন্তু আমরা যে ফলন (output) পেতে চাই তা ডান পাশের মতো।

```
(1,1)                                           (1,1)
(2,2) (2,1)                               (2,2) (2,1)
(3,3) (3,2) (3,1)                   (3,3) (3,2) (3,1)
(4,4) (4,3) (4,2) (4,1)       (4,4) (4,3) (4,2) (4,1)
```

খেয়াল করো এখনও ঠিক হয়ে ওঠে নি। যথাযথ ভাবে ফাঁকা দিলেই হয়ে যাবে। তো ফাঁকা দেওয়ার ক্ষেত্রে খেয়াল করো আমরা ১ম আড়িতে ফাঁকা দিয়েছি ৩টি, ২য়টিতে ২টি, ৩য়টিতে ১টি, ৪র্থটিতে ০টি। অর্থাৎ `ari` এর মান অনুযায়ী 4−`ari` সংখ্যক ফাঁকা দিয়েছি। অথবা বলতে পারো ৪ থেকে গোনা ও ফাঁকা দিতে শুরু করেছি, কিন্তু `ari`র চেয়ে বড় সংখ্যা পর্যন্ত ফাঁকা দিয়েছি, আর সমান হলে তো ঘরের স্থানাংক দেখানো শুরু করেছি। তাহলে সব মিলিয়ে আমরা নীচের ক্রমলেখয়ের (program) মতো করে আরেকটি ঘূর্ণী (loop) ব্যবহার করতে পারি ফাঁকা দেওয়ার জন্য। এই ঘূর্ণীটিও কত বার ঘুরবে, সেটা কিন্তু বাইরের ঘূর্ণীর ওপর নির্ভর করবে।

```
// প্রতি আড়ির জন্য একটা করে পাক।
for(int ari = 1; ari <= 4; ++ari)
{
  // প্রতি আড়িতে প্রথমে ফাঁকা দেওয়ার জন্য
  for (int khari = 4; khari > ari; --khari)
    cout << "      "; // মালার ভিতরে ছয়টি ফাঁকা

  // প্রতি আড়িতে ঘরগুলোর স্থানাঙ্ক লেখার জন্য
```

```
    for (int khari = ari; khari >= 1; —khari)
      cout << "(" << ari << "," << khari << ") ";
    cout << endl;
  }
```

তুমি অবশ্য চাইলে কেবল এই সমস্যাটির ক্ষেত্রে ভিতরের ঘূর্ণী দুটিকে ঘূর্ণী যদি মিথস্ক্রিয়া (Loop and If interaction) বিবেচনা করে একটা ঘূর্ণী দিয়েই সারতে পারতে। কারণ ভিতরের দুটি ঘূর্ণী মিলিয়ে তো 4 হতে 1 গুনতি চলে, khariর মান ari হতে বড় হলে ফাঁকা দেখানো হয় আর না হলে ঘরের স্থানাঙ্ক দেখানো হয়। সুতরাং একটা যদি-নাহলে (if else) লাগালেই হবে।

```
  for(int ari = 1; ari <= 4; ++ari)
  {
    for (int khari = 4; khari >= 1; —khari)
      if (khari > ari)
        cout << "      ";
      else
        cout << "(" << ari << "," << khari << ") ";
    cout << endl;
  }
```

৮.২০ গভীর অন্তান্তি ঘূর্ণী (Deeply Nested Loops)

ঘূর্ণীর (loop) ভিতরে ঘূর্ণী তার ভিতরে ঘূর্ণী ব্যবহার করে তিনটি সংখ্যা 1, 2, 3 এর বিন্যাস (permutation) ফলন (output) দাও। বিন্যাসগুলোতে একই সংখ্যা বারবার ব্যবহার করা যাবে হলে কী করবে, আর একই সংখ্যা একের অধিকবাব ব্যবহার না করা গেলে কী করবে?

```
  for (int x = 1; x <= 3; ++x)
    for (int y = 1; y <= 3; ++y)
      for (int z = 1; z <= 3; ++z)
        cout << x << " " << y << " " << z << endl;
```

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

এই ক্রমলেখয়ের (program) জন্য আমরা এভাবে চিন্তা করি: প্রথম স্থানটিতে সংখ্যা তিনটি একে একে বসাতে হবে কাজেই একটা ঘূর্ণী (loop) লাগবে। তারপর দ্বিতীয় স্থানের জন্যেও সংখ্যা

তিনটি একে একে বসাতে হবে, সুতরাং আরেকটা ঘূর্ণী লাগবে। আর একই ভাবে তৃতীয় স্থানের জন্যেও আরেকটি ঘূর্ণী দিয়ে সংখ্যা তিনটি একে একে বসাতে হবে। কাজেই সব মিলিয়ে আমাদের ঘূর্ণী লাগবে তিনটি, আর বিন্যাস পাওয়া যাবে সর্বমোট ২৭ টি। তো এরকম একটি ক্রমলেখ আমরা উপরে দেখালাম, খুবই সহজ ক্রমলেখ। আর ওই ক্রমলেখয়ের ফলন (output) কেমন হবে সেটাও উপরে দেখানো হয়েছে। তবে স্থানের ব্যবহার বাড়ানোর জন্য ২৭ টি বিন্যাস নীচে নীচে না লেখে তিন স্তম্ভে (column) দেখানো হয়েছে, আসলে ওগুলো একের পর এক নীচে নীচে আসবে।

উপরের ক্রমলেখতে (program) কিন্তু একই সংখ্যা একের অধিকবার ব্যবহার করা হয়েছে। যদি সেটা করতে না দেয়া হয়, তাহলে আমরা যেটা করতে পারি তা হলো যখনই দুটি সংখ্যা এক হয়ে যাবে তখন আমরা ফলন (output) দিবো না। অর্থাৎ x যদি yএর সমান হয় অথবা x যদি zএর সমান হয়, অথবা y যদি zএর সমান হয় তাহলে ফলন হবে না, আর না হলে ফলন হবে। তার মানে ফলন দেয়া হবে !(x == y || x == y || y == z) শর্ত সত্য হলে, আর দেয়া হবে না শর্ত মিথ্যা হলে। বুলক বীজগণিতের ডি মরগ্যানের সূত্রানুযায়ী আমরা এটাকে সরলীকরণ করতে পারি। তাহলে পাবো !(x == y) && !(x == z) && !(y == z) বা (x != y && x != z && y != z)। সবমিলিয়ে এমন ক্রমলেখ আর তার ফলন হতে পারে নীচের মতো।

```
for (int x = 1; x <= 3; ++x)
  for (int y = 1; y <= 3; ++y)
    for (int z = 1; z <= 3; ++z)
      if (x != y && x != z && y != z)
        cout << x << " " << y << " " << z << endl;
```

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

একটা বিষয় খেয়াল করো উপরের ক্রমলেখতে (program) তিনটি ঘূর্ণীই কিন্তু তিনবার করে ঘুরবে, ফলে মোট ২৭ টি পাকই সম্পন্ন হবে, তবে এই ২৭টি পাকের মাত্র ৬টিতে ফলন (output) আসবে, বাঁকীগুলোতে যদি-নাহলের (if else) শর্ত মিথ্যা হওয়ায় ফলন আসবে না। কথা হচ্ছে ওই যে ২১টি পাক যেগুলোতে কোন ফলন আসবে না, সেগুলো কমানো সম্ভব কিনা। কিছুটা তো সম্ভব। দ্বিতীয় ঘূর্ণীর কথা বিবেচনা করো, যখন আমরা জানিই যে y এর মান x এর সমান, তখন তো তৃতীয় ঘূর্ণীটি ঘুরিয়ে লাভ নেই, আমাদের কোন ফলন আসবে না। কাজেই আমরা যদি নাহলে মিথস্ক্রিয়া (if else interaction) বিবেচনা করে নীচের ক্রমলেখয়ের মতো করে if (x != y) কে তৃতীয় ঘূর্ণীর উপরে নিয়ে আসতে পারি। এই ক্রমলেখয়ের ক্ষেত্রে কোন ঘূর্ণী কত বার ঘুরবে? তুমি কি নিজে নিজে সেগুলো হিসাব করতে পারবে, চেষ্টা করে দেখো?

```
for (int x = 1; x <= 3; ++x)                    //৩ বার
  for (int y = 1; y <= 3; ++y)                  // ৩*৩ = ৯বার
    if (x != y)                                 // ৩*১ = ৩ বার মিথ্যা
      for (int z = 1; z <= 3; ++z)              // ৬*৩ = ১৮বার
        if (x != z && y != z)                   // ৬ বার সত্য
          cout << x << " " << y << " " << z << endl;
```

আচ্ছা তোমাকে তিনটি সংখ্যা না দিয়ে বরং চারটি বা পাঁচটি বা আরো বেশী সংখ্যার বিন্যাস (permutation) ফলন (output) দিতে বলা হয় তুমি কী পারবে তার জন্যে ক্রমলেখ (program) লিখতে? নিশ্চয় পারবে, যতটি সংখ্যা নিয়ে বিন্যাস করতে হবে ততটি ঘূর্ণী (loop) নিলেই হয়ে গেলো। এই যে ঘূর্ণীর ভিতরে ঘূর্ণী (loop inside loop), তার ভিতরে ঘূর্ণী, তার ভিতরে আরো ঘূর্ণী এগুলো হলো গভীর অন্তান্তি ঘূর্ণী (deeply nested loop), যতটা ভিতরে একটা ঘূর্ণী ততটা হলো তার গভীরতা। যেমন উপরের ক্রমলেখতে সবচেয়ে ভিতরের ঘূর্ণীর গভীরতা হলো ৩, মাঝখানেরটার গভীরতা হলো ২ আর বাইরেরটার গভীরতা হলো ১। আমরা সাধারণত খুব বেশী গভীরতার অন্তান্তি ঘূর্ণী তৈরী করতে চাই না। যেমন আরো বেশী সংখ্যার বিন্যাস করতে গেলেই আমরা আর এ রকম গভীর অন্তান্তি ঘূর্ণী ব্যবহার করবো না, বরং আমরা অন্য কোন পদ্ধতির খোঁজ করবো। তাছাড়া এরকম গভীর ঘূর্ণী আরো একটা ক্ষেত্রেও অসুবিধাজনক। যেমন ধরো তোমাকে যোগান (input) নিতে হবে কয়টা সংখ্যার বিন্যাস করতে চাও। তো সেটাতো আগে থেকে মানে ক্রমলেখ লেখার সময় জানা সম্ভব না, কাজেই ক্রমলেখ লেখার সময় কত গভীরতা পর্যন্ত ঘূর্ণী লিখবো সেটাও জানা সম্ভব না, আর তাই এরকম করে ক্রমলেখ লেখা আসলেই সম্ভব হবে না।

৮.২১ অন্তান্তি ঘূর্ণী হ্রাসকরণ (Deflating Nested Loops)

ধরো তোমাকে এমন একটা ক্রমলেখ (program) লিখতে হবে যেটি একদিনের ২৪ ঘন্টায় প্রতি সেকেন্ডে সময় ফলন (output) দিবে ১০:৩৯:৪৬ এই ছাঁচে। এই ক্রমলেখ তোমাকে অন্তান্তি ঘূর্ণী (nested loops) ব্যবহার না করে কেবল একটি ঘূর্ণী (loop) ব্যবহার করেই লিখতে হবে।

ফিরিস্তি ৮.৯: ঘড়ির সময় দেখানো (Displaying Clock Time)

```
for(int h = 0; h < 24; ++h)
  for(int m = 0; m < 60; ++m)
    for(int s = 0; s < 60; ++s)
      cout << h << ":" << m << ":" << s << endl;
```

প্রথমে আমরা অন্তান্তি ঘূর্ণী (nested loop) ব্যবহার করেই ক্রমলেখটি (program) লিখি। আমাদের ঘন্টা চলবে ০ হতে ২৩ পর্যন্ত, মিনিট চলবে ০ হতে ৫৯ পর্যন্ত, আর সেকেন্ডও চলবে ০ হতে ৫৯ পর্যন্ত। সুতরাং ৩ গভীরতার অন্তান্তি ঘূর্ণী হলেই আমাদের চলবে। উপরের ক্রমলেখতে দেখো তিনটি ঘূর্ণী একটার ভিতরে আরেকটা লিখে আমরা তা করেছি।

এবার আমরা অন্তান্তি ঘূর্ণী (nested loop) ব্যবহার না করে একটা ঘূর্ণী ব্যবহার করে ক্রমলেখটি (program) লেখার চেষ্টা করবো। সারাদিনে আমাদের মোট সেকেন্ড আছে কতটি? ২৪ * ৬০ * ৬০ = ৮৬৪০০টি। তাহলে আমাদের একটি ঘূর্ণী চালাতে হবে ৮৬৪০০ বার। আর প্রতিবারে সেকেন্ডকে ৬০ দিয়ে ভাগ করে মিনিটে আর মিনিটকে ৬০ দিয়ে ভাগ করে ঘন্টায় প্রকাশ করতে হবে। তারপর অবশিষ্ট সেকেন্ড, অবশিষ্ট মিনিট, ও কত ঘন্টা হলো তা ফলনে (output) দেখাতে হবে। ভাগফল / আর ভাগশেষ % ব্যবহার করে আমরা এই ক্রমলেখ নীচের মতো করে লিখবো।

```
for(int k = 0; k < 86400; ++k)
{
  int h, m, s = k;  // k কে ঘন্টা মিনিট সেকেন্ড নিতে হবে

  m = s / 60;       // মিনিটে রূপান্তর
  s = s % 60;       // অবশিষ্ট সেকেন্ড
```

```
    h = m / 60;          // ঘন্টায় রূপান্তর
    m = m % 60;          // অবশিষ্ট ঘন্টা

    cout << h << ":" << m << ":" << s << endl;
  }
```

তুমি কিন্তু চাইলে উপরের মতো করে প্রতিবার সেকেন্ডকে ৬০ দিয়ে ভাগ করে মিনিটে, তারপর আবার ৬০ দিয়ে ভাগ করে ঘন্টায় প্রকাশ না করে অন্যভাবেও করতে পারো। ধরো সেকেন্ড ঘূর্ণীর প্রতি পাকে ১ করে বাড়লো। আর যখন ৬০ সেকেন্ড হয়ে গেলো তখন আমরা মিনিটে এক যোগ করে দিলাম, আর সেকেন্ডকে আবার ০ বানিয়ে দিলাম। একই ভাবে মিনিট যদি ৬০ হয়ে যায় তাহলে ঘন্টাকে এক বাড়িয়ে দিলাম, আর মিনিটকে ০ বানিয়ে দিলাম। আর যখন ঘন্টা ২৪ হয়ে গেলো তখন ক্রমলেখ শেষ করে দিলাম। তো এই রকম ক্রমলেখ (program) আমরা নীচে দেখালাম।

```
  int h = 0, m = 0, s = 0;      // আদি মান
  while(h < 24)
  {
    cout << h << ":" << m << ":" << s << endl;

    if (++s == 60)      // সেকেন্ড এক বাড়িয়ে ৬০ হলে
    {
      s = 0;            // সেকেন্ড হবে শুন্য
      if (++m == 60)    // মিনিট এক বাড়বে, আর ৬০ হলে
      {
        m = 0;          // মিনিট হবে শুন্য
        ++h;            // ঘন্টা এক বাড়বে
      }
    }
  }
```

আসলে যে কোন অন্তান্তি ঘূর্ণীকে (nested loop) এই ভাবে কেবল একটা ঘূর্ণী দিয়েই লিখে ফেলা যায়। অন্তান্তি ঘূর্ণীতে সূচকগুলোর মান যে ক্রমে বদল হতে থাকে, উপরের এই একটা ঘূর্ণীতেও চলকগুলোর মান সেই একই ক্রমেই বদল হতে থাকে।

৮.২২ ছদ্মবেশের অন্তান্তি ঘূর্ণী (Nested Loop in Disguise)

অন্তান্তি ঘূর্ণী (nested loop) ব্যবহার করে এবং না করে (1) + (1 + 2) + (1 + 2 + 3) + ... + (1 + 2 + ... + n) ধারাটির যোগফল নির্ণয়ের ক্রমলেখ (program) রচনা করো। এখানে তুমি 1 + 2 + ... + n = n(n + 1)/2 এ রকম সূত্র ব্যবহার করতে পারবে না।

প্রদত্ত ধারাটিতে বন্ধনীর ভিতরে অংশগুলোকে যদি একটা করে পদ ধরে নাও তাহলে প্রথম পদ (1), দ্বিতীয় পদ (1 + 2), আর এই ভাবে nতম পদ (1 + 2 + ... + n)। কাজেই উপরের ধারাটিতে আমাদের nটি পদ আছে, সুতরাং আমাদের একটি ঘূর্ণী (loop) লাগবে যেটি 1 থেকে n পর্যন্ত ঘুরবে। এবার বন্ধনীর ভিতরের প্রতিটি পদের দিকে তাকাই। ধরা যাক আমরা kতম পদ

বিবেচনা করছি, তাহলে বুঝতেই পারছো পদটি হবে (1 + 2 + ... + k)। এখানে এই পদটি নিজেও একটা ধারা। কাজেই আমাদের পুরো ধারাটি আসলে ধারার ভিতরে ধারা, বা অন্তান্তি ধারা (nested series)। যাইহোক, kতম পদ (1 + 2 + ... + k) তো আমরা আরেকটি ঘূর্ণী 1 থেকে k পর্যন্ত ঘুরিয়ে সহজেই হিসাব করে ফেলতে পারি। তাহলে সব মিলিয়ে প্রদত্ত ধারার জন্য আমাদের ঘূর্ণীর ভিতরে ঘূর্ণী বা অন্তান্তি ঘূর্ণী (nested loop) ব্যবহার করতে হবে।

```
int n = 10;    // যোগান নিতে পারো
int s = 0;     // পুরো ধারার যোগফল
for(int k = 1; k <= 10; ++k)
{
  int t = 0;   // বন্ধনীতে পদের যোগফল
  for(int l = 1; l <= k; ++l)
    t += l;    // বন্ধনীর ভিতরে যোগফল
  s += t;      // পুরো ধারার যোগফল
}
cout << s << endl;
```

উপরের ক্রমলেখতে (program) দুটো অন্তান্তি ঘূর্ণী (nested loop) মিলিয়ে ঠিক কতবার ঘুরবে? বাইরের ঘূর্ণীতে kএর মান যখন 1 তখন ভিতরের ঘূর্ণী ঘুরবে 1 বা, বাইরের ঘূর্ণীতে kএর মান যখন 2 তখন ভিতরের ঘূর্ণী ঘুরবে 2 বার, এই ভাবে বাইরের ঘূর্ণীতে kএর মান যখন n তখন ভিতরের ঘূর্ণী ঘুরবে n বার। কাজেই বাইরের ঘূর্ণীর সব পাক মিলিয়ে ভিতরের ঘূর্ণী ঘুরবে 1 + 2 + ... + n = n(n+1)/2 বার। তার মানে nএর মান 10 হলে দুই ঘূর্ণী মিলে পাক খাবে সর্বমোট 10(10+1)/2 = 55 বার। কথা হচ্ছে এই ধারাটির যোগফল বের করতে আসলেই কি এত পাকের দরকার আছে? বিশেষ করে বন্ধনীর ভিতরের প্রতিটি পদ কেন আলাদা করে আবার নতুন করে হিসাব করতে হবে? আগের বন্ধনীর ভিতরের পদ জানা থাকলে তো তার সাথে কেবল পরের পূর্ণক (integer) যোগ করেই পরের বন্ধনীর ভিতরের পদ বের করা সম্ভব।

```
int n = 10;    // যোগান নিতে পারো
int s = 0, t = 0; // ধারা ও পদের যোগফল
for(int k = 1; k <= 10; ++k)
{
  t += k;    // বন্ধনীর ভিতরে যোগফল
  s += t;      // পুরো ধারার যোগফল
}
cout << s << endl;
```

উপরের ক্রমলেখতে (program) দেখো আমরা শুরুতে পদ tএর আদি মান শুন্য ধরে নিয়েছি। আর ঘূর্ণীর ভিতরে ঢুকেই tএর সাথে k যোগ করে দিচ্ছি, যাতে বন্ধনীর ভিতরে থাকা আগের পদে t এর মান যত ছিলো, এই পাকে যাতে tএর মান তার চেয়ে যাতে k বেশী হয়, কারণ পরের বন্ধনীর ভিতরে পদে তো kটাই অতিরিক্ত আছে। তারপর t টাকে sএর সাথে যোগ করলেই ধারার যোগফল হয়ে গেলো। তো এই ক্রমলেখতে ঘূর্ণী কত বার ঘুরবে? বুঝতেই পারছো মাত্র 10 বার।

তাহলে আমরা দেখলাম দেখতে অন্তান্তি ঘূর্ণী (nested loop) মনে হলেও অনেক সময় একটা ঘূর্ণী ব্যবহার করেই দক্ষ ক্রমলেখ (program) রচনা করা যায়। অন্তান্তি ঘূর্ণী লিখার সময় সেটা আসলেই অন্তান্তি ঘূর্ণী নাকি স্রেফ ছদ্মবেশী এ ব্যাপারে সতর্ক থাকবে কেমন!

৮.২৩ অনুশীলনী সমস্যা (Exercise Problems)

ধারণাগত প্রশ্ন: নীচে কিছু ধারণাগত প্রশ্ন রয়েছে। প্রশ্নগুলোর উত্তর নিজে নিজে বের করবে।

১. পুনালি পরিগণনা (iterative programming) বলতে কী বুঝো? আলোচনা করো।

২. জন্য ঘূর্ণীতে (for loop) চারটি অংশ আছে। এগুলো হলো আদ্যায়ন (initialisation), শর্ত (condition), বৃদ্ধি (update), বিবৃতি (statement)। কোন অংশ কখন নির্বাহিত হয়, কোনটি কতবার নির্বাহিত হয়, আর কার পরে কোনটি নির্বাহিত হয় আলোচনা করো।

৩. সমান্তর ধারার বর্তমান পদটিকে ঘূর্ণীর (loop) সূচকের সাথে সম্পর্কিত করা বনাম আগের পাকের সাথে সম্পর্কিত করার মধ্যে পরিগণনায় কী তফাৎ ঘটে আলোচনা করো।

৪. ঘূর্ণীতে (loop) ক্ষান্তির (break) ব্যবহার উদাহরণসহ আলোচনা করো।

৫. ঘূর্ণীতে (loop) পাক ডিঙানো (continue) উদাহরণসহ আলোচনা করো।

৬. ঘূর্ণীতে (loop) শর্ত ফাঁকা (empty condition) হলে ঘূর্ণী থামবে কী করে?

৭. অসীম ঘূর্ণী (infinite loop) কী? অসীম ঘূর্ণী কি কাঙ্খিত না অনাকাঙ্খিত?

৮. জন্য ঘূর্ণীকে (for loop) কী ভাবে সাধারণ (general) ঘূর্ণী হিসাবে ব্যবহার করা যায়?

৯. পাকের আগে শর্ত পরীক্ষণ ও পাকের পরে শর্ত পরীক্ষণ বিষয়ে আলোচনা করো।

১০. ঘূর্ণী ও যদির মিথস্ক্রিয়া (interaction) কী ভাবে ক্রমলেখয়ের গতিতে প্রভাব ফেলে?

১১. অন্তান্তি (nested) একাধিক ঘূর্ণীকে কী ভাবে একটা ঘূর্ণী ব্যবহার করেই সামলানো যায়?

পরিগণনার সমস্যা: নীচে আমরা কিছু পরিগণনার সমস্যা দেখবো। এই সমস্যাগুলো আগে ধৈর্য্য ধরে নিজে নিজে সমাধান করতে চেষ্টা করবে। যখন একেবারেই পারছো না বলে মনে হয় তখনই কেবল সমাধান দেখে নিতে পারো। সমাধানগুলো পরিগণনার প্রশ্নগুলোর শেষে আছে।

১. নীচের নকশার মতো নকশা তৈরী করো। এই নকশার কোনার বিন্দুগুলোতে + আছে, একদম বাম আর ডান পাশে আছে |, আর অন্য সবগুলো হলো —, প্রতিটি সারিতে — আছে ২০টি করে। প্রত্যেক সারির —গুলোর জন্য তোমাকে একটি করে ঘূর্ণী (loop) লিখতে হবে।

```
+————————————————————+
|————————————————————|
|————————————————————|
+————————————————————+
```

২. এমন একটি ক্রমলেখ (program) রচনা করো যেটি একটি ধনাত্মক (positive) পূর্ণক (integer) যোগান (input) নিয়ে সেটা মৌলিক (prime) সংখ্যা কিনা নির্ণয় করবে।

৩. এমন একটি ক্রমলেখ (program) লিখো যেটি দুটো ধনাত্মক পূর্ণক (integer) যোগান নিয়ে তাদের গসাগু (HCF) ও লসাগু (LCM) নির্ণয় করে।

৮.২৩. অনুশীলনী সমস্যা (Exercise Problems)

৪. নীচের ক্রমলেখয়ের (program) ফলন (output) কী হবে, গণনিতে (computer) না চালিয়ে বের করো। তারপর গণনিতে চালিয়ে তোমার হিসাব করা ফলাফল যাচাই করো। যদি কোন অসীম ঘূর্ণী (infinite loop) থেকে থাকে সেটাকে মেরামতো করো।

```
int n = 3;
while (n >= 0)     // প্রথম ঘূর্ণী
{
  cout << n * n << " ";
  --n;
}
cout << n << endl;

while (n < 4)      // দ্বিতীয় ঘূর্ণী
  cout << ++n << " ";
cout << n << endl;

while (n >= 0)     // তৃতীয় ঘূর্ণী
  cout << (n /= 2) << " ";
cout << endl;
```

৫. একজন অনভিজ্ঞ ক্রমলেখক নীচের ক্রমলেখটি (program) লিখেছে। ক্রমলেখটির ছাড়ন (indentation) দেখে যেমন মনে হচ্ছে ক্রমলেখটি ঠিক তেমন ফলন (output) দিচ্ছে না। অর্থাৎ ক্রমলেখক চেয়েছিলেন ১০ থেকে শুরু করে প্রতিবার ২ দিয়ে ভাগ করবেন আর ভাগফলের বর্গ দেখাবেন। ভাগ করতে গিয়ে শুন্য হয়ে গেলে থেমে যাবেন। সুতরাং তার কাঙ্খিত ফলন হচ্ছে 25 4 1 কিন্তু ক্রমলেখটি হতে সেরকম ফলন আসছে না। তো তুমি প্রথমে এই ক্রমলেখ যেমন আছে তেমন রেখেই এর ফলন নির্ণয় করো। আর সেক্ষেত্রে ছাড়ন কেমন হবে সেটাও দেখাও। তারপর কাঙ্খিত ফলাফল পেতে গেলে ক্রমলেখতে কী পরিবর্তন করতে হবে সেটাও করে দেখাও।

```
int n = 10;
while (n > 0)
  n /= 2;
  cout << n * n << " ";
cout << endl;
```

৬. নীচের ক্রমলেখটি (program) কী করবে বর্ণনা করো। তারপর এটিকে এমন ভাবে আবার লিখো যাতে এতে ক্ষণ ঘূর্ণীর (while loop) বদলে করো ঘূর্ণী (do loop) ব্যবহৃত হয়, কিন্তু সব মিলিয়ে ক্রমলেখয়ের বৈশিষ্ট্য একই থাকে।

```
int n;
cout << "dhonatok sonkhya: ";
cin >> n;

while (n <= 0)
{
```

```
    cout << "dhonatok noy." << endl;
    cout << "dhonatok sonkhya: ";
    cin >> n;
  }
```

৭. নীচের ক্রমলেখটির (program) ফলন কী? এটিকে এমন ভাবে বদলে লেখো যাতে মহল্লা (block) ব্যবহার না করেই একই ফলাফল পাওয়া যায়। তারপর ক্রমলেখটিকে ক্ষণ ঘূর্ণী (while loop) ব্যবহার না করে জন্য ঘূর্ণী (for loop) ব্যবহার করে লিখো।

```
int i = 5;
while (i > 0)
{
  i = i + 1;
  cout << i << endl;
}
```

৮. এমন একটি ক্রমলেখ রচনা করো যেটি একটি ঘূর্ণীর (loop) ভিতরে ব্যবহারকারীর কাছে থেকে একের পর এক একটি করে পূর্ণক (integer) যোগান নিবে। যোগান নেওয়া সংখ্যাটি ধনাত্মক না হলে ক্রমলেখ থেকে বের হয়ে যাবে, আর ধনাত্মক হলে মানের ক্রমানুসারে সংখ্যাটির উৎপাদকগুলোকে পরপর এক সারিতে ফলন (output) দিবে, আর পরের সংখ্যা যোগান নিতে চাইবে। নমুনা যোগান-ফলন (sample input output) নিম্নরূপ:

```
> 0 utpadok <= 0 shesh
sonkhya koto? 36
utpadok talika: 36 18 12 9 4 3 2 1
> 0 utpadok <= 0 shesh
sonkhya koto? -1
kromolekho shesh!
```

৯. এমন একটি ক্রমলেখ (program) রচনা করো যেটি একটি ধনাত্মক পূর্ণ সংখ্যা যেমন 23154 যোগান (input) নিয়ে ফলন দিবে 45132।

১০. এমন একটি ক্রমলেখ (program) রচনা করো যেটি একে একে সংখ্যা যোগান (input) নিবে যতক্ষণ ধনাত্মক সংখ্যা দেওয়া হচ্ছে, অধনাত্মক সংখ্যা হলে ক্রমলেখ শেষ হবে। ক্রমলেখটির ফলন (output) হবে যোগান নেয়া সংখ্যাগুলোর মধ্যে সবচেয়ে বড়টি আর সেটি কত নম্বরে যোগান দেওয়া হয়েছিলো সেই ক্রমিক নম্বরটি।

১১. ফিবোনাস্যি (Fibonacci) প্রগমন হলো ০, ১, ১, ২, ৩, ৫, ৮, ...। লক্ষ্য করো এই প্রগমনের প্রথম দুটি পদ হলো ০ আর ১। আর এর পর থেকে প্রতিটি পদ তার আগের দুটো পদের যোগফল। এমন একটি ক্রমলেখ (program) রচনা করো যেটি একটি ধনাত্মক পূর্ণক n যোগান (input) নিয়ে nতম পদ নির্ণয় করবে।

১২. একটি ধনাত্মক পূর্ণক n যোগান নিয়ে প্রথম n স্বাভাবিক সংখ্যার (১, ২, ৩, ...) যোগফল ও গুণফল নির্ণয়ের ক্রমলেখ (program) রচনা করো। তুমি হয়তো জানো প্রথম n স্বাভাবিক

সংখ্যার গুণফলকে উৎপাদকীয় (factorial) বলা হয়। উৎপাদকীয় খুবই বড় সংখ্যা হয় যা intএ নাও ধরতে পারে, কাজেই n বড় হলে আমরা উল্টাপাল্টা ফল পেতে পারি।

১৩. নীচের ধারাগুলোর প্রথম n পদের সমষ্টি নির্নয় করো। তোমার ক্রমলেখতে (program) তুমি n যোগান (input) হিসাবে নিবে, আর ধারাটির সমষ্টি ফলন (output) দিবে।

ক) লম্বানুপাত ধারা (sine series): $\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$

খ) লগ্নানুপাত ধারা (cosine series): $\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$

গ) ভাজিত ধারা (harmonic series): $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$

ঘ) অয়লার সংখ্যা (Euler number): $e_n = 1 + \frac{1}{1!} + \frac{1}{2!} + \ldots + \frac{1}{(n-1)!}$

ঙ) গুণোত্তর ধারা (geometric series): $\frac{1}{1-x} = 1 + x + x^2 + \ldots + x^{n-1}$

১৪. দশটা সংখ্যা যোগান (input) নিয়ে তাদের গড় (mean) ও প্রমিত বিচ্যুতি (standard deviation) নির্ণয় করো। প্রমিত বিচ্যুতি হলো সংখ্যাগুলোর বর্গের গড় থেকে গড়ের বর্গ বিয়োগ করে বিয়োগফলের বর্গমূল অর্থাৎ $\sqrt{\frac{\sum x^2}{n} - (\frac{\sum x}{n})^2}$।

১৫. একটি শ্রেণীতে n সংখ্যক শিক্ষার্থী আছে আর তাদের প্রত্যেকে m সংখ্যক বিষয়ে পরীক্ষা দিয়েছে। প্রত্যেক ছাত্রের প্রত্যেক বিষয়ের নম্বর যোগান (input) নিয়ে প্রত্যেক ছাত্রের মোট নম্বর ফলন (output) দাও।

১৬. ০ থেকে ৯ পর্যন্ত নামতার সারণী (table) লিখার জন্য একটি ক্রমলেখ রচনা করো। সারণীকে সুন্দর করে সাজানোর জন্য তোমাকে এক অঙ্কের সংখ্যা ও দুই অঙ্কের সংখ্যা মাথায় রাখতে হবে। এক অঙ্কের সংখ্যার জন্য তুমি প্রথমে একটি অতিরিক্ত ফাঁকা (space) দিয়ে নিবে। তাতে সারণীতে আড়ি ও খাড়ি (row and column) ঠিক মতো থাকবে। আর নামতার সারণীতে সবচেয়ে বামের খাড়িতে (column) আর উপরের আড়িতে (row) অবশ্যই ০ থেকে ৯ পর্যন্ত সংখ্যাগুলো থাকবে শিরোনাম হিসাবে।

১৭. এমন একটি ক্রমলেখ (program) রচনা করো যেটি ৫০ জন ছাত্রের নম্বর যোগান (input) নিবে, আর ফলন দিবে কোন পাল্লায় কতজন ছাত্রের নম্বর পড়েছে তা। প্রতি ১০ নম্বরের জন্য একটি করে পাল্লা চিন্তা করতে পারো, আর মোট নম্বর হলে ৩০ যেখানে এক জন ছাত্র ১ থেকে ৩০ পর্যন্ত যে কোন নম্বর পেতে পারে। কাজেই মোটা পাল্লা রয়েছে ৩টি।

১৮. তারকা ব্যবহার করে নীচের বিভিন্ন রকম আকৃতিগুলো ফলন দাও। প্রতিটি ক্ষেত্রে তুমি দরকার মতো তুমি পরামিতি (parameter) n যোগান (input) নিবে। পরামিতি মানে হচ্ছে সারির সংখ্যা বা সারিতে সর্বোচ্চ কয়টি তারা থাকবে বা থাকবে না এগুলো n ওপর নির্ভরশীল। নীচের প্রতিটি ক্ষেত্রে nএর মান 5।

```
*                    *                  *
**                  **                 ***
***                ***                *****
****              ****               *******
*****            *****             *********
```

```
*              *                *
**             **              ***
***            ***            *****
****           ****          *******
*****          *****        *********
****            ****        *******
***              ***        *****
**                **        ***
*                  *        *
```

১৯. দ্বিপদী সহগ (binomial coefficient) হলো ${}^nC_r = \frac{n}{1} * \frac{n-1}{2} * \ldots * \frac{n-r+1}{r}$। দ্বিপদী সহগগুলো নিয়ে একটা ত্রিভুজ তৈরী করা যায় যাকে বলা হয় প্যাসক্যাল ত্রিভুজ। প্যাসক্যাল ত্রিভুজ নীচে দেখানো হলো। প্রথম সারিতে মানটি হলো ${}^0C_0$, দ্বিতীয় সারিতে মানগুলো হলো ${}^0C_1$ ও ${}^1C_1$, তৃতীয় সারিতে ${}^0C_2$, ${}^1C_2$ ও ${}^2C_2$। পরামিতি n যোগান (input) নিয়ে তার জন্য প্যাসক্যাল ত্রিভুজ ফলন (output) দাও। নীচের প্যাসক্যাল ত্রিভুজের পরামিতি ৪। তুমি চাইলে উল্টিয়ে পাল্টিয়ে নানান রকমের প্যাসক্যাল ত্রিভুজ তৈরী করতে পারো।

```
        1
      1   1
    1   2   1
  1   3   3   1
1   4   6   4   1
```

২০. এমন একটি ক্রমলেখ (program) রচনা করো যেটি তোমার সাথে একটি খেলা খেলবে। ক্রমলেখটি তোমাকে মনে মনে ০ থেকে ১০২৩ এর মধ্যে একটি নম্বর মনে মনে ধরতে বলবে। আর তারপর ক্রমলেখটি তোমাকে বেশ কিছু প্রশ্ন করবে, এই যেমন তোমার ধরেন নেওয়া সংখ্যাটি অমুক সংখ্যার চেয়ে বড় বা সমান নাকি ছোট। তুমি মূলত সত্য না মিথ্যা উত্তর দিবে। উত্তরগুলোর ভিত্তিতে ক্রমলেখটি বলে দিবে তোমার ধরে নেওয়া সংখ্যাটি কতো? এই খেলায় আসলে প্রতিবার অর্ধেক করে পাল্লা কমাতে হয়। তাই এটাকে দ্বিখন্ডন পদ্ধতি (bisection method) বলা হয়। এই পদ্ধতিটিকে অনেক সময় দুয়িক অনুসন্ধান (binary search) ও বলা হয় অবশ্য।

পরিগণনা সমাধান: এবার আমরা পরিগণনার সমস্যাগুলোর সমাধান দেখবো। মনে রাখবে সমাধানগুলো দেয়া হয়েছে তুমি যদি একান্তই নিজে নিজে করতে পারছোনা বলে মনে হয়, তখন কেবল একটু সাহায্য যাতে পেতে পারো তাই। কাজেই পারতপক্ষে সমাধান দেখবে না।

১. নীচের নকশার মতো নকশা তৈরী করো। এই নকশার কোনার বিন্দুগুলোতে + আছে, একদম বাম আর ডান পাশে আছে |, আর অন্য সবগুলো হলো −, প্রতিটি সারিতে − আছে ২০টি করে। প্রত্যেক সারির −গুলোর জন্য তোমাকে একটি করে ঘূর্ণী (loop) লিখতে হবে।

```
+--------------------+
|--------------------|
|--------------------|
+--------------------+
```

এই ক্রমলেখ (program) রচনা করা খুবই সহজ। আমাদের চারটি সারির জন্য চারটি জন্য ঘূর্ণী (for loop) লাগবে। প্রত্যেক সারির শুরু ও শেষে সংশ্লিষ্ট বিশেষ চিহ্নগুলো দিতে হবে। আর ঘূর্ণী লাগবে মাঝখানের – চিহ্ন বারবার লেখার জন্য।

```
cout << "+";                    // উপরে বাম কোনা
for(int i = 0; i < 20; ++i)
  cout <<"-";                   // প্রথম সারি মাঝ
cout << "+" << endl;            // উপরে ডান কোনা
cout << "|";                    // দ্বিতীয় সারি শুরু
for(int i = 0; i < 20; ++i)
  cout <<"-";                   // দ্বিতীয় সারি মাঝ
cout << "|" << endl;            // দ্বিতীয় সারি শেষ
cout << "|";                    // তৃতীয় সারি শুরু
for(int i = 0; i < 20; ++i)
  cout <<"-";                   // তৃতীয় সারি মাঝ
cout << "|" << endl;            // তৃতীয় সারি শেষ
cout << "+";                    // নীচে বাম কোনা
for(int i = 0; i < 20; ++i)
  cout <<"-";                   // চতুর্থ সারি মাঝ
cout << "+" << endl;            // উপরে ডান কোনা
```

২. এমন একটি ক্রমলেখ (program) রচনা করো যেটি একটি ধনাত্মক (positive) পূর্ণক (integer) যোগান (input) নিয়ে সেটা মৌলিক (prime) সংখ্যা কিনা নির্ণয় করবে।

এই ক্রমলেখটি (program) নানান ভাবে করা যেতে পারে। আমরা প্রথমে সবচেয়ে সহজটি কিন্তু সবচেয়ে ধীর গতির উপায়টি দেখি। একটি সংখ্যা n মৌলিক কিনা সেটার পরীক্ষা হলো একটি ঘূর্ণী (loop) চালিয়ে ২ থেকে শুরু করে n – 1 পর্যন্ত প্রতিটি দিয়ে n বিভাজ্য কিনা পরীক্ষা করে দেখো। যদি একটি দিয়েও বিভাজ্য হয় তাহলে n মৌলিক নয়, আর সেক্ষেত্রে ঘূর্ণী আর চালানো দরকার নেই, ক্ষান্তি (break) দিয়ে বের হয়ে আসতে হবে। আর ঘূর্ণী যদি শেষ পর্যন্ত চলে, মানে ঘূর্ণীর সূচকের (loop index) মান যদি n হয়, তাহলে n মৌলিক। নীচে ক্রমলেখটি (program) দেখো।

```
int n, k;                  // n মুল সংখ্য, k সূচক
cout << "nombor: ";
cin >> n;

if (n <= 0)                // ঋণাত্মক কিনা পরীক্ষা
{
  cout << "rinatok" << endl;
  return EXIT_FAILURE;
}

// ২ থেকে n–1 পর্যন্ত কোনটি দিয়ে বিভাজ্য কিনা
```

```
for(k = 2; k < n; ++k)
    if (n % k == 0)
        break;          // বিভাজ্য হলে আগেই ক্ষান্তি

if (k == n)             // শেষ পর্যন্ত ঘূর্ণী চলেছে
    cout << "moulik hoy" << endl;
else                    // আগেই বের হয়ে এসেছে
    cout << "moulik noy" << endl;
```

একটু খেয়াল করলেই বুঝবে কোন সংখ্যা মৌলিক কিনা তার জন্য আসলে ২ থেকে n পর্যন্ত পরীক্ষা করা দরকার নেই। আসলে n/2 পর্যন্ত অথবা আরো ভালো করে বলতে গেলে n এর বর্গমূল পর্যন্ত পরীক্ষা করলেই চলে। কাজেই উপরের ক্রমলেখটি চাইলে আমরা আর একট দক্ষ করে লিখতে পারি। নীচে আমরা কেবল পরিবর্তন সংশ্লিষ্ট অংশ দেখালাম।

ফিরিস্তি ৮.১০: মৌলিক সংখ্যা কিনা নির্ণয় (Whether a Number is Prime)

```
for(k = 2; k < sqrt(n); ++k)     // sqart(n) পর্যন্ত
    if (n % k == 0)
        break;          // বিভাজ্য হলে আগেই ক্ষান্তি

if (k >= sqrt(n))       // শেষ পর্যন্ত ঘূর্ণী চলেছে
    cout << "moulik hoy" << endl;
else                    // আগেই বের হয়ে এসেছে
    cout << "moulik noy" << endl;
```

৩. এমন একটি ক্রমলেখ (program) লিখো যেটি দুটো ধনাত্মক পূর্ণক (integer) যোগান নিয়ে তাদের গসাগু (HCF) ও লসাগু (LCM) নির্ণয় করে।

দুটি সংখ্যা a ও bএর গসাগু হলো এমন একটি সংখ্যা g যেটি দ্বারা a ও b উভয় সংখ্যা বিভাজ্য হয়। এই রকম একাধিক সংখ্যা থাকলে সবচেয়ে বড়টি হবে গসাগু। গসাগু বের করা হয়ে গেলে আমরা a ও b এর গুণফল কে গসাগু দিয়ে ভাগ করে লসাগু পেতে পারি। তো এই ক্রমলেখ (program) লিখতে আমরা ১ থেকে শুরু করে প্রতিটি সংখ্যা দিয়ে ভাগ করে দেখবো a ও b উভয় সংখ্যা বিভাজ্য কিনা। যদি বিভাজ্য হয় তাহলে ভাজকটি আমাদের গসাগু হতে পারে, আর গুণফলকে গসাগু দিয়ে ভাগ করে লসাগু পেতে পারি। তবে আমাদের এখানেই থেমে গেলে হবে না, কারণ এর চেয়ে বড় কোন সংখ্যা সাধারণ ভাজক হিসাবে পাওয়া যায় কিনা তা দেখতে হবে। তবে একটা বিষয় মনে রাখতে হবে গসাগু g কখনই a বা b কোনটার চেয়েই বড় হবে না, দুটোর চেয়েই ছোট হবে।

```
cout << "nombor duti: ";     // যোগান যাচনা
int a, b;                    // চলক দুটি
cin >> a >> b;               // যোগান নেওয়া

if (a < 0 || b < 0)          // ঋণাত্মক কিনা?
{
  cout << "rinatok" << endl;
```

```
    return EXIT_FAILURE;
  }

  int p = a * b, g, l;     // গুণফল, গসাগু ও লসাগু

  // একে একে পরীক্ষা করো উভয়ে বিভাজ্য কিনা
  for(int k = 1; k <= a && k <= b; ++k)
    if (a % k == 0 && b % k == 0) // উভয়ে বিভাজ্য
      { g = k; l = p/k; }

  cout << "gosagu = " << g << endl;
  cout << "losagu = " << l << endl;
```

চাইলে উপরের ক্রমলেখকে আর একটু দক্ষ করতে পারো। যেহেতু গসাগু সংখ্যা দুটোর কোনটা থেকেই বড় হয়, কাজেই আমরা সংখ্যা দুটোর ছোটটি থেকে ঘূর্ণী (loop) শুরু করতে পারি। আর দুটোকে ভাগ করা যায় এমন সবচেয়ে বড় ভাজকটি যেহেতু আমাদের দরকার, আমরা তাই ঘূর্ণীটি ছোট থেকে শুরু করে বড়র দিকে চালাবো, আর প্রথমটি পাওয়া মাত্র ঘূর্ণী থেকে বের হয়ে আসবো।

```
  // চলক ঘোষণা, যোগান নেওয়া ও ঋণাত্মক পরীক্ষণ এখানে করো
  int p = a * b, g, l;     // গুণফল, গসাগু ও লসাগু

  int m = a > b ? b : a;   // দুটোর মধ্যে ছোটটি
  // একে একে পরীক্ষা করো উভয়ে বিভাজ্য কিনা
  for(int k = m; k; ++k)   // m হতে যতক্ষণ শুন্য নয়
    if (a % k == 0 && b % k == 0) // উভয়ে বিভাজ্য
      { g = k; l = p/k; break}

  cout << "gosagu = " << g << endl;
  cout << "losagu = " << l << endl;
```

আরো এক ভাবে যেমন ক্রমাগত ভাগের মাধ্যমেও আমরা গসাগু নির্ণয় করতে পারি। প্রথমে একটি সংখ্যাকে ভাজক আর আরেকটিকে ভাজ্য ধরে নিয়ে ভাগশেষ বের করবো। তারপর আগের ভাজকটি হয়ে যাবে নতুন ভাজ্য আর ভাগশেষটি নতুন ভাগশেষ। তারপর আবার ভাগ ও ভাজকটিকে নতুন ভাজ্য, ভাগশেষকে নতুন ভাজক । এই করে চলবে যতক্ষণ ভাগ-শেষ শুন্য না হচ্ছে। আর সেই মুহূর্তের ভাজকটিই হবে গসাগু।

ফিরিস্তি ৮.১১: গসাগু ও লসাগু নির্ণয় (Determining HCF and LCM)

```
  int t, g, l;     // সাময়িক, গসাগু, লসাগু
  int p = a * b;   // গুণফল

  do
  {
    t = a % b;     // ভাগশেষ নির্ণয়
```

```
    a = b;          // আগের ভাজক হবে নতুন ভাজ্য
    b = t;          // ভাগশেষটি হবে নতুন ভাজক
  }
  while(t);         // ভাগশেষ শুন্য হলে শেষ

  g = a, l = p/a; // গসাগু ও লসাগু

  cout << "gosagu = " << g << endl;
  cout << "losagu = " << l << endl;
```

৪. নীচের ক্রমলেখয়ের (program) ফলন (output) কী হবে, গণনিতে (computer) না চালিয়ে বের করো। তারপর গণনিতে চালিয়ে তোমার হিসাব করা ফলাফল যাচাই করো। যদি কোন অসীম ঘূর্ণী (infinite loop) থেকে থাকে সেটাকে মেরামতো করো।

```
  int n = 3;
  while (n >= 0)     // প্রথম ঘূর্ণী
  {
    cout << n * n << " ";
    --n;
  }
  cout << n << endl;

  while (n < 4)      // দ্বিতীয় ঘূর্ণী
    cout << ++n << " ";
  cout << n << endl;

  while (n >= 0)     // তৃতীয় ঘূর্ণী
    cout << (n /= 2) << " ";
  cout << endl;
```

উপরের ক্রমলেখয়ের (program) ফলন (output) নীচে দেখানো হলো। প্রথম ঘূর্ণী 3 থেকে শুরু করে 0 পর্যন্ত সংখ্যাগুলোর বর্গ দেখাবে। কাজেই 9 4 1 0 ফলনে আসবে, তারপর প্রথম ঘূর্ণীর (loop) ঠিক পরের cout এর কারণে আসবে −1। দ্বিতীয় ঘূর্নী 4 হওয়ার আগে পর্যন্ত প্রতিবার এক বাড়িয়ে সংখ্যাটি ফলন দেখাবে। কাজেই আমরা পাবো 0 1 2 3 4, দ্বিতীয় ঘূর্ণীর ঠিক পরের coutএর কারণে 4 আরো একবার আসবে। তারপর তৃতীয় ঘূর্ণীতে nএর মান শুন্য বা বেশী হলে আগে 2 দিয়ে ভাগ করবে তারপর ফলন দিবে। তো 4 হতে শুরু করলে আমরা ফলনে পাবো 2 1 0 কিন্তু একবার শুন্য হওয়ার পরে তারপর প্রতিবার 2 দিয়ে ভাগ করলেও nএর মান শুন্যই থাকবে। কাজেই ঘূর্ণীর (loop) শর্ত কখনো মিথ্যা হবে না। কাজেই আমরা একের পর এক অসীম সংখ্যক বার শুন্য পেতে থাকবো। অর্থাৎ এটি একটি অসীম ঘূর্ণী (infinite loop) হয়ে যাবে।

```
9 4 1 0 −1
0 1 2 3 4 4
2 1 0 0 0 0 ....
```

অসীম ঘূর্ণী ঠিক করতে চাইলে আমরা তৃতীয় ঘূর্ণীর শর্তটি n >= 0 বদলে n > 0 লিখে দিতে পারি। তাতে তৃতীয় ঘূর্ণীর কারণে ফলন আসবে 2 1 0।

৫. একজন অনভিজ্ঞ ক্রমলেখক নীচের ক্রমলেখটি (program) লিখেছে। ক্রমলেখটির ছাড়ন (indentation) দেখে যেমন মনে হচ্ছে ক্রমলেখটি ঠিক তেমন ফলন (output) দিচ্ছে না। ক্রমলেখক চেয়েছিলেন ১০ থেকে শুরু করে প্রতিবার ২ দিয়ে ভাগ করবেন আর ভাগফলের বর্গ দেখাবেন। ভাগ করতে গিয়ে শুন্য হয়ে গেলে থেমে যাবেন। সুতরাং তার কাঙ্খিত ফলন হচ্ছে 25 4 1 কিন্তু ক্রমলেখটি হতে সেরকম ফলন আসছে না। তো তুমি প্রথমে এই ক্রমলেখ যেমন আছে তেমন রেখেই এর ফলন নির্ণয় করো। আর সেক্ষেত্রে ছাড়ন কেমন হবে সেটাও দেখাও। তারপর কাঙ্খিত ফলাফল পেতে গেলে ক্রমলেখতে কী পরিবর্তন করতে হবে সেটাও করে দেখাও।

```
int n = 10;
while (n > 0)
  n /= 2;
  cout << n * n << " ";
cout << endl;
```

উপরের ক্রমলেখটিতে ছাড়ন দেখে মনে হয়ে ঘূর্ণীর (loop) পরের দুই সারি ঘূর্ণীর আওতার মধ্যে। কিন্তু গঠনরীতি অনুযায়ী আসলে তা হবে না, কারণ এখানে বক্র বন্ধনী দেয়া নেই। ফলে কেবল n /= 2টাই ঘূর্ণীর আওতায়। কাজেই ঘূর্ণী চলবে ঠিকই, প্রতিবার 2 দিয়ে ভাগ হবে, আর শুন্য হলে ঘূর্ণী থেমে যাবে। তারপর ঘূর্ণীর বাইরে থাকা coutএর কারণে আমরা 0এর বর্গ 0ই ফলনে পাবো। ফলে ফলন হবে কেবল 0। আর এই ক্ষেত্রে ছাড়নের বিষয়টি ঠিকঠাক করলে ক্রমলেখ দেখতে হবে নীচের মতো।

```
int n = 10;
while (n > 0)
  n /= 2;                 // কেবল এটি ঘূর্নীর ভিতরে
cout << n * n << " "; // ছাড়ন ঠিক করা হলো
cout << endl;
```

এবার কাঙ্খিত ফলন (output) পেতে গেলে আমাদের আসলে ক্রমলেখতে বক্র বন্ধনী (curly brackets) { } ব্যবহার করে coutটাকেও ঘূর্নীর একটি মহল্লা (block) তৈরী করে তার ভিতরে আনতে হবে। সুতরাং সেইক্ষেত্রে ক্রমলেখটি হবে নীচের মতো।

```
int n = 10;
while (n > 0)
{                                 // মহল্লা শুরু
  n /= 2;                         // ঘূর্নীর ভিতরে ছিলোই
  cout << n * n << " ";           // ঘূর্ণীর ভিতরে এখন
}                                 // মহল্লা শেষ
cout << endl;
```

৬. নীচের ক্রমলেখটি (program) কী করবে বর্ণনা করো। তারপর এটিকে এমন ভাবে আবার লিখো যাতে এতে ক্ষণ ঘূর্ণীর (while loop) বদলে করো ঘূর্ণী (do loop) ব্যবহৃত হয়, কিন্তু সব মিলিয়ে ক্রমলেখয়ের বৈশিষ্ট্য একই থাকে।

```
int n;
cout << "dhonatok sonkhya: ";
cin >> n;

while (n <= 0)
{
  cout << "dhonatok noy." << endl;
  cout << "dhonatok sonkhya: ";
  cin >> n;
}
```

উপরের ক্রমলেখটি (program) ধনাত্নক সংখ্যা দরকার এরকম যোগান যাচনা (input prompt) করে nএর মান যোগান (input) নিবে। তারপর n যদি ধনাত্নক না হয় তাহলে ঘূর্ণীর (loop) ভিতরে ঢুকবে আর বার্তা (message) দেখাবে ধনাত্নক নয়, আর আবার যোগান যাচনা করে nএর মান যোগান নিবে। তারপর ঘূর্নীর ভিতরে আবার পরীক্ষা করবে অধনাত্নক কিনা, এবং এই ভাবে চলতে থাকবে যতক্ষণ না nএর মান ধনাত্নক হচ্ছে। সব মিলিয়ে বলা যায়, কমপক্ষে একবার যোগান যাচনা দিয়ে n এর মান যোগান নেওয়া হবে: ঘূর্ণীর বাইরের যোগান যাচনা (input prompt) ও যোগান (input) নেওয়াটা হলো সেটি। সুতরাং আমরা খুব সহজেই করো ঘূর্নী (do loop) ব্যবহার করতে পারি এখানে।

```
int n;
do
{
  cout << "dhonatok sonkhya: ";
  cin >> n;
  if (n <= 0)
    cout << "dhonatok noy." << endl;
}
while (n <= 0);
```

তুমি চাইলে নীচের মতো করেও লিখতে পারো, যেখানে আমরা ধনাত্নক হলে বরং ঘূর্ণী (loop) থেকে ক্ষান্তি (break) নিবো। আর সেক্ষেত্রে অবশ্য while(n <= 0) না লিখে আমরা কেবল while(true) ও লিখতে পারি। আবার চাইলে করো ঘূর্নী (do loop) থেকে ক্ষণ ঘূর্ণীতে (while loop) ফেরতও যেতে পারি, যেখানে নীচের while(true) টাকে সরিয়ে নিয়ে গিয়ে do এর বদলে বসিয়ে দিবো।

```
int n;
do
{
  cout << "dhonatok sonkhya: ";
  cin >> n;
```

```
    if (n > 0) break;
    cout << "dhonatok noy." << endl;
  }
  while (n <= 0);
```

৭. নীচের ক্রমলেখটির (program) ফলন কী? এটিকে এমন ভাবে বদলে লেখো যাতে মহল্লা (block) ব্যবহার না করেই একই ফলাফল পাওয়া যায়। তারপর ক্রমলেখটিকে ক্ষণ ঘূর্ণী (while loop) ব্যবহার না করে জন্য ঘূর্ণী (for loop) ব্যবহার করে লিখো।

```
int i = 5;
while (i > 0)
{
  i = i + 1;
  cout << i << endl;
}
```

উপরের ক্রমলেখতে (program) ঘূর্ণীর ভিতরে iএর মান আগে কমানো হচ্ছে তারপর সেটা ফলনে (output) দেখানো হচ্ছে। কাজটি আমরা ফলনে দেখানোর সময়েই করতে পারি পূর্ব বৃদ্ধি (pre increment) ব্যবহার করে, যা নীচের ক্রমলেখতে দেখানো হলো।

```
int i = 5;
while (i > 0)
  cout << --i << endl;
```

তুমি চাইলে জন্য ঘূর্ণী (for loop) ব্যবহার করে নীচের মতো করেও লিখতে পারো।

```
for (int i = 5; --i; )
  cout << --i << endl;
```

৮. এমন একটি ক্রমলেখ রচনা করো যেটি একটি ঘূর্ণীর (loop) ভিতরে ব্যবহারকারীর কাছে থেকে একের পর এক একটি করে পূর্ণক (integer) যোগান নিবে। যোগান নেওয়া সংখ্যাটি ধনাত্মক না হলে ক্রমলেখ থেকে বের হয়ে যাবে, আর ধনাত্মক হলে মানের ক্রমানুসারে সংখ্যাটির উৎপাদকগুলোকে পরপর এক সারিতে ফলন (output) দিবে, আর পরের সংখ্যা যোগান নিতে চাইবে। নমুনা যোগান-ফলন (sample input output) নিম্নরূপ:

```
> 0 utpadok <= 0 shesh
sonkhya koto? 36
utpadok talika: 36 18 12 9 4 3 2 1
> 0 utpadok <= 0 shesh
sonkhya koto? -1
kromolekho shesh!
```

আমরা এখানে একটা অসীম ঘূর্ণী (infinite loop) নিবো শুরুতে while(true) লিখে, তার মানে ঘূর্ণীর ভিতরে আমাদের অবশ্যই একটা ক্ষান্তি (break) দিতে হবে। তো নীচের ক্রমলেখতে (program) দেখো আমরা ঘূর্ণীর ভিতরে যথাযথ যোগান যাচনা (input

prompt) দিয়ে সংখ্যাটি যোগান নিয়েছি। তারপর সংখ্যাটি ধনাত্মক না হলে ক্ষান্তি দিয়েছি, আর সেক্ষেত্রে ঘূর্ণীর বাইরে "ক্রমলেখ শেষ!" বার্তা (message) দেখিয়েছি। আর সংখ্যাটি ধনাত্মক হলে আমরা বড় থেকে ছোটর দিকে প্রতিটি সংখ্যা দিয়ে প্রদত্ত সংখ্যাটিকে ভাগ করেছি। ভাগশেষ শুন্য হওয়া মানে ভাজকটি একটি উৎপাদক, সেটি ফলনে (output) দেখাতে হবে। তুমি চাইলে এখানে করো ঘূর্ণী (do while) ব্যবহার করতে পারতে, আমরা সেটি তোমার নিজের চেষ্টার ওপরে ছেড়ে দিলাম, চেষ্টা করে দেখো।

ফিরিস্তি ৮.১২: উৎপাদক তালিকা দেখাও (Display List of Factors)

```
while(true)
{
  cout << "> 0 utpadok <= 0 shesh" << endl;
  int;  cin >> n;          // যোগান

  if (n <= 0) break;       // ঋনাত্মকে ক্ষান্তি

  cout << "utpadok talika: ";
  for(int k = n; k > 0; --k)
    if (n % k == 0)
      cout << " " << k;
  cout << endl;
}
cout << "kromolekho shesh!" << endl;
```

৯. এমন একটি ক্রমলেখ (program) রচনা করো যেটি একটি ধনাত্মক পূর্ণ সংখ্যা যেমন 23154 যোগান (input) নিয়ে ফলন দিবে 45132।

```
int n, r, t; // নম্বর, উল্টা, সাময়িক
cout << "nombor? ";
cin >> n;
if (n <= 0)
{
  cout << "dhonatok noy" << endl;
  return EXIT_FAILURE;
}

r = 0;  // শুরুতে উল্টা নম্বর শুন্য
while(n > 0)
{
  t = n % 10;    // এককের অঙ্ক
  r = r * 10 + t; // উল্টার পিছে
  n = n/10;      // অবশিষ্ট অংশ
}

cout << "ulta = " << r << endl;
```

উপরের ক্রমলেখতে প্রথমে উল্টা নম্বর r শুন্য ধরে নিয়েছি। তারপর n যতক্ষণ শুন্য না হচ্ছে ততক্ষণ ঘূর্ণী (loop) চালানো হবে। প্রতিবার nএর যে এককের অঙ্ক আছে সেটি নিয়ে r পিছে লাগিয়ে দিতে হবে। n এর এককের অংক পাওয়া যায় ১০ দিয়ে ভাগ করে ভাগশেষ নিলে। এখন rএ যা আছে তার পিছনে ওই অঙ্কটি লাগাতে হলে rএর মানকে আগে ১০ দিয়ে গুণ করে নিতে হবে কারণ এগুলো তে বামের দিকে এক ঘর সরে যাবে, আর তাতে ডানের যে স্থানটি ফাঁকা হলো সেখানে ওই অঙ্কটি বসিয়ে দিতে হবে, অর্থাৎ যোগ করতে হবে। ঘূর্ণীর পরের পাকের জন্যে n হবে আগের পাকের এককের অঙ্ক ছাড়া বাঁকী অংশ, আর সেটি পাওয়া যাবে ১০ দিয়ে n কে ভাগ করে।

১০. এমন একটি ক্রমলেখ (program) রচনা করো যেটি একে একে সংখ্যা যোগান (input) নিবে যতক্ষণ ধনাত্নক সংখ্যা দেওয়া হচ্ছে, অধনাত্নক সংখ্যা হলে ক্রমলেখ শেষ হবে। ক্রমলেখটির ফলন (output) হবে যোগান নেয়া সংখ্যাগুলোর মধ্যে সবচেয়ে বড়টি আর সেটি কত নম্বরে যোগান দেওয়া হয়েছিলো সেই ক্রমিক নম্বরটি।

```
int boro = 0, suchok = 0; // শুরুতে দুটোই শুন্য

int k = 0;        // ক্রমিক গোনার জন্য
while(true)
{
  cout << "nombor? ";    // যোগান যাচনা
  int n; cin >> n;       // যোগান নেওয়া

  if (n <= 0) break;     // অধনাত্নক সংখ্যা

  k = k + 1;          // আর একটি যোগান হলো

  if (n > boro)    // এটি আগের বড়র চেয়েও বড়
  {
    boro = n;      // এটি তাই নতুন বড়
    suchok = k;    // আর নতুন বড়টির সূচক
  }
}

if (k > 0)  // যদি কোন নম্বর যোগান হয়ে থাকে
{
  cout << "boro = " << boro << endl;
  cout << "suchok = " << suchok << endl;
}
```

উপরের ক্রমলেখটির (program) শুরুতে আমরা বড় সংখ্যা হিসাবে আদিতে ধরে নিয়েছি শুন্য, যেটি যোগান দেওয়া যে কোন ধনাত্নক সংখ্যার চেয়ে ছোট হবে। অনেকগুলো সংখ্যার মধ্যে সবচেয়ে বড় সংখ্যাটি বের করতে চাইলে আমরা সাধারণত শুরুতে ছোট একটা সংখ্যাকে ফলাফল হিসাবে ধরে নেই। যাতে সেটার চেয়ে তুলনা করে করে আরো বড় আরো বড় সংখ্যা পাওয়া যায়। তুমি যদি অনেকগুলো সংখ্যার মধ্যে সবচেয়ে ছোট সংখ্যাটি বের করতে চাও তাহলে তোমাকে শুরুতে বড় একটা সংখ্যাকে ফলাফল হিসাবে ধরে নিতে হবে।

যাইহোক এরপর উপরের ক্রমলেখটি দেখো যোগান যাচনা করে যোগান নিয়ে প্রথমে পরীক্ষা করেছে ধনাত্মক কিনা। ধনাত্মক না হলে ঘূর্ণীতে (loop) ক্ষান্তি (break) দিতে হবে আর না হলে যেহেতু আরেকটি ধনাত্মক সংখ্যা পাওয়া গেলো তাই ক্রমিক নম্বর এক বাড়বে। এরপর বর্তমানের বড়টি সাথে তুলনা করে যদি দেখা যায় নতুন নম্বরটি বড়, তাহলে নতুন নম্বরটিই হবে বড় আর তার সূচকটি আরেকটি চলকে নিতে হবে। ঘূর্ণীর বাইরে কেবল বড় সংখ্যাটি আর তার সূচক ফলনে (output) যাবে।

১১. ফিবোনাস্যি (Fibonacci) প্রগমন হলো ০, ১, ১, ২, ৩, ৫, ৮, ...। লক্ষ্য করো এই প্রগমনের প্রথম দুটি পদ হলো ০ আর ১। আর এর পর থেকে প্রতিটি পদ তার আগের দুটো পদের যোগফল। এমন একটি ক্রমলেখ (program) রচনা করো যেটি একটি ধনাত্মক পূর্ণক n যোগান (input) নিয়ে nতম পদ নির্ণয় করবে।

ফিরিস্তি ৮.১৩: ফিবোনাস্যি প্রগমন নির্ণয় (Fibonacci Progression)

```
int n; // মান তুমি যোগান নিয়ে ধনাত্মক কিনা পরীক্ষা করবে।

// প্রথম দুটো সংখ্যা আর তারপরেরটির চলক
int prothom = 0, ditiyo = 1, tarpor;

if (n == 1) // প্রথমটি সরাসরি ফলন
  cout << "prothom = " << prothom << endl;
else if (n == 2)  // দ্বিতীয়টি সরাসরি ফলন
  cout << "ditiyo = " << ditiyo << endl;
else  // অন্য যে কোনটা ক্রমাগত হিসাব করতে হবে
{
  for(int k = 3; k <= n; ++k)
  {
    // k–তম পদ হিসাব করা হবে
    tarpor = prothom + ditiyo;  // যোগ
    prothom = ditiyo; // পরেরটার জন্য ২য়টাই ১ম
    ditiyo = tarpor;  // পরেরটার জন্য নতুনটা ২য়
  }
  // নতুন যেটি সেটি তমk পদ, সুতরাং ফলন
  cout << n << "tom = " << tarpor << endl;
}
```

প্রথমে n চলক ঘোষণা variable declare) করে, তুমি যোগান যাচনা (input prompt) দিয়ে nএর মান যোগান নিবে। তারপর n ধনাত্মক কিনা পরীক্ষা করে দেখবে। n ধনাত্মক না হলে একটি ঘূর্ণীর (loop) ভিতরে আবার যোগান নিতে পারো যতক্ষণ না ধনাত্মক মান দেওয়া হচ্ছে এই অংশটুকু করে দেওয়া হলো না, নিজের করো। আমরা কেবল ফিবোনাস্যি পদগুলো নির্ণয়ের অংশটুকু দেখি। প্রথম দুটি পদ ধারণ করার জন্য আমাদের prothom আর ditiyo নামে দুটি আর পরের পদের জন্য tarpor নামে আরেকটি চলক (variable) আছে। যদি nএর মান ১ বা ২ হয় তাহলে আমরা তো প্রথম পদ দুটি থেকে কোনরূপ হিসাব করা ছাড়া সরাসরিই ফলন দিতে পারি। আর যদি nএর মান 3 বা বেশী হয় তাহলে একটি জন্য ঘূর্ণীতে (for loop) আমরা প্রথমে tarpor পদটি হিসাব করবো prothom ও

ditiyo পদ দুটি যোগ করে। এবার তারওপরের পদটির জন্য আমাদের যা করতে হবে এই পাকে সেটা ঠিক করে রাখতে হবে, যাতে পরের পাকের শুরুতেই আমরা ওই পদটি হিসাব করতে পারি। তো tarpor হিসাব করার পরে এখন আমাদের প্রথম পদটি হবে আগে যেটি দ্বিতীয় পদ ছিলো সেটি, আর দ্বিতীয় পদটি হবে tarpor পদটি। এবং তারফলেই পরের পাকে prothom ও ditiyo যোগ করলে আমরা তারওপরের পদটি পাবো। ঘূর্ণী (loop) চলবে এখানে ৩ থেকে n হওয়া পর্যন্ত। খেয়াল করো kএর মান যত আমরা ঘূর্ণীর (loop) মহল্লার ভিতরে ততোম পদটি হিসাব করছি, আর তারপরের পদটির জন্য প্রস্তুতি নিচ্ছি।

১২. একটি ধনাত্মক পূর্ণক n যোগান নিয়ে প্রথম n স্বাভাবিক সংখ্যার (১, ২, ৩, ...) যোগফল ও গুণফল নির্ণয়ের ক্রমলেখ (program) রচনা করো। তুমি হয়তো জানো প্রথম n স্বাভাবিক সংখ্যার গুণফলকে উৎপাদকীয় (factorial) বলা হয়। উৎপাদকীয় খুবই বড় সংখ্যা হয় যা intএ নাও ধরতে পারে, কাজেই n বড় হলে আমরা উল্টাপাল্টা ফল পেতে পারি।

```
int n; // মান তুমি যোগান নিয়ে ধনাত্মক কিনা পরীক্ষা করবে।

// নীচের সারিতে যোগফল কেন ০ আর গুণফল ১?
int jogfol = 0, gunfol = 1;
for(int k = 1; k <= n; ++k)
{
  jogfol += k;     // যোগ করো
  gunfol *= k;     // গুণ করো
}
cout << "jogfol = " << jogfol << endl;
cout << "gunfol = " << gunfol << endl;
```

এই ক্রমলেখটি (program) খুবই সহজ। আমরা n চলক (variable) ঘোষণা দেখিয়েছি, কিন্তু এর যোগান যাচনা (input prompt) ও যোগান (input) নেওয়া তুমি করবে। তারপর সেটি ধনাত্মক কিনা সেটিও পরীক্ষা করবে। এর পরে খেয়াল করো আমরা দুটি চলক (variable) নিয়েছি যোগফল ও গুণফলের জন্যে। মজার ব্যাপার হচ্ছে jogfolএর আদি মান (initial value) দিয়েছি ০ কিন্তু gunfolএর আদি মান দিয়েছি ১। কেন বলতে পারবে? যোগের মানে আগে থেকে ১ থাকলে তো যোগফল সঠিক আসবে না, ১ বেশী আসবে, তাই আদি মান শুন্য। আর গুণফলের ক্ষেত্রে আদি মান ০ হলে এরপরের সকল গুণফলই তো ০ হয়ে যাবে, ১ দিলে সেটি হবে না। যোগ ও গুণের জন্য আদি মানের এই তফাৎ সবসময় মনে রাখবে। এই ক্রমলেখয়ের বাঁকী অংশটুকুতো আর ব্যাখ্যা করছি না।

১৩. নীচের ধারাগুলোর প্রথম n পদের সমষ্টি নির্নয় করো। তোমার ক্রমলেখতে (program) তুমি n যোগান (input) হিসাবে নিবে, আর ধারাটির সমষ্টি ফলন (output) দিবে।

ক) লম্বানুপাত ধারা (sine series): $\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$

খ) লগ্নানুপাত ধারা (cosine series): $\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$

গ) ভাজিত ধারা (harmonic series): $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$

ঘ) অয়লার সংখ্যা (Euler number): $e_n = 1 + \frac{1}{1!} + \frac{1}{2!} + \ldots + \frac{1}{(n-1)!}$

ঙ) গুণোত্তর ধারা (geometric series): $\frac{1}{1-x} = 1 + x + x^2 + \ldots + x^{n-1}$

এখানে আমরা শুধু $\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$ এর ক্রমলেখ (program) দেখাবো। বাঁকীগুলো তুমি একই রকম করে নিজে করো। আমাদের nএর পাশাপাশি xও যোগান (input) নিতে হবে। তবে n যেখানে ধনাত্মক পূর্ণক (positive integer) x সেখানে ভগ্নক (fractioner)। তারপর আমাদের একটি ঘূর্ণী (loop) নিতে হবে যেটি 1 থেকে n পর্যন্ত ঘুরবে, আর প্রতিপাকে kতম পদটি হিসাব করে যোগফলের সাথে যোগ করে দিবে। একটু খেয়াল করলে দেখবে এখানে kতম পদটি আসলে $-\frac{(-x)^k}{k!}$। সুতরাং আমরা পদ নির্ণয়ের জন্য একটি ঘূর্ণী (loop) চালিয়ে শক্তি (power) $(-x)^k$ নির্ণয় করবো, আরেকটি ঘূর্ণী (loop) চালিয়ে উৎপাদকীয় (factorial) $k!$ নির্ণয় করবো, আর তারপর পদটি যোগ করে দেবো যোগফলের সাথে।

```
int n;    // যোগান নাও
float x;  // যোগান নাও

float jogfol = 0; // যোগফল
for(int k = 1; k <= n; ++k)
{
  float shokti = 1;    // (-x)^k
  float utpadokio = 1;  // k!

  for(int p = 1; p <= k; ++p)
  {
    shokti *= (-x); // শক্তি নির্ণয়
    utpadokio *= p; // উৎপাদকীয়
  }
  jogfol += - shokti / utpadokio;
}

cout << jogfol << endl;
```

উপরের ক্রমলেখটি আসলে দক্ষ হয় নি। কারণ এতে ঘূর্ণীর ভিতরে ঘূর্নী বা অন্তান্তি ঘূর্ণী (nested loop) ব্যবহৃত হয়েছে। আসলে ভিতরের ঘূর্ণীটা ব্যবহার না করেই এই ক্রমলেখ (program) লেখা সম্ভব। একটা ব্যাপার খেয়াল করো আগের পদের সাথে আমরা কেবল $frac-x^2(2k-1)(2k-2)$ গুণ করলেই পরের পদ পাবো। কাজেই আগের পদ আরেকটা চলক (variable) ব্যবহার করে মনে রাখলে পরের পদ সহজে বের করা যাবে, এবং কোন ঘূর্ণী (loop) না চালিয়েই তা বের করা সম্ভব হবে।

```
int n;    // যোগান নাও
float x;  // যোগান নাও

float jogfol = 0; // যোগফল
float pod = x;
for(int k = 1; k <= n; ++k)
{
  jogfol += x;
```

```
    pod *= -x*x/((2*k-1)*(2*k - 2));
  }

  cout << jogfol << endl;
```

১৪. দশটা সংখ্যা যোগান (input) নিয়ে তাদের গড় (mean) ও প্রমিত বিচ্যুতি (standard deviation) নির্ণয় করো। প্রমিত বিচ্যুতি হলো সংখ্যাগুলোর বর্গের গড় থেকে গড়ের বর্গ বিয়োগ করে বিয়োগফলের বর্গমূল অর্থাৎ $\sqrt{\frac{\sum x^2}{n} - (\frac{\sum x}{n})^2}$।

```
int n; // কয়টি সংখ্যা যোগান নাও
// নীচের চলক দুটি যোগফল ও বর্গের যোগফল
float sumx = 0, sumx2 = 0

for(int k = 0; k < n; ++k)
{
  cout << k << "tom pod? "; // যাচনা
  float x; cin >> x;        // যোগান

  sumx += x;          // যোগফল নির্ণয়
  sumx2 += x*x;       // বর্গের যোগফল
}

float mean = sumx / n;     // গড়, নীচে প্রমিত বিচ্যুতি
float std = sqrt(sumx2 / n - mean * mean);
```

উপরের ক্রমলেখটি (program) খুবই সাধারণ। একটি ঘূর্ণীর ভিতরে যোগফল ও বর্গের যোগফল বের করে নাও। তারপর ঘূর্ণীর বাইরে প্রথমে গড় আর তারপর প্রমিত বিচ্যুতি নির্ণয় করো। এখানে sqrt() বিপাতক (function) ব্যবহার করে বর্গমূল নির্ণয় করা হয়েছে। কাজেই ক্রমলেখয়ের শুরুতে #include <cmath> লিখে cmath শির নথি (header file) অন্তর্ভুক্ত (include) করে নিতে হবে।

১৫. একটি শ্রেণীতে n সংখ্যক শিক্ষার্থী আছে আর তাদের প্রত্যেকে m সংখ্যক বিষয়ে পরীক্ষা দিয়েছে। প্রত্যেক ছাত্রের প্রত্যেক বিষয়ের নম্বর যোগান (input) নিয়ে প্রত্যেক ছাত্রের মোট নম্বর ফলন (output) দাও।

```
int m, int n; // যোগান নাও

// প্রত্যেক ছাত্রের জন্য ঘূর্ণী
for(int i = 1; i <= n; ++i)
{
  float jogfol = 0;

  // প্রত্যেক বিষয়ের জন্য ঘূর্ণী
```

```
    for(int j = 1; j <= m; ++j)
    {
      float nombor;
      cout << j << "th bishoy? ";
      cout << nombor;

      jogfol += nombor;
    }
    cout << "mot nombor = " << jogfol;
    cout << endl;
  }
```

এই ক্রমলেখতে (program) দুটো অন্তান্তি ঘূর্ণী (nested loop) লাগবে। প্রথমটি প্রতি শিক্ষার্থীর জন্য, আর দ্বিতীয়টি তাদের প্রতিটি বিষয়ের জন্য। বাঁকী অংশ সহজ, দেখে নাও।

১৬. এক পরীক্ষায় মোট নম্বর ৯। কাজেই একজন ছাত্র ০ হতে ৯ পর্যন্ত যে কোন নম্বর পেতে পারে। এমন একটি ক্রমলেখ (program) রচনা করো যেটি এইরূপ ২০ জন ছাত্রের প্রত্যেকের নম্বর যোগান (input) নিয়ে তাদের নম্বর তারকা চিহ্ন দিয়ে আনুভূমিক চিত্রে দেখাবে। তিনজন ছাত্রের জন্য নমুনা যোগান ও ফলন (input and output) নিম্নরূপ:

```
1tom nombor? 9
1: ********* (9)
2tom nombor? 5
2: ***** (5)
3tom nombor? 6
3: ****** (6)
```

এখানেও আমদের অন্তান্তি ঘূর্ণী (nested loop) লাগবে। প্রথম ঘূর্ণীটি প্রতিটি ছাত্রের জন্য আর ভিতের একটা ঘূর্ণী লাগবে দরকার মতো তারকা দেখানোর জন্য।

```
  // প্রতিটি ছাত্রের ঘূর্ণী
  for(int k = 1; k <= 20; ++k)
  {
    // যাচনা করে নম্বর যোগান
    cout << k << "tom nombor? ";
    int nombor; cin >> nombor;

    // শুরুতে ক্রমিক নম্বর
    cout << k << ": ";

    // তারকা দেখানোর ঘূর্ণী
    for(i = 1; i <= nombor; ++i)
      cout << "*";

    // শেষে বন্ধনীতে নম্বর দেখানো
```

```
    cout << " (" << nombor << ")";
    cout << endl;
  }
```

১৭. ০ থেকে ৯ পর্যন্ত নামতার সারণী (table) লিখার জন্য একটি ক্রমলেখ রচনা করো। সারণীকে সুন্দর করে সাজানোর জন্য তোমাকে এক অঙ্কের সংখ্যা ও দুই অঙ্কের সংখ্যা মাথায় রাখতে হবে। এক অঙ্কের সংখ্যার জন্য তুমি প্রথমে একটি অতিরিক্ত ফাঁকা (space) দিয়ে নিবে। তাতে সারণীতে আড়ি ও খাড়ি (row and column) ঠিক মতো থাকবে। আর নামতার সারণীতে সবচেয়ে বামের খাড়িতে (column) আর উপরের আড়িতে (row) অবশ্যই ০ থেকে ৯ পর্যন্ত সংখ্যাগুলো থাকবে শিরোনাম হিসাবে।

এই ক্রমলেখটি একটু খুঁটিনাটি খেয়াল করে লিখতে হবে, যাতে সারণীটি আসলেই সুন্দর লাগে দেখতে। সর্বপ্রথমে সারণীর উপরের বাম কোনা খেয়াল করো সেটি অবশ্যই ফাঁকা হবে। আর প্রথম সারিতে আসলে সংখ্যাগুলো থাকবে ০ থেকে ৯ পর্যন্ত শিরোনাম হিসাবে। দ্বিতীয় সারি থেকে মূলত গুণফলগুলো থাকবে। তবে প্রথমেই থাকবে শিরনাম হিসাবে সংখ্যা, আর তারপর গুণফলগুলো। গুণফল দেখানোর আগে অবশ্যই খেয়াল করবে ৯ বা ছোট কিনা, সেক্ষেত্রে একটা অতিরিক্ত ফাঁকা (space) দেখাতে হবে।

```
  cout << " ";        // উপরে বাম কোনা

  // প্রথম সারি হলো শির নাম
  for(int i = 0; i <= 9; ++i)
    cout << "  " << i;  // দুইটা ফাঁকা
  cout << endl;

  // প্রতিটি সংখ্যার জন্য সারি
  for(int i = 0; i <= 9; ++i)
  {
    cout << i;  // প্রথম খাড়ি

    // প্রতিটি সংখ্যার জন্য খাড়ি
    for(int j = 0; j <= 9; ++j)
    {
      int p = i * j;  // গুণফল

      cout << " ";     // পৃথকী
      if (p <= 9)      // এক অঙ্কের
        cout << " ";   // অতিরিক্ত ফাঁকা
      cout << p;       // গুণফল
    }
    cout << endl;      // সারি শেষ
  }
```

১৮. এমন একটি ক্রমলেখ (program) রচনা করো যেটি ৫০ জন ছাত্রের নম্বর যোগান (input)

নিবে, আর ফলন দিবে কোন পাল্লায় কতজন ছাত্রের নম্বর পড়েছে তা। প্রতি ১০ নম্বরের জন্য একটি করে পাল্লা চিন্তা করতে পারো, আর মোট নম্বর হলে ৩০ যেখানে এক জন ছাত্র ১ থেকে ৩০ পর্যন্ত যে কোন নম্বর পেতে পারে। কাজেই মোটা পাল্লা রয়েছে ৩টি।

```
int prothom = 0, ditiyo = 0, titiyo = 0;

for(int k = 1; k <= 50; ++k)
{
  cout << k << "tom nombor? ";
  int nombor; cin >> nombor;

  if (nombor <= 0)
    cout << "pallar baire" << endl;
  else if (nombor <= 10)
    ++prothom;
  else if (nombor <= 20)
    ++ditiyo;
  else if (nombor <= 30)
    ++tritiyo;
  else
    cout << "pallar baire" << endl;
}

cout << "prothom = " << prothom << endl;
cout << "ditiyo = " << ditiyo << endl;
cout << "tritiyo = " << tritiyo << endl;
```

১৯. তারকা ব্যবহার করে নীচের বিভিন্ন রকম আকৃতিগুলো ফলন দাও। প্রতিটি ক্ষেত্রে তুমি দরকার মতো তুমি পরামিতি (parameter) n যোগান (input) নিবে। পরামিতি মানে হচ্ছে সারির সংখ্যা বা সারিতে সর্বোচ্চ কয়টি তারা থাকবে বা থাকবে না এগুলো n ওপর নির্ভরশীল। নীচের প্রতিটি ক্ষেত্রে nএর মান 5।

```
*                  *             *
**                **            ***
***              ***           *****
****            ****          *******
*****          *****         *********


*              *                 *
**             **               ***
***            ***             *****
****           ****           *******
*****          *****         *********
****            ****          *******
```

```
***          ***          *****
**            **           ***
*              *            *
```

এই সব আকৃতির প্রতিটির জন্য আমাদের অন্তান্তি ঘূর্ণী (nested loop) লাগবে। আমরা এখানে কেবল উপরের ডানপাশেরটি করে দিবো। বাঁকীগুলো তুমি নিজে করবে।

```
int n = 5; // যোগান নিতে পারো

// প্রতিটি আড়ির (row) জন্য ঘূর্ণী
for(int ari = 1; ari <= n; ++ari)
{
  // শুরুতে ফাঁকা আছে n - ari সংখ্যক
  for(int khari = 1; khari <= n - ari; ++khari)
    cout << "-";
  // মাঝখানে তারা আছে 2 * ari - 1 সংখ্যক
  for(int khari = 1; khari <= 2*ari - 1; ++khari
 )
    cout << "*";
  // শেষে ফাঁকা আছে n - ari সংখ্যক
  for(int khari = 1; khari <= n - ari; ++khari)
    cout << "-";
  cout << endl;
}
```

এই ক্রমলেখ (program) লেখার সুবিধার্থে আমরা প্রথমে ফলনটাকে (output) একটু বদলে লিখি। মূলত ফাঁকাগুলোকে (space) নীচের ডানপাশের মতো করে - বসিয়ে চিন্তা করি। তাহলে ফাঁকা গুনতে সুবিধা হওয়ায় নকশাটির ধাঁচ (pattern) পাওয়া সহজ হবে, ফলনের সময় আমরা ফাঁকাই লিখবো - নয়। মনে রাখবে এখানে আমাদের পরামিতি ৫। এখানে ৫ টি সারি আছে। প্রতিটি সারিতে তারা অক্ষর আছে ৯টি মানে 2*5 - 1। আরো খেয়াল করো ১ম সারিতে শুরু ও শেষে ফাঁকা আছে ৪টি করে, ২য় সারিতে ৩টি করে, ৩টি সারিতে ২টি করে। এসবের প্রতিটি ক্ষেত্রে যোগফল পাঁচ মানে ১ + ৪ = ২ + ৩ = ৩ + ২ = ৪ + ১। অর্থাৎ যততম সারি, শুরু ও শেষে ফাঁকার সংখ্যা ৫ থেকে তত বিয়োগ করলে পাওয়া যাবে। এরপর দেখো তারকার সংখ্যা হলো পরপর সারিতে ১, ৩, ৫, ৭, ৯, অর্থাৎ যততম সারি তার দ্বিগুণের চেয়ে এক কম সংখ্যক তারা আছে।

```
    *              ----*----
   ***             ---***---
  *****            --*****--
 *******           -*******-
*********          *********
```

২০. দ্বিপদী সহগ (binomial coefficient) হলো ${}^nC_r = \frac{n}{1} * \frac{n-1}{2} * \ldots * \frac{n-r+1}{r}$। দ্বিপদী সহগগুলো নিয়ে একটা ত্রিভুজ তৈরী করা যায় যাকে বলা হয় প্যাসক্যাল ত্রিভুজ। প্যাসক্যাল

ত্রিভুজ নীচে দেখানো হলো। প্রথম সারিতে মানটি হলো ${}^0C_0$, দ্বিতীয় সারিতে মানগুলো হলো ${}^0C_1$ ও ${}^1C_1$, তৃতীয় সারিতে ${}^0C_2$, ${}^1C_2$ ও ${}^2C_2$। পরামিতি n যোগান (input) নিয়ে তার জন্য প্যাসক্যাল ত্রিভুজ ফলন (output) দাও। নীচের প্যাসক্যাল ত্রিভুজের পরামিতি ৪। তুমি চাইলে উল্টিয়ে পাল্টিয়ে নানান রকমের প্যাসক্যাল ত্রিভুজ তৈরী করতে পারো।

```
    1                ———1———
   1 1               ——1–1——
  1 2 1              —1–2–1—
 1 3 3 1             –1–3–3–1–
1 4 6 4 1            1–4–6–4–1
```

আমরা আপাতত ধরে নেই প্রতিটি সংখ্যাতে একটাই অঙ্ক থাকবে। বেশী অঙ্কের সংখ্যার জন্য তোমাকে আর একটু কষ্ট করে অতিরিক্তি ফাঁকা (space) ব্যবহার করতে হবে। যা-ইহোক নকশার ধাঁচ বুঝার জন্য আমরা ফাঁকাগুলোতে সাময়িক ভাবে – বসিয়ে নিয়েছি। আমাদের এখানে nএর মান ৪। তবে হিসাবের সুবিধার্থে আমরা এখানে গণনা শুরু করবো ০ থেকে। কাজেই ০তম সারিতে শুরু ও শেষে ফাঁকা রয়েছে ৪টি। ১ম সারিতে ফাঁকা রয়েছে ৩টি। এইভাবে ৪র্থ সারিতে ফাঁকা শুন্যটি। অর্থাৎ যত নম্বর সারি ৪ বিয়োগ তত হলো শুরু ও শেষের ফাঁকার সংখ্যা। আর প্রতি দুটো সংখ্যার মাঝখানে একটা ফাঁকা আছে অথবা বলতে পারে প্রতিটি সারির প্রথম সংখ্যাটি ছাড়া পরের সংখ্যাগুলোর সামনে ফাঁকা আছে।

```
int n = 4;  // যোগান নিতে পারো, শুন্যও নেয়া যেতে পারে

// প্রতিটি সারির জন্য
for(int k = 0; k <= n; ++k)
{
  // সারির শুরুতে ফাঁকা
  for(int i = 0; i < n - k; ++i)
    cout << " ";

  // মাঝখানে সংখ্যাগুলো
  for(int i = 0; i <= k; ++i)
  {
    // দুটি সংখ্যার মধ্যবর্তী ফাঁকা
    if (i > 0)
      cout << " ";

    // এবার সংখ্যাটি C(k, i) নির্নয়
    int gunfol = 1;
    for(int j = 1; j <= i; ++j)
      gunfol = gunfol * (k - j + 1) / j;
      // উপরের সারিতে gunfol *= (k - j + 1) / j;
      // লিখলে ঠিক মতো ফলন আসবে না,
      // কারণ পূর্ণকের ভাগ সমস্যা করবে

    cout << gunfol;
```

```
    }

    // সারির শেষে ফাঁকা
    for(int i = 0; i < n - k; ++i)
      cout << " ";
    cout << endl;
  }
```

২১. এমন একটি ক্রমলেখ (program) রচনা করো যেটি তোমার সাথে একটি খেলা খেলবে। ক্রমলেখটি তোমাকে মনে মনে ০ থেকে ১০২৩ এর মধ্যে একটি নম্বর মনে মনে ধরতে বলবে। আর তারপর ক্রমলেখটি তোমাকে বেশ কিছু প্রশ্ন করবে, এই যেমন তোমার ধরেন নেওয়া সংখ্যাটি অমুক সংখ্যার চেয়ে বড় বা সমান নাকি ছোট। তুমি মূলত সত্য না মিথ্যা উত্তর দিবে। উত্তরগুলোর ভিত্তিতে ক্রমলেখটি বলে দিবে তোমার ধরে নেওয়া সংখ্যাটি কতো? এই খেলায় আসলে প্রতিবার অর্ধেক করে পাল্লা কমাতে হয়। তাই এটাকে দ্বিখন্ডন পদ্ধতি (bisection method) বলা হয়। এই পদ্ধতিটিকে অনেক সময় দুয়িক অনুসন্ধান (binary search) ও বলা হয় অবশ্য।

```
  int soto = 0, boro = 1024;

  cout << "mone mone ekta nombor nao" << endl;
  cout << "nomborti hote hobe 0—1023" << endl;
  cout << "nicher prosnogulor uttor dao" << endl;
  cout << "sotyo hole 1, mithya hole 0" << endl;

  do
  {
    float modhyo = (soto + boro) / 2.0;
    cout << "sonkhya >= " << modhyo << "? ";
    int uttor; cin >> uttor;

    if (uttor)
      soto = modhyo;
    else
      boro = modhyo - 1;
        cout << soto << " " << boro << endl;
  }
  while (soto < boro);
  cout << "sonkhya = " << soto << endl;
```

চলো আমরা উপরের ক্রমলেখটি (program) বিশ্লেষণ করি। আমরা দুটি সংখ্যা নিয়েছি soto আর boro যেখানে আমরা আমাদের পাল্লা রাখতে চাই। আমরা পাল্লা হিসাবে ধরে নিলাম ০ আর ১০২৪। তো প্রথমবারে আমরা তাহলে প্রশ্ন করবো ধরে নেওয়া সংখ্যাটি অর্ধেক অর্থাৎ ৫১২ এর চেয়ে বড় বা সমান কিনা। উত্তর যদি হ্যাঁ হয় তার মানে ধরে নেওয়া সংখ্যাটি সর্বনিম্ন ৫১২, কাজেই sotoকে আমরা বদলে করে ফেলতে পারি ৫১২। এতে

আমাদের পাল্লা অর্ধেক হয়ে গেলো। এবার ৫১২ ও ১০২৪ এর মাঝখানেরটি ৭৬৮ দিয়ে একই রকম প্রশ্ন করবো। উত্তর যদি না হয় তার মানে ধরে নেওয়া সংখ্যাটি সর্বোচ্চ ৭৬৭, কাজেই boroকে আমরা বদলে করে ফেলতে পারি ৭৬৭। এবারেও পাল্লা অর্ধেক হয়ে গেলো। এইভাবে চলতে থাকবে। উপরের ক্রমলেখ খেয়াল করো, দেখো এত ক্ষণ যা বললাম, তাই করেছি কি না? তাহলে হয়ে গেলো আমাদের মজার খেলা!

৮.২৪ গণনা পরিভাষা (Computing Terminologies)

- পুনালি (iterative)
- ঘূর্ণী (loop)
- সূচক (index)
- জন্য ঘূর্ণী (for loop)
- ক্ষণ ঘূর্ণী (while loop)
- করো ঘূর্ণী (do loop)
- পুনরাবৃত্তি (repetition)
- আদ্যায়ন (initialisation)
- হালায়ন (update)
- ডিঙানো (continue)
- পতাকা (flag)
- অসীম (infinite)
- মিথস্ক্রিয়া (interaction)

খন্ড ২

English (বাংলা)

অধ্যায় ৯

Program Writing in c++ (সিপিপিতে ক্রমলেখ রচনা)

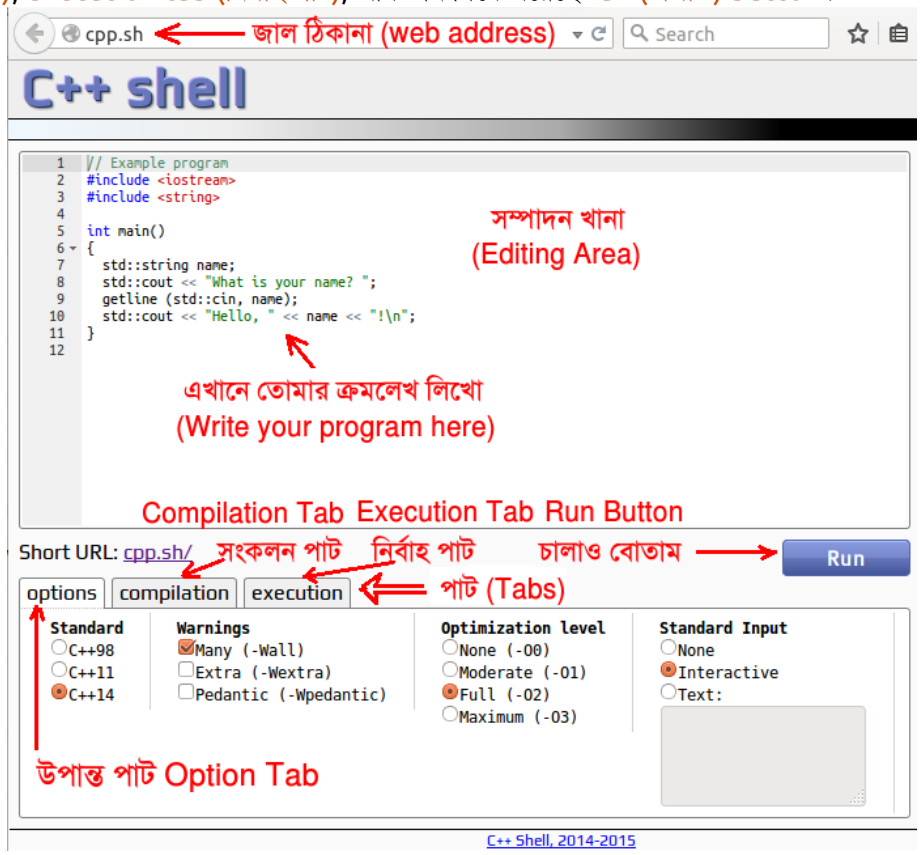
কোন programmerএর (পরিগণকের) কাছে নিজের লেখা program (ক্রমলেখ) একদম সন্তানের মতো। তিল তিল করে সময় নিয়ে programmer একটি program গড়ে তোলে। যে problemএর(সমস্যা) জন্য program তৈরী করতে হবে, সেটা জানার পরে programmer প্রথমে চিন্তা ভাবনা করে কী লিখবে, তারপর সেটা program লেখার যথাযথ নিয়ম মেনে লিখে ফেলে, তারপর সেটা চালিয়ে দেখে ঠিক ঠাক কাজ করে কি না। যদি ভুল কিছু থেকে থাকে, ভুলটা বের করে, সেটা ঠিক করে, তারপর আবার program চালিয়ে দেখে। এই চলতে থাকে যতক্ষণ না মন মতো problemটির solution পাওয়া যাচ্ছে। আমরা সারা বইতে পড়বো programএ কী লিখবো আমরা, আর যথাযথ ভাবে program রচনার নিয়মই বা কী। তবে এইখানে আলোচনা করবো, program (ক্রমলেখ) লিখবো কোথায় আর সেটা চালাবো কী করে।

Computerএ (গণনি) execute (নির্বাহ) করার জন্য আমরা যখন একটি program লিখতে চাই, তখন প্রথমে আমরা সেটা edit (সম্পাদনা) করি সাধারণত কোন একটা programming languageএ (পরিগণনা ভাষা)। এই programming language ঠিক computerএ executable (নির্বাহযোগ্য) language নয়, আবার ঠিক মানুষের natural languageও (স্বাভাবিক ভাষা) নয়, বরং এ দুটোর মাঝামাঝি কিছু একটা। Programmming languageএ লিখিত আমাদের programকে আমরা তাই এরপরে compile (সংকলন) করে যন্ত্রের পাঠোপযোগী machine languageএ (যন্ত্রভাষায়) রূপান্তর করি যাতে computer সেটা বুঝতে পারে। তারপর রূপান্তরিত programটিকে আমরা execute (নির্বাহ) করি।

৯.১ Online Software (হয়মান মন্ত্রপাতি)

Online (হয়মান) editing & compilation এর (সম্পাদনা ও সংকলন) জন্য আমরা cpp.sh নামক webpage (জালপাতা) ব্যবহার করবো। তুমি খুঁজলে এরকম আরো অনেক webpage পেতে পারো। যাই হোক তোমার internet browserএ (আন্তর্জাল বাজকে) web address (জাল ঠিকানা) লিখবার জায়গায় cpp.sh লিখে তুমি উপরে উল্লেখিত ওই webpage এ যেতে পারো। তারপর browser এ (ব্রাজকে) ওই webpage কেমন দেখা যাবে সেটা আমরা নীচের ছবিতে দেখতে পাবো। খেয়াল করো ওই ছবিতে বিভিন্ন অংশ তীর চিহ্ন দিয়ে চিহ্নিত করা হয়েছে। বড় সাদা অংশে editing area (সম্পাদন খানা) লেখা হয়েছে। Editing areaএর নীচে বাম

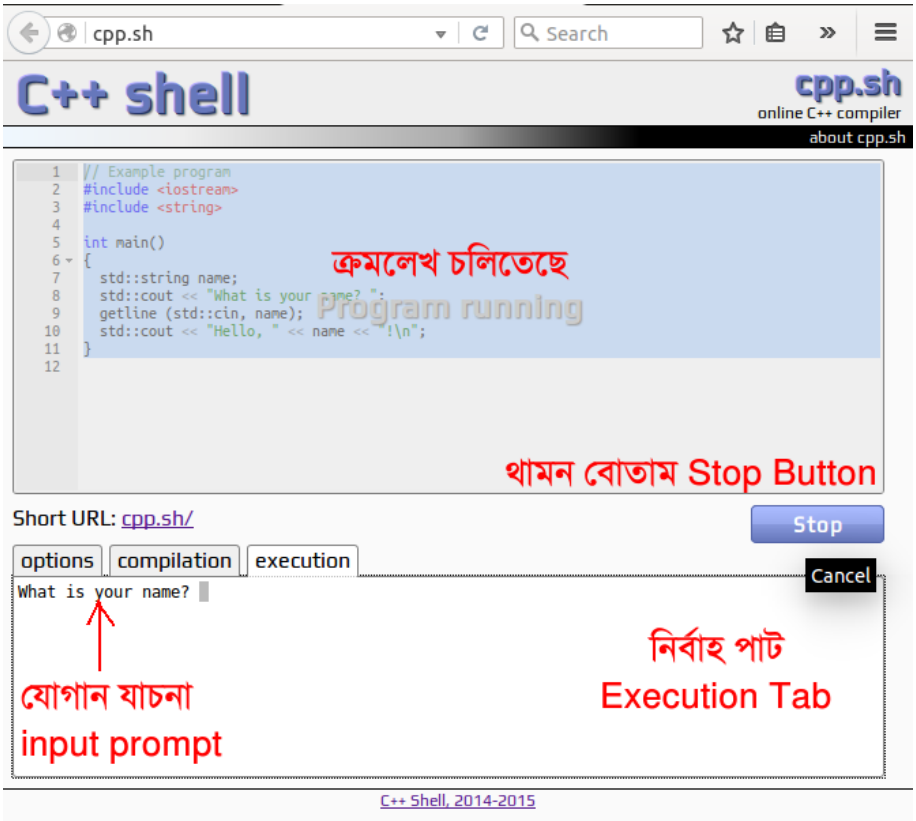
দিকে রয়েছে তিনটি tab (পাট): options tab (উপান্ত পাট), compilation tab (সংকলন পাট), execution tab (নির্বাহ পাট), আর ডান দিকে রয়েছে run (চালাও) button।



আমরা মুলত program writing ও editing করবো editing areaতে (সম্পাদন খানা)। Sample (নমুনা) হিসাবে editing areaতে আগে থেকে কিছু থাকতে পারে, তুমি সেগুলো মুছে দিতে পারো বা তোমার programএর জন্য দরকার মতো বদলে নিতে পারো। cpp.sh webpageএ (জালপাতা) গেলে সাধারণত নীচে দেখানো programটিই (ক্রমলেখ) সেখানে থাকে। আমরা আপাতত options tabএ (উপান্ত পাট) কোন পরিবর্তন না করে সরাসরি run buttonএ (চালাও বোতাম) click করে sample (নমুনা) programটিই run করবো।

```
// Example program
#include <iostream>
#include <string>

int main()
{
  std::string name;
  std::cout << "What is your name? ";
  getline (std::cin, name);
  std::cout << "Hello, " << name << "!\n";
}
```

Program editing শেষ হলে অথবা মাঝামাঝি অবস্থাতেও পরীক্ষা করে দেখার জন্য আমরা সাধারণত run buttonএ click (টিপ) করবো। তাতে এক clickএই প্রথমে program compile (সংকলন) হবে তারপর execution (নির্বাহ) হবে। যখন program compile হতে থাকবে তখন editing areaএর মাঝখানে দেখবে "Please Wait Compiling" "অপেক্ষা করো সংকলন হচ্ছে" লেখা আসবে। আর একই সাথে run buttonটি বদলে গিয়ে হয়ে যাবে cancel button (বাতিল বোতাম)। অনেক ক্ষেত্রে compile হতে সময় লাগে, তুমি যদি কোন কারণে compilation cancel করতে চাও তাহলে cancel buttonএ click করলেই হবে। যখন compile হতে থাকে তখন বাম দিকের tabগুলো (পাট) খেয়াল করবে, option tabএর (উপান্ত পাট) বদলে compilation tab (সংকলন পাট) সামনে চলে আসবে। এসময় সময় কোন error (ত্রুটি) পাওয়া গেলে compile tabএ দেখা যাবে। আর কোন compilation error না থাকলে Compilation successful message দেখা যাবে compilation tabএ আর তারপর execution tab (নির্বাহ পাট) সামনে আসবে। Execution timeএ input and output (যোগান ও ফলন) execution tabএ চলবে আর cancel buttonটি (বাতিল বোতাম) বদলে হয়ে যাবে stop button (থামন বোতাম), যাতে যে কোন সময় execution থামিয়ে দেয়া যায়। Stop button click করলে অথবা execution শেষ হয়ে গেলে আবার options tab (উপান্ত পাট) সামনে আসবে আর run button (চালাও বোতাম) ফিরে আসবে।

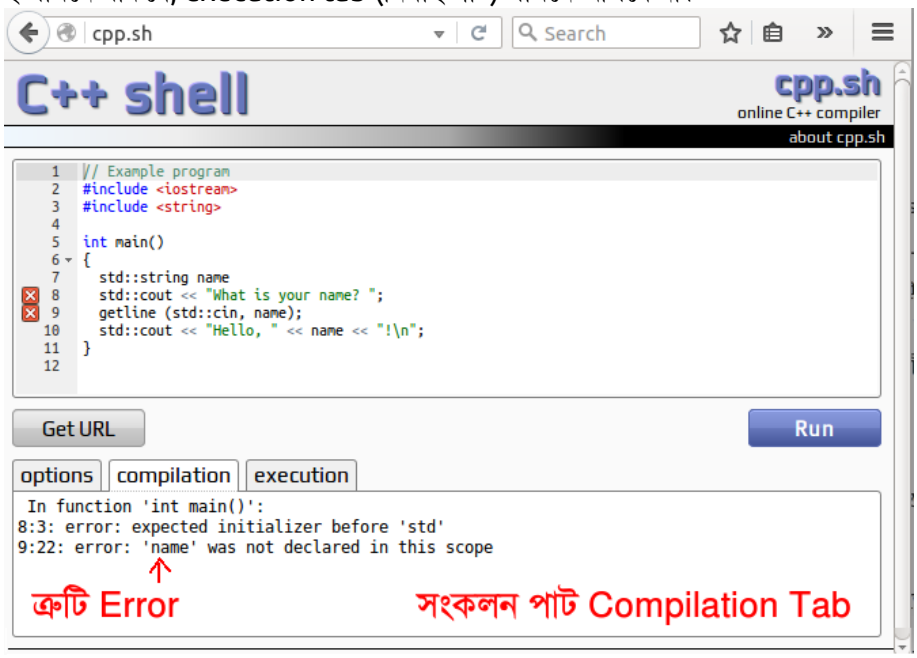


Sample programটি আমাদের আপাতত বিস্তারিত বুঝার দরকার নাই, আমরা পরে ব্যাপারগুলো বিস্তারিত শিখবো। তবে Sample programটি চালালে উপরের ছবির মতো প্রথমে execution tabএ (নির্বাহ পাট) দেখাবে What is your name? তখন তুমি যদি তোমার নাম লিখে দাও gonimia আর তারপর enter (ভুক্তি) চাপ দাও, তাহলে পরের সারিতে লেখা আসবে Hello, gonimia! নামটুকু নেওয়ার আগে What is your name? দেখানোকে আমরা বলি

input prompt (যোগান যাচনা) আর নাম gonimia দেওয়াটাকে আমরা বলি input (যোগান) দেওয়া আর পরের সারিতে Hello, gonimia! দেখানোকে আমরা বলি output (ফলন) দেওয়া। তো তুমি run (চালাও) buttonএ টিপ দিয়ে দেখো কী হয়। প্রথমে compilation tab হয়ে execution tabএ (নির্বাহ পাট) গিয়ে উপরে যে ভাবে বলা হলো সে রকম হয় কী না দেখো। তোমার বোঝার সুবিধার্থে execution tabএ শেষ পর্যন্ত কী থাকবে তা নীচে দেখানো হলো।

```
What is your name? gonimia
Hello, gonimia!
```

এবার আমরা একটু দেখি compilationএ (সংকলন) error হলে কী ঘটে, আর আমাদের কী করতে হয়! এটার জন্য আমরা ইচ্ছে করে একটা error (ত্রুটি) তৈরী করে দেই। যেমন ধরো std::string name; লেখা রয়েছে যে সারিতে সেখানে একদম শেষ হতে semicolon (দির্তি) ; তুমি মুছে দাও। আর তারপর run buttonএ (চালাও বোতাম) টিপ দাও। দেখবে নীচের মতো করে error message দেখাবে compilaiton tabএ (সংকলন পাট), আর compilation tabই সামনে থাকবে, execution tab (নির্বাহ পাট) সামনে আসবে না।



Compilation tabএ (সংকলন পাটে) যে messageগুলো আসবে তা নীচে দেখানো হলো। দ্বিতীয় সারিতে দেখো $8:3$ মানে বুঝাচ্ছে ৮ম সারিতে error আছে আর ৩য় অক্ষরে, আর errorটা হলো ; থাকতে হবে। আসলে ; দরকার আমাদের ৭ম সারির শেষে। সাধারণত যে সারিতে error আছে বলা হয়, error সেই সারি বা আগের সারিতে থাকে। এখানে ; থাকায় compiler (সংকলক) আসলে ঠিক ৭ম আর ৮ম সারি নিয়ে কিঞ্চিত বিভ্রান্তিতে রয়েছে। Program রচনার সময় আমরা নানান রকম ভুল ত্রুটি করি, তুমি program লেখার চর্চা করতে থাকলে এই ত্রুটিগুলোর সাথে পরিচিত হয়ে যাবে। তখন দেখা মাত্রই বুঝতে পারবে ভুলটুকু কী আর কী করে সেটা ঠিক করতে হবে। যাইহোক ত্রুটিটুকু বুঝতে পারলে আমরা সেটি ঠিক করে আবার run buttonএ টিপ দিবো, আর তখন programটি সফল ভাবেই execute হবে।

```
In function 'int main()':
8:3: error: expected initializer before 'std'
```

```
9:22: error: 'name' was not declared in this scope
```

সবশেষে আমরা options tab (উপান্ত পাট) সংক্ষেপে আলোচনা করবো। সেখানে থাকা নানা optionগুলোর (উপান্ত) কী কাজ মুলত সেটাই জানা আমাদের উদ্দেশ্য। তবে এগুলো নিয়ে আমরা আপাতত পরীক্ষা-নিরীক্ষা করবো না, বরং যে রকম অবস্থায় আছে সে রকম অবস্থাতেই program (ক্রমলেখ) editing (সম্পাদনা), compile (সংকলন) ও execute (নির্বাহ) করবো।

১. সবচেয়ে বামের columnএ (স্তম্ভে) দেখো standard (প্রমিত) optionগুলো রয়েছে। সিপিপি ভাষার নানান version (সংস্করণ) রয়েছে, তুমি চাইলে আগের version ব্যবহার করতে পারো, সাধারণত এখানে C++14 version নির্বাচন করা থাকে।

২. বাম থেকে দ্বিতীয় স্তম্ভে আছে warning messageএর (সতর্কবার্তার) optionগুলো, অর্থাৎ compile (সংকলন) করার সময় compiler (সংকলক) কতটা খুঁটি নাটি ত্রুটি ধরবে সেটা এখানে বলে দেয়া হয়। সাধারণত এখানে সব Many (-Wall) optionটি (উপান্ত) selected (নির্বাচিত) থাকে।

৩. বামথেকে তৃতীয় columnএ আছে optimisationএর (অনুকুলায়ন) optionগুলো। একই program (ক্রমলেখ) compiler (সংকলক) চাইলে এমন ভাবে compile (সংকলন) করতে পারে যে programটি অনেক দ্রুত execute (নির্বাহ) হবে, আবার সেটিই এমন ভাবে compile করতে পারে যে programটি অনেক ধীরে execute হবে। দ্রুত execute হবে এমন compile করতে স্বাভাবিক ভাবেই বেশী সময় লাগে, আর ধীরে execute হবে সেরকম compile করতে সময় কম লাগে। এখানে সাধারণত পূর্ণ Full (-O2) optionটি (উপান্ত) selected (নির্বাচিত) থাকে।

৪. সবচেয়ে ডানের columnএ আছে standard input (প্রমিত যোগান) optionসমুহ। সাধারণত এখানে interactive (মিথস্ক্রিয়) option (উপান্ত) select (নির্বাচিত) করা থাকে যার অর্থ keyboard (চাপনি) ব্যবহার করে input (যোগান) দেওয়া যাবে। তোমার programএ কোন input না থাকলে তুমি none (কিছুনা) option select করতে পারো। অথবা তুমি যদি আগেই input দিয়ে রাখতে চাও তাহলে text (পাঠনিক) option select করে ওইখানে থাকা বাক্সে আগে থেকে তোমার inputগুলো দিয়ে রাখতে পারো। তাতে program (ক্রমলেখ) keybaord (চাপনি) থেকে input না নিয়ে ওইখান থেকে নিয়ে নিবে। তোমাকে আর প্রতিবার input promptএ input দিতে হবে না।

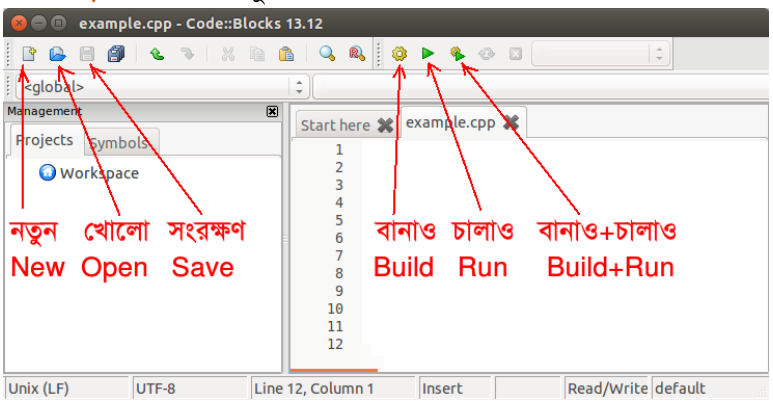
৯.২ Offline Software (নয়মান মন্ত্রপাতি)

কোডব্লকস (Code::Blocks) একটি offline software (নয়মান মন্ত্রপাতি) যেটি লিনাক্স, উইন্ডোজ, ম্যাক ওএস, সব operating systemএই (পরিচালনা তন্ত্র) ব্যবহার করা যায়। কোডব্লকসে তুমি c/c++ program editing and compilation (সম্পাদনা ও সংকলন) করতে পারবে। কোডব্লকস তোমার computerএ (গণনি) install (সংস্থাপন) করে নিলে তুমি internet (আন্তর্জাল) ব্যবহার করা ছাড়াই তোমার program compilation ও editing করে যেতে পারবে। কোডব্লকস পাওয়া যায় http://www.codeblocks.org/ webpage (জালপাতা), এর software তুমি http://www.codeblocks.org/downloads/binaries link হতে নামিয়ে নিতে পারো। তুমি উইন্ডোজ user হলে codeblocks-13.12mingw-setup.exe version নামিয়ে install (সংস্থাপন) করবে। এই versionএ GCC compiler (সংকলক)

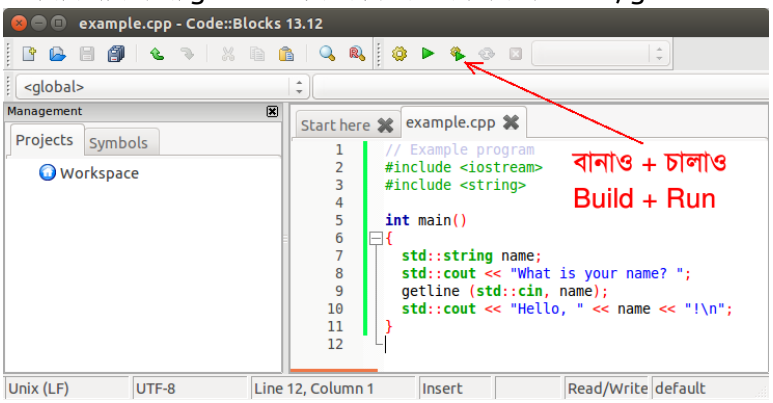
আর GDB debugger (আপদনাশক) আছে। তুমি লিনাক্স বা ম্যাক user হলে তোমার দরকারী version নামিয়ে installl (সংস্থাপন) করবে। Computerএ (গণনি) কোডব্লকস install (সংস্থাপন) বিষয়ে সাহায্য পেতে চাইলে নীচের প্রথম দুটি link হতে video (ছবিও) দেখতে পারো আর user manual (ব্যবহার পুস্তিকা) পেতে পারো তৃতীয় link হতে ।

Windows: https://www.youtube.com/watch?v=zOGU8fC3bvU
Linux: https://www.youtube.com/watch?v=3B4hPHZNtNw
User manual: http://www.codeblocks.org/user-manual

এরপরে তোমার computerএ (গণনি) operating system (পরিচালনা তন্ত্র) কী আছে তার জন্য দরকারী নির্দেশনা মেনে তুমি কোডব্লকস ঠিকঠাক মতো install (সংস্থাপন) করে নাও। Install হয়ে গেলে তারপর তুমি কোডব্লকস programটি (ক্রমলেখ) চালাও (run)। দেখবে নীচের মতো window (জানালা) খুলে যাবে। তারপর তুমি menu (প্রাপণ্য) থেকে File এর অধীনে Newএর (নতুন) ভিতরে Empty Fileএ (ফাঁকা নথি) click (টিপ) দাও। নতুন fileএ আপাতত কিছু থাকবে না। তারপর আবার menu (প্রাপণ্য) থেকে Save Fileএ click করে দরকার মতো file name (নথির নাম) যেমন example.cpp দিয়ে তোমার নতুন সৃষ্ট fileটিকে save (সংরক্ষণ) করো। Menu (প্রাপণ্য) থেকে এসব না করে নীচের ছবিতে দেখানো icon buttonগুলোতে (মূর্তি বোতাম) click করেও তুমি তোমার fileতৈরী ও তা নাম দিয়ে save করতে পারো।



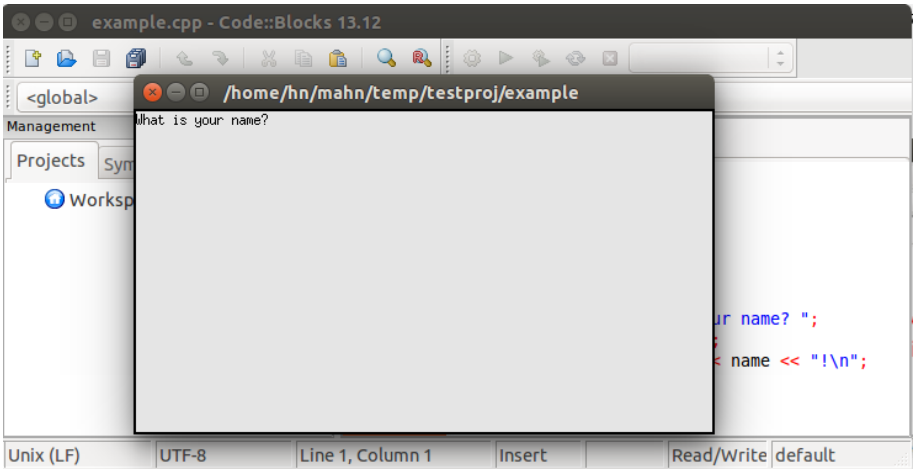
এরপর example.cppতে নীচের মতো code (সংকেত) লিখো। এই programটি আমাদের আপাতত বিস্তারিত বুঝার দরকার নাই, আমরা পরে ব্যাপারগুলো বিস্তারিত শিখবো। তবে সংক্ষেপে বলি এই program (ক্রমলেখ) তোমাকে জিজ্ঞেস করবে What is your name? আর তুমি তোমার নাম ধরো gonimia লিখে দিলে তখন বলবে Hello, gonimia!



উপরের ছবিতে লক্ষ্য করো আমরা যে program লিখেছি সেটাতে আসলে নীচের codeগুলোই লিখেছি। এটি ঠিক online software (হয়মান মন্ত্রপাতি) হিসাবে cpp.sh webpage (জালপাতা) ব্যবহার করে যে program লিখেছিলাম সেটিই। কোডব্লকসে sample হিসাবে আগে থেকে এই রকম ক্রমলেখ থাকে না, তোমাকে নিজে এটা লিখে নিতে হবে। তারপর উপরের ছবিতে দেখানো build+run (বানাও+চালাও) buttonএ click করে তুমি programটি চালাবে। Build+run buttonএ click করলে আসলে প্রথমে তোমার লেখা program compile (সংকলন) হয়ে executable (নির্বাহযোগ্য) program তৈরী হয়, আর তারপর সেই executable program আসলে run করে।

```
// Example program
#include <iostream>
#include <string>

int main()
{
  std::string name;
  std::cout << "What is your name? ";
  getline (std::cin, name);
  std::cout << "Hello, " << name << "!\n";
}
```

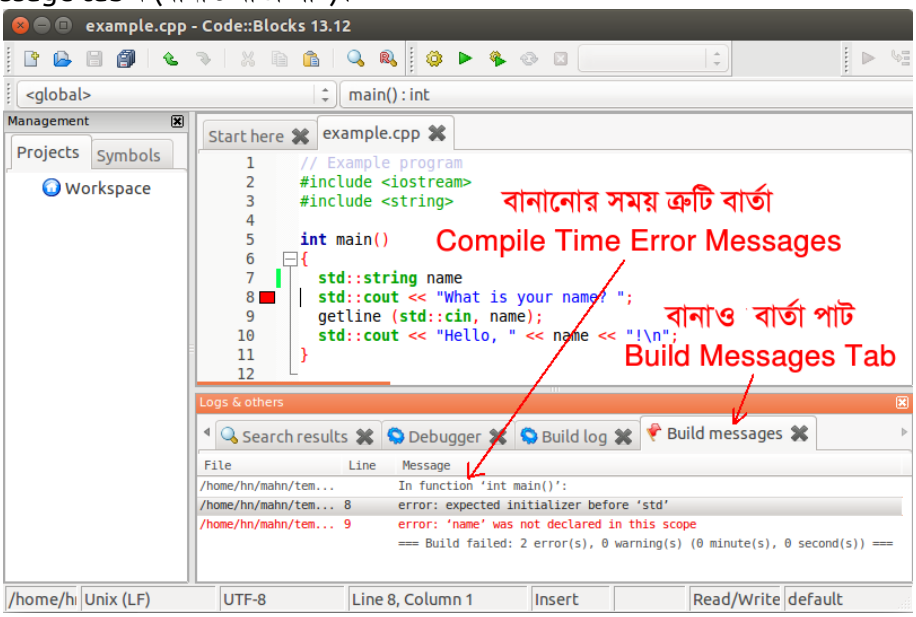
তো build+run (বানাও+চালাও) buttonএ click দিলে সাধারণত নীচের ছবির মতো করে একটা অতিরিক্ত window (জানালা) আসবে। আর তাতে লেখা থাকবে What is your name ? তখন তুমি যদি তোমার নাম লিখে দাও gonimia আর তারপর enter (ভুক্তি) চাপ দাও, তাহলে পরের সারিতে দেখবে লেখা আসবে Hello, gonimia!। নামটুকু নেওয়ার আগে What is your name? দেখানোকে আমরা বলি input prompt (যোগান যাচনা) আর নাম gonimia দেওয়াটাকে আমরা বলি input (যোগান) দেওয়া আর পরের সারিতে Hello, gonimia! দেখানোকে আমরা বলি output (ফলন) দেওয়া।



তো তুমি Build+Run buttonএ (বানাও+চালাও বোতাম) click করে দেখো কী হয়। উপরে যে ভাবে বলা হলো সে রকম হয় কী না দেখো। তোমার বোঝার সুবিধার্থে execution (নির্বাহ) শেষে ওই অতিরিক্ত windowতে কী থাকবে তা নীচে দেখানো হলো।

```
What is your name? gonimia
Hello, gonimia!
```

এবার আমরা দেখবো কোডব্লকসে লেখা আমাদের programএ যদি কোন ভুল থাকে তাহলে build+run (বানাও+চালাও) buttonএ click দিলে কী ঘটবে? Build+run buttonএ click দেওয়ার আগে তোমাকে menu (প্রাপণ্য) থেকে Viewএর (দৃষ্টি) অধীনে Logsএ (ঘটচা) টিপ দিতে বলবো, অথবা F2 key (চাপনি) চাপলেও একই কাজ হবে। এর ফলে নীচের ছবিতে দেখানো build messages tabএর (বানাও বার্তা পাট) দেখা যাবে, যেখানে আসলে compilationএ (সংকলনে) কোন error (ত্রুটি) থাকলে তা দেখানো হবে। এবার আমরা ইচ্ছে করে একটা error (ত্রুটি) তৈরী করে দেই। যেমন ধরো std::string name; লেখা রয়েছে যে সারিতে সেখানে একদম শেষ হতে semicolon (দির্তি) ; তুমি মুছে দাও। আর তারপর build+run buttonএ (বানাও+চালাও বোতাম) click করো। দেখবে error message (ত্রুটি বার্তা) দেখাবে build message tabএ (বানাও বার্তা পাট)।



Build messages tabএ (বানাও বার্তা পাট) যে messageগুলো আসবে তা নীচে দেখানো হলো। দ্বিতীয় সারিতে দেখো 8 মানে বুঝাচ্ছে ৮ম সারিতে ত্রুটি আছে, আর ত্রুটিটা হলো ; থাকতে হবে। আসলে ; দরকার আমাদের ৭ম সারির শেষে। সাধারণত যে সারিতে ত্রুটি আছে বলা হয়, ত্রুটি সেই সারি বা আগের সারিতে থাকে। এখানে ; থাকায় compiler (সংকলক) আসলে ঠিক ৭ম আর ৮ম সারি নিয়ে কিঞ্চিত বিভ্রান্তিতে রয়েছে। Program রচনার সময় আমরা নানান রকম ভুল ত্রুটি করি, তুমি program লেখার চর্চা করতে থাকলে এই ত্রুটিগুলোর সাথে পরিচিত হয়ে যাবে। তখন দেখা মাত্রই বুঝতে পারবে ভুলটুকু কী আর কী করে সেটা ঠিক করতে হবে। যাইহোক ত্রুটিটুকু বুঝতে পারলে আমরা সেটি ঠিক করে আবার build+run buttonএ (বানাও+চালাও বোতাম) টিপ দিবো, আর তখন program (ক্রমলেখ) সফল ভাবেই নির্বাহিত হবে।

```
In function 'int main()':
8: error: expected initializer before 'std'
9: error: 'name' was not declared in this scope
```

৯.৩ গণনা পরিভাষা (Computing Terminologies)

- Programmer (পরিগণক)
- Programming (পরিগণনা)
- Language (ভাষা)
- Natural (স্বাভাবিক)
- Program (ক্রমলেখ)
- Programmer (ক্রমলেখক)
- Computer (গণনি)
- Execution (নির্বাহ)
- Executable (নির্বাহযোগ্য)
- Edit (সম্পাদনা)
- Compile (সংকলন)
- Online (হয়মান)
- Offline (নয়মান)
- Software (মন্ত্র, মন্ত্রপাতি)
- Webpage (জালপাতা)
- Internet (আন্তর্জাল)
- Browser (ব্রাজক)
- Web address (জাল ঠিকানা)
- Editing area (সম্পাদন খানা)
- Tab (পাট)
- Option (উপান্ত)
- Options tab (উপান্ত পাট)
- Compilation tab (সংকলন পাট)
- Execution tab (নির্বাহ পাট)
- Run (চালাও)
- Build (বানাও)
- Sample (নমুনা)
- Click (টিপ)
- Button (বোতাম)
- Cancel (বাতিল)
- Input (যোগান)
- Output (ফলন)
- Stop (থামন)
- Enter (ভুক্তি)
- Prompt (যাচনা)
- Error (ত্রুটি)
- Column (স্তম্ভ)
- Standard (প্রমিত)
- Optimisation (অনুকুলায়ন)
- Interaction (মিথস্ক্রিয়া)
- Interactive (মিথস্ক্রিয়)
- Keyboard (চাপনি)
- None (কিছুনা)
- Text (পাঠনিক)
- Operating system (পরিচালনা তন্ত্র)
- Install (সংস্থাপন)
- Debugger (আপদনাশক)
- Video (ছবিও)
- User manual (ব্যবহার পুস্তিকা)
- Window (জানালা)
- Menu (প্রাপণ্য)
- File (নথি)

৯.৩. গণনা পরিভাষা (Computing Terminologies)

- Empty file (ফাঁকা নথি)
- New (নতুন)
- Save (সংরক্ষণ)
- Icon (মূর্তি)
- Code (সংকেত)
- Semicolon (দির্তি)
- View (দৃষ্টি)
- Log (ঘটচা)

অধ্যায় ১০

Program Structure (ক্রমলেখয়ের কাঠামো)

Computerএ (গণনিতে) executable (নির্বাহযোগ্য) এক গুচ্ছ instructionএর (নির্দেশ) sequenceকে (ক্রম) program (ক্রমলেখ) বলা হয়। আমরা সিপিপি (c++) languageএ program তৈরী করবো। Program সাধারণত একটি editor (সম্পাদনা) software (মন্ত্র) ব্যবহার করে তৈরী করা হয়। আমরা একাজে আপাতত cpp.sh নামের একটি webpage (জালপাতা) ব্যবহার করবো। সিপিপি languageএ তৈরী programকে প্রথমে একটি compiler (সংকলক) দিয়ে compile (সংকলন) করে computerএ executable code (সংকেত) তৈরী করা হয়। তারপর সেই code run (চালানো) করলে বা execute (নির্বাহ) করলে আমরা সাধারণত consoleএর (যন্ত্রালয়ের) monitorএ (নজরিতে) output (ফলন) দেখতে পাই । Program অনেক সময় আমাদের কাছ থেকে consoleএর keyboard (চাপনি), mouseএর (টিপনি) মাধ্যমে input (যোগান) নিতে পারে। আসলে console (যন্ত্রালয়) বলতে inputএর (যোগান) জন্য keyboard and mouse (চাপনি ও টিপনি) আর outputএর (ফলন) জন্য monitor (নজরি) বুঝানো হয়। Program লিখতে গেলে console (যন্ত্রালয়) থেকে input (যোগান) নেয়ার ও consoleএ (যন্ত্রালয়ে) output (ফলন) দেখানোর কথা তুমি প্রায়শই শুনতে পাবে। কাজেই এগুলো কী বুঝায় সেটা ভালো করে মনে রেখো।

১০.১ Wishing Program (শুভেচ্ছা বার্তার ক্রমলেখ)

সিপিপি (c++) languageএ এমন একটি program (ক্রমলেখ) লিখো যেটি run করলে (চালালে) তোমার program userকে (ব্যবহারকারী) শুভেচ্ছা জানাবে। আসলে এটিই হবে সিপিপি languageএ তোমার লেখা প্রথম program (ক্রমলেখ)। প্রায় প্রত্যেক programming languageএই এমন একটা করে program রচনা করা হয়।

নীচে wishing message দেখানোর জন্য একটি program লিখা হয়েছে। Programটি compile (সংকলন) করে execute (নির্বাহ) বা run (চালালে) করলে যে output (ফলন) পাওয়া যাবে তাও দেখানো হয়েছে। ওই programএ মুল যে statementটি (বিবৃতি) আমাদের wishing the best দেখাবে সেটি হল `cout << "wishing the best" << endl;` এখানে cout হল console out মানে consoleএর output device (ফলন যন্ত্র)। আর endl হল end line অর্থাৎ যেখানে endl বলা আছে সেখানে outputএ ওই সারি শেষ হবে। খেয়াল করো

আমরা monitorএ যা দেখাতে চাই তা হুবহু quotattion "" চিহ্নের ভিতরে লেখা হয়েছে। আর << দিয়ে আমরা "wishing the best" ও endl কথাগুলোকে cout এর কাছে পাঠাই।

মনে রেখো cout এর statementটি (বিবৃতি) ছাড়া আমাদের programএ আরো অন্যান্য statement যেগুলি আছে সেগুলি আমাদের লেখা প্রায় সকল programএই থাকবে। আমরা তাই আপাতত ওগুলো একরকম জোর করে মনে রাখার চেষ্টা করবো। তারপরেও অবশ্য আমরা নীচের আলোচনা থেকে সংক্ষেপে জেনে নেব বাঁকী statementগুলোর কোনটার কাজ কী।

ফিরিস্তি ১০.১: Wishing Program (শুভেচ্ছা জানানোর ক্রমলেখ)

```
#include <iostream>
using namespace std;
int main()
{
  cout << "wishing the best" << endl;

  return 0;
}
```

ফলন (output)

```
wishing the best
```

একদম শুরুতে আমরা #include <iostream> ব্যবহার করেছি কারণ iostream নামে একটা header file (শির নথি) আছে যেটা আমরা আমাদের programএ অন্তর্ভুক্ত করতে চাই। ওই header fileএ নানান function (বিপাতক) আছে যেগুলো আমরা পরে জানবো ও ব্যবহার করবো। আপাতত জেনে নেই, ওই fileএ cout আর endl আছে। মুলত আমাদের programএ cout আর endl ব্যবহার করার জন্যই আমরা iostream include করেছি। এরকম আরো header fileএর (শির নথি) কথা আমরা পরে বিস্তারিত জানবো ও অবশ্যই ব্যবহার করবো।

using namespace std; আমরা ব্যবহার করেছি কারণ cout আর endl আসলে দুটো নাম, আর ওই নাম দুটো সিপিপিতে বিদ্যমান std (standard বা প্রমিত) namespaceএর (নামাধার) অন্তর্গত। সিপিপিতে একই নাম ভিন্ন ভিন্ন namespaceএর অন্তর্গত হতে পারে। তো কোনো নাম বললে সেটি কোন namespace থেকে আসবে সেটি আমরা আগেই বলে দিচ্ছি, যেমন আমাদের সকল নাম আসলে std namespace থেকে আসবে। Namespace কী আর একটু ভালো করে বুঝতে নীচের paraতে (পরিচ্ছেদ) ঢাকার ও বগুড়ার গাবতলি নিয়ে আলোচনা পড়ো।

গাবতলি নামে ঢাকায় একটি জায়গা আছে আবার গাবতলি নামে বগুড়ায় আরেকটি জায়গা আছে। তো গাবতলি বলতে গেলে আমাদের বলতে হবে 'বগুড়ার গাবতলি' অথবা 'ঢাকার গাবতলি', কেবল গাবতলি বললে তো বুঝা যাবে না কোথাকার গাবতলি। বিকল্প হিসাবে আমরা আগেই বলে নিতে পারি যে আমরা এখন ঢাকার কথা আলোচনা করছি। তখন কেবল গাবতলি বললেই আমরা বুঝব এটি ঢাকার গাবতলি। আবার যদি আগেই বলে নেই যে এখন থেকে আমরা বগুড়ার কথা আলোচনা করবো তাহলে গাবতলি বললেই আমরা বগুড়ার গাবতলি বুঝব, ঢাকারটা নয়।

উপরের programএ using namespace std; বলে আমরা আগেই বলে নিয়েছি যে এর-পর থেকে আমরা std namespace নিয়ে কাজ করবো। কাজেই পরে যখন cout আর endl ব্যবহার করেছি, তখন আর std এর কথা বলতে হয় নি। কিন্তু কেউ যদি তার programএ using namespace std; না লেখে, তাহলে তাকে cout << "wishing the best" << endl;

এর বদলে লিখতে হবে std::cout << "wishing the best" << std::endl; মানে cout ও endl দুটোর পূর্বে std:: লাগিয়ে নিতে হবে, ঠিক যেমন গাবতলি বলার আগে ঢাকা লাগিয়ে বলতে হবে ঢাকার গাবতলি। cout আর endl এর আগে std:: না লিখলে program সফল ভাবে compile (সংকলন) করা যাবে না, নানান error message (ত্রুটি বার্তা) দেখাবে। Compile করার সময়ে দেখানো errorগুলোকে compile-time (সংকলন কালীন) error বলা হয়।

যে কোন সিপিপি programএ একটি মুল function (বিপাতক) থাকে main যার নাম। এই main functionএ কোন parameter (পরামিতি) থাকবে না, কাজেই main( ) এর পরে round bracket দুটোর মধ্যে কিছু বলা হয় নি। আর প্রতিটি function চাইলে একটি মান return করে, main function সাধারণত একটি integer (পূর্ণক) return করে, যা main লেখার আগে int হিসাবে উল্লেখ করা হয়েছে। Function নিয়ে বিস্তারিত আলোচনা আমরা পরে করবো। আপাতত সংক্ষেপে এইটুকুই জেনে রাখি। তো আমাদের programএ return 0; statementটি আসলে বলছে যে আমাদের main functionটি শুন্য return করবে। কার কাছে return করবে? যে আমাদের program চালাচ্ছে তার কাছে। main function 0 return করা মানে হলো, এটি সফল ভাবে শেষ হয়েছে, কোন ত্রুটি বিচ্যুতি ঘটে নি। কোন function (বিপাতক) থেকে 0 ছাড়া অন্যকিছু return করা নিয়েও আমরা পরে আলোচনা করবো।

সিপিপিতে দুটো curly bracketএর {} ভিতরে যা থাকে তাকে বলা হয় একটি block (মহল্লা)। প্রতিটি functionএর (বিপাতক) একটি bdoy (শরীর) থাকে যেটি blockএর ভিতরে থাকে। লক্ষ্য করে দেখো আমাদের main functionএর cout আর return দিয়ে শুরু হওয়া statement দুটি একটি blockএর ভিতরে রয়েছে। আর একটি বিষয় খেয়াল করো, আমাদের statementগুলোর শেষে কিন্তু একটি করে semicolon (দির্তি) ; রয়েছে। সিপিপিতে বেশীরভাগ statementএর পরেই আমরা এইরকম semicolon ; দিয়ে statement শেষ করি। ঠিক বাংলা ভাষায় প্রতিটি বাক্যের পরে দাঁড়ি । দেয়ার মতো ব্যাপার।

১০.২ Detailing Program (নাম-ধাম-বৃত্তান্তের ক্রমলেখ)

সিপিপিতে এমন একটি program (ক্রমলেখ) রচনা করো যেটি run করলে userকে তোমার নাম-ধাম-বৃত্তান্ত কয়েক সারিতে string (মালা) আকারে বলে দেয়। সাথে number (সংখ্যা) হিসাবে তোমার বয়স ও তোমার ফলাফলের জিপিএও বলে দেয়।

ফিরিস্তি ১০.২: Detailing Program (নাম-ধাম-বৃত্তান্তের ক্রমলেখ)

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
  cout << "my name is goni mia" << endl;
  cout << "i am from bogra" << endl;
  cout << "i live in dhaka" << endl;
  cout << "i am " << 20 << " years old" << endl;
  cout << "my result gpa " << 3.99 << endl;
  return EXIT_SUCCESS;
}
```

ফলন (output)

```
my name is goni mia
i am from bogra
i live in dhaka
i am 20 years old
my result gpa 3.99
```

নীচের programএ আমরা নাম-ধাম-বৃত্তান্ত কয়েক সারিতে দেখিয়েছি। এই programএর প্রতিটি cout দিয়ে শুরু statementএর (বিবৃতি) সাথে পরে দেখানো output (ফলন) মিলিয়ে নাও। লক্ষ্য করো cout দিয়ে " " quotationএর (উদ্ধৃতি) ভিতরে আমরা যে stringগুলো (মালা) দেখাতে বলেছি সেগুলোই outputএ হুবহু সেভাবেই দেখানো হয়েছে। আর প্রতিবার endl অর্থাৎ end line পেলে output পরের সারিতে চলে গেছে। শেষের দুটো cout statement খেয়াল করো। এইদুটোতে বয়স ও জিপিএ আমরা number (সংখ্যা) হিসাবে দেখিয়েছি। তুমি চাইলে কিন্তু সংখ্যা হিসাবে না দেখিয়ে stringএর ভিতরেই দেখাতে পারতে যেমন নীচের মতো, সেক্ষেত্রে output কিন্তু দেখতে একই রকম হতো।

```
cout << "i am 20 years old" << endl;
cout << "my result gpa 3.99" << endl;
```

সবশেষে একটা বিষয় খেয়াল করো। আমরা এই programএ return 0; এর বদলে লিখেছি return EXIT_SUCCESS; আর এই EXIT_SUCCESS আছে cstdlib header fileএ (শির নথি)। আমরা তাই #include <cstdlib> লিখে cstdlib header fileটিও আমাদের programএ include করেছি। মনে রাখবে EXIT_SUCCESS এর value আসলে 0 কিন্তু 0 তো একটা number যেটা দেখে সরাসরি ঠিক অনুধাবন করা যায় না আমরা কী বুঝাতে চাইছি, মানে program সফল না বিফল হয়েছে। আমরা তাই স্পষ্ট করে EXIT_SUCCESS লিখবো যাতে চোখে দেখেই আমরা বুঝতে পারি ব্যাপারটা কী। বলে রাখি computerএর (গণনি) জন্য কিন্তু 0 আর EXIT_SUCCESS একই ব্যাপার কারণ EXIT_SUCCESS এর value যে 0 ওইটা তো cstdlib fileএ বলা আছে, compile করার পরে EXIT_SUCCESS আসলে 0 হয়ে যাবে, computer ওইটা শুন্যই দেখতে পাবে। আমরা 0 এর বদলে EXIT_SUCCESS আসলে লিখছি কেবল মানুষের বুঝার সুবিধার জন্য, program পড়ে চোখে দেখেই যাতে সহজে বুঝা যায় programটি সফল না বিফল ভাবে শেষ হচ্ছে, সেটাই আমাদের উদ্দেশ্য। তাহলে এখন থেকে programএর main functionএ return 0; না লিখে return EXIT_SUCCESS; লিখবে আর cstdlib header fileও অন্তর্ভুক্ত করে নেবে!

তো তোমরা এখন থেকে কয়েক সারিতে কিছু দেখানোর program রচনা করতে চাইলে এই programএর মতো করে রচনা করবে। দরকার মতো number (সংখ্যা) ও string (মালা) মিশিয়েও কিন্তু যা দেখাতে চাও তা দেখাতে পারবে। চেষ্টা করে দেখো কেমন?

১০.৩ Writing Program Comments (ক্রমলেখতে টীকা লিখন)

এমন একটা program (ক্রমলেখ) রচনা করো যেটি বর্তমান সাল ২০১৫ থেকে তোমার বয়স ২০বছর বিয়োগ করে তোমার জন্ম বছর দেখায়। এই programএ দরকার অনুযায়ী পর্যাপ্ত পরিমানে comment (টীকা) লিখো, যাতে অনেক দিন পরে তুমি যখন programটি প্রায় ভুলে যাওয়ার মতো অবস্থায় যাবে তখন programটি আবার দেখতে গিয়ে দ্রুত চোখ বুলিয়েই সহজে বুঝতে পারো যে এটি তোমার কীসের program ছিল। Programএ comment থাকলে তুমি

ছাড়া অন্য কেউও তোমার লেখা program পড়ে সহজে বুঝতে পারবে। Comment লেখা হয় মানুষ যে ভাষায় কথা বলে সেই ভাষায় যেমন বাংলায় বা ইংরেজীতে, সিপিপি languageএও নয়, machine languageএও নয়, কাজেই comment লিখলে অনেক দিন পরেও আমাদের program বুঝতে সুবিধা হয়।

ফিরিস্তি ১০.৩: Commenting in Programs (ক্রমলেখতে টীকা লেখন)

```
// list of header files needed for this program.
#include <iostream>
#include <cstdlib>

using namespace std;  // use the std namespace

int main()
{
  // Subtract 20years from 2015 to get birthyear
  cout << "my birthyear " << 2015 - 20 << endl;

  return EXIT_SUCCESS;  /* return with success */
}
```

```
my birthyear 1995
```

উপরের program খেয়াল করো। কঠিন কিছু নয়। আগের মতোই iostream আর cstdlib include (অন্তর্ভুক্ত) করা আছে। তারপর বলা হয়েছে using namespace std; তারপর main function (বিপাতক) হিসাবে int main() যেটির কোন parameter (পরামিতি) নাই কারণ () round bracketএর ভিতরে কিছু নাই আর যেটি একটি integer (পূর্ণক) ফেরত দেয় কারণ int বলা আছে শুরুতে। তারপর main functionএর শরীরে দুটো {} curly bracketএর (বাঁকাবন্ধনী) ভিতরের blockএ (মহল্লা) বলা আছে cout << "my birthyear " << 2015 - 20 << endl; অর্থাৎ outputএ my birthyear দেখিয়ে তারপর 2015 থেকে 20 minus করলে যে 1995 পাওয়া যায় তা দেখাবে। তারপর blockএর ভিতরে শেষ statement (বিবৃতি) আছে return EXIT_SUCCESS; যা আগের মতোই বলছে যে আমাদের program ওইখানে সফল ভাবে শেষে হয়ে যাবে। EXIT_SUCCESS নিয়ে আমরা আগের পাঠ বিস্তারিত আলোচনা করেছি, ওই পাঠ থেকেই দেখে নিতে পারো, কাজেই সেটা আবার এখানে আলোচনা করছি না।

যাইহোক, খেয়াল করে দেখো ওপরে বর্ণিত বিষয়গুলো ছাড়াও উপরের programএ আরো কিছু বাক্য ও সারি দেখা যাচ্ছে যেমন প্রথম সারিটিই হল // list of header files needed for this program. বলতে গেলে এই বাক্যটি আসলে আমাদের programএর অংশ নয়, অর্থ্যাৎ program যখন run করা (চালানো) হবে তখন এই বাক্যের কোন প্রভাব থাকবে না। Program এমন ভাবে চলতে থাকবে যাতে মনে হবে ওই বাক্যটি যেন ওখানে নাই। এরকমের বাক্যগুলোকে বলা হয় comment (টীকা)। খেয়াল করো commentএর বাক্যটির একদম সামনে রয়েছে // অর্থাৎ সামনের দিকে হেলানো দুটো দাগ। ওই দুটো দাগ হতে শুরু করে ওই সারিতে তারপরে যাই থাকবে সব মিলিয়ে হবে একটি comment। এইরকম comment যেহেতু কেবল এক সারিতে সীমাবদ্ধ তাই একে বলা হয় line comment (সারি টীকা)।

Line comment যদি সারির শুরুতে লেখা হয় তাহলে সাধারণত এটি commentএর ঠিক নীচে যে code (সংকেত) থাকে তার জন্য লেখা হয়। যেমন দেখো // list of header files needed for this program এই commentটি একদম সারির শুরু থেকে লেখা হয়েছে, এটি তাই পরের দুই সারিতে #include <iostream> আর #include <cstdlib> কেন লেখা হয়েছে সেটি ব্যাখ্যা করছে। Line comment অনেক সময় সারির শেষদিকেও লেখা হয়। যেমন // we will use the std namespace commentটি লেখা হয়েছে using namespace std; দিয়ে শুরু হওয়া সারির শেষে। সারির শেষ দিকে লেখা এইরকম line comment সাধারণত সারির প্রথমে যে code (সংকেত) লেখা হয়েছে তা ব্যাখ্যা করতে ব্যবহার করা হয়। অনেক সময় comment লিখা হয় শুরুতে /* আর শেষে */ চিহ্ন দিয়ে, যেমন return EXIT_SUCCESS; এর সারিতে শেষে লেখা হয়েছে। এইরকম comment একাধিক সারি মিলিয়ে হতে পারে, তাই এদেরকে line comment না বলে block comment (মহল্লা টীকা) বলা হয়। সিপিপিতে আমরা অধিকাংশ সময় আসলে line commentই ব্যবহার করি।

তুমি যখন তোমার programএ comment লিখবে তখন হয়তো ইংরেজীতেই comment লিখবে। অথবা ইংরেজী অক্ষরে বাংলায়ও comment লিখতে পারো। এখন অনেক compiler (সংকলক) ও editor (সম্পাদক) ইউনিকোড (unicode) সংকেত বুঝতে পারে। কাজেই চাইলে comment বাংলায়ও লেখা সম্ভব। আমরা এরপর থেকে সিপিপিতে লেখা সকল programএ comment বাংলায় লিখবো, যাতে আমরা আমাদের নিজের ভাষায় সহজে বুঝতে পারি। এগুলো যেহেতু execute (নির্বাহ) হবে না, কাজেই খামোকা কেন কষ্ট করে ইংরেজীতে লিখতে যাবো! আর বিদেশী কেউ তো আমাদের programএর code দেখবে না, কাজেই আমরা আমাদের বাংলা ভাষাতেই comment লিখবো। তবে মনে রাখবে বিদেশী কারো পড়ার সম্ভাবনা থাকলে আমাদের comment সহ সবকিছু ইংরেজী ভাষাতেই লিখতে হবে। তাহলে line comment আর block comment শেখা হলো। এখন থেকে program লেখার সময় যথেষ্ট পরিমানে comment দিবে কেমন? আমিও programএ comment দেবো, যাতে তোমাদের বুঝতে সুবিধা হয়।

১০.৪ Spacing and Indentation (ক্রমলেখতে ফাঁকা দেওয়া)

সিপিপি program (ক্রমলেখ) লিখতে কখন নতুন সারি শুরু করবে? কখন ফাঁকা ফাঁকা করে লিখবে? কখন সারিতে একটু indentation দিয়ে লিখবে। একটি program লিখে এই বিষয়গুলো আলোচনা করো। চলো আমরা আমাদের wish করার ছোট programটি দিয়েই আলোচনা করি।

ফিরিস্তি ১০.৪: Spacing and Indentation (ক্রমলেখতে ফাঁকা দেওয়া)

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  cout << "wishing the best" << endl;

  return EXIT_SUCCESS;
}
```

উপরের programএ আমরা আপাতত comment (টীকা) লিখি নাই। এই programএ # বা কাটাকাটি (octothorpe) দিয়ে শুরু হওয়া সারিগুলো তোমাকে আলাদা আলাদা সারিতে লিখতে হবে। আর এটি সারির শুরু থেকে হলেই ভালো। তুমি অবশ্য সারির শেষের দিকে চাইলে line comment (সারি টীকা) লিখতে পারো যাতে বুঝা যায় ওই সারির শুরুর দিকে তুমি আসলে কী করতে চেয়েছো। তোমার programএ লেখা কাটাকাটি # দিয়ে শুরু হওয়া সারিগুলো সাধারণত compiler (সংকলক) দিয়ে process করা হয় না। আলাদা একটি software (মন্ত্র) যার নাম preprocessor (পুর্ব-প্রক্রিয়ক) সেটা দিয়ে compile করারও আগে এইগুলো process করা হয়, কাজটা বেশীর ভাগ সময়ে অবশ্য compilerই করিয়ে নেয়। Preprocessor (পূর্ব-প্রক্রিয়ক) বিষয়ে বিস্তারিত আলোচনা পরে হবে।

```
#include <iostream> //  input output stream header
 file
```

কোন statement preprocessor (পুর্বপ্রক্রিয়ক) না compiler দিয়ে process করা হবে এটা বুঝার আরেকটা সহজ উপায় আছে। এইরকম semicolon (দির্তি) ; আর blockএর (মহল্লা) জন্য যে curly bracket } ব্যবহার হয় তা দিয়ে শেষ হওয়া statementগুলো সাধারণত compiler দিয়ে process করা হবে, preprocessor দিয়ে নয়। যাইহোক, compiler দিয়ে যে codeগুলো প্রক্রিয়া করা হয় সেগুলো যে ভিন্ন ভিন্ন সারিতেই লিখতে হবে, বা অনেক ফাঁকা ফাঁকা (space) করেই লিখতে হবে এ রকম কোন কথা নেই। তুমি চাইলে তোমার পুরো programএ থাকা সকল compilable code এক সারিতে লিখতে পারো। যেমন উপরের programএর compilable অংশটুকু আমরা চাইলে নীচের মতো টানা এক সারিতে লিখতে পারি।

```
#include <iostream>
#include <cstdlib>
using namespace std; int main() { cout << "wishing
 the best" << endl; return EXIT_SUCCESS; }
```

উপরে যদিও দুই সারিতে দেখা যাচ্ছে আমরা আসলে using থেকে শুরু করে } পর্যন্ত টানা একসাথে লিখেছি, কিন্তু এখানে পাশের দিকে স্থানের স্বল্পতার কারণে টানা সারিটি ভেঙে দুই সারি হয়ে গেছে। তোমার editorএ (সম্পাদকে) এ যদি পাশের দিকে অনেক জায়গা থাকে তুমি এক সারিতেই লিখতে পারবে। আসলে নুন্যতম একটি space (ফাঁকা) দেয়া বাধ্যতামুলক হয়ে যায় যখন পরপর দুটো শব্দ লেখা হয়। যেমন using, namespace, std, int, main এইরকম শব্দ পরপর দুটো থাকলে তোমাকে কমপক্ষে একটি space (ফাঁকা) দিতে হবে। দুটো চিহ্ন যেমন bracket () বা semicolon ; বা আরো অনেক symbol আছে, এইগুলো পরপর দুটো থাকলেও কোন সমস্যা নাই; একাধিক symbol কোন ফাঁকা না দিয়েও তুমি একসাথে লিখতে পারবে।

এখন প্রশ্ন করতে পারো space দেয়া যদি ব্যাপার না হয়, তাহলে program লিখতে কেন ফাঁকা দেবো। বেশী বেশী ফাঁকা আসলে computerএর (গণনি) জন্য দরকার নেই কিন্তু দরকার মানুষের জন্য। আগের পাঠের কথা মনে করো। আমরা কেন comment (টীকা) লিখেছিলাম? Comment তো আর execute হয় না। আমরা যাতে অনেকদিন পরে programএর code (সংকেত) দেখে সহজে বুঝতে পারি, আমরা তাই comment লিখেছিলাম। তো program যদি পুরোটা একটা লম্বা সারি হয়, আমাদের মানুষের পক্ষে সেটা দেখে বুঝে ওঠা খুবই কষ্টকর হবে। মুলত আমাদের মানুষের বুঝার সুবিধার্থে আমরা program সারিতে সারিতে ভেঙ্গে ভেঙ্গে লিখি বা দরকার মতো একসাথে লিখি।

Programএ ফাঁকা দেয়ার ব্যাপারটি রচনা লেখার মতোই, কখন তুমি আলাদা বাক্য করবে, কখন তুমি আলাদা para (পরিচ্ছেদ) করবে, কখন তুমি আলাদা section (অনুচ্ছেদ) করবে, এই

রকম। কোন বিষয়ের সাথে বেশী সম্পর্কিত statementগুলো আমরা সাধারণত পরপর সারিতে কোন ফাঁকা (blank line) না দিয়ে লিখবো। আর দুটো বিষয়ের সারিগুলোর মাঝে হয়তো এক সারি ফাঁকা দিয়ে লিখবো, আর বিষয়গুলোর মধ্যে খুব বেশী যোগাযোগ না থাকলে হয়তো আমরা দুই বা আরো বেশী সারি ফাঁকা দিয়ে লিখবো। তাহলে এখন থেকে program লেখার সময় দরকার মতো ফাঁকা দিয়ে দিয়ে লিখবে যাতে তোমার program পড়া সহজ হয়।

সবচেয়ে উপরে যেভাবে আমরা program লিখেছি সেখানে আরো একটা ব্যাপার খেয়াল করো, আমরা cout বা return এর statementগুলো লেখার আগে তাদের নিজ নিজ সারিতে বেশ কিছুটা ফাঁকা দিয়ে লিখেছি, একদম সারির শুরু থেকে লিখি নাই। এটি কেন করলাম? এটি করলাম এ কারণে যে ওই দুটো সারি আসলে আমাদের blockএর ভিতরে আছে। লক্ষ্য করো blockএর curly bracket দুটো কেমন দেখেই বুঝা যায় যে এরা দুজনে দুজনার আর blockএর ভিতরের statementদুটো কেমন একটু ভিতরের দিকে থাকায় পরিস্কার বুঝা যায় যে ওরা আসলেই ওই blockএর ভিতরে। তো দরকার মতো কোন statement এরকম সারির একটু ভিতরের দিকে থেকে লেখার ব্যাপারটিকে বলা হয় indentation (ছাড়ন) দেয়া। Program লেখার সময় এখন থেকে তোমরা অবশ্যই দরকার মতো indentation দিয়ে লিখবে, তাহলে দেখবে program পড়া ও বোঝা কত সহজ হয়ে যায়।

এই পর্যায়ে জিজ্ঞেস করতে পারো, প্রত্যেক সারিতে এভাবে অতগুলা করে space চাপবো কেমনে এইটা তো বিরক্তিকর। আসলে তোমার keyboard (চাপনিমাঁচায়) একটা tab (লম্ফ) key আছে, দেখো ওইটা চাপলে একসাথে ৪টা বা ৮টা space (ফাঁকা) এর সমপরিমান ফাঁকা একবারে আসে। তো দরকার মতো একবার বা দুবার tab চাপলেই হয়ে গেলো। কাজেই program লেখার সময় কখনোই এই আলসেমি টুকু করবে না। Indentation দেয়া program লেখার জন্য গুরুত্বপূর্ণ ব্যাপার, সুন্দর দেখা যাওয়া আর তাড়াতাড়ি পড়ার জন্য দরকারী, Programএ কোন ভুল থাকলে আমরা যখন ভুল বের করতে চাই তখনও খুব খুব দরকারী, বড় বড় program যখন লিখবে তখন ব্যাপারটা খানিকটা ঠেকে ঠেকে শিখে অভিজ্ঞতা দিয়ে ভালো করে বুঝতে পারবে।

১০.৫ Exercise Problems (অনুশীলনী সমস্যা)

Conceptual Questions: নীচে কিছু conceptual প্রশ্ন রয়েছে। প্রশ্নগুলোর উত্তর নিজে বের করবে।

১. Program (ক্রমলেখ) বলতে কী বুঝো? Program কি কেবল computerএই executable?

২. সিপিপি languageএ program তৈরী হতে সেটার ফলাফল দেখা পর্যন্ত কী কী ঘটনা ঘটে?

৩. Console (যন্ত্রালয়) কী? এর input (যোগান) ও output (ফলন) deviceগুলো কী কী?

৪. সিপিপিতে header file (শিরনথি) বলতে কী বুঝো? আমাদের programএ header file iostream ও cstdlib আমরা কেন ব্যবহার করেছি?

৫. Name space (নামাধার) কী? বাস্তব জীবনে ও programmingএর উদাহরণ সহ ব্যাখ্যা করো।

৬. সিপিপিতে main function হতে returnএর সময় return 0; না লিখে তার বদলে return EXIT_SUCCESS; লিখা কেন উত্তম? ব্যাখ্যা করো।

৭. Programএ indentation (ছাড়ন) দেয়া মানে ঠিক কী? Indentation দেয়ার পক্ষে-বিপক্ষে যুক্তি কী? Program কেন বেশ ফাঁকা ফাঁকা করে লিখা উচিৎ?

৮. Programএ comment (টীকা) লেখা কী? Programএ comment লিখার কয়েকটি কারণ ব্যাখ্যা করো? line (সারি) comment ও block (মহল্লা) comment কী?

৯. একটি সিপিপি programএ (ক্রমলেখ) নীচের কোন functionটি অবশ্যই থাকতে হবে?

ক) start() খ) system() গ) main() ঘ) program()

১০. Program সফল ভাবে শেষ হলে main function হতে সাধারণত কত return করা হয়?

ক) -1 খ) 0 গ) 1 ঘ) কিছুই না

১১. সিপিপিতে block (মহল্লা) বুঝানোর জন্য নীচের কোনগুলো ব্যবহার করা হয়?

ক) { } খ) < > গ) ( ) ঘ) begin end

১২. সিপিপিতে একটি statementএর (বিবৃতি) শেষে সাধারণত কোন চিহ্ন ব্যবহার করা হয়?

ক) . খ) ; গ) : ঘ) ,

১৩. সিপিপিতে নীচের কোনটি সঠিক comment (টীকা)?

ক) */comment*/ গ) /*comment*/
খ) **comment** ঘ) { comment }

Programming Problems: নীচে আমরা কিছু programmingএর সমস্যা দেখবো। এই সমস্যাগুলো আগে ধৈর্য্য ধরে নিজে নিজে সমাধান করতে চেষ্টা করবে। যখন একেবারেই পারছো না বলে মনে হয় তখন কেবল সমাধান দেখে নিতে পারো। সমাধানগুলো programmingএর প্রশ্নগুলোর শেষে আছে।

১. নীচের কথাগুলো outputএ (ফলন) দেখানোর জন্য সিপিপিতে একটি program লিখো। দেখতে সুন্দর লাগার জন্য তোমার programএ দরকার মতো ফাঁকা ফাঁকা দিবে আর সহজে বুঝার জন্য দরকার মতো comment (টীকা) লিখবে।

```
your age is nine years.
want to learn programming?
programming is so easy!
```

২. সিপিপিতে একটি program রচনা করো যেটি নীচের নকশাটির মতো নকশা তৈরী করে। খেয়াল করে দেখো নকশাটি বাংলা অঙ্ক 8 এর মতো। তুমি চাইলে আরো নানান নকশা, নানান বর্ণ বা অঙ্ক নিজের মতো করে ভেবে নিয়ে সেইমতো নকশা তৈরী করতে পারো। যাইহোক দেখতে সুন্দর লাগার জন্য তোমার programএ দরকার মতো ফাঁকা ফাঁকা দিবে আর সহজে বুঝার জন্য দরকার মতো comment (টীকা) লিখবে।

```
*****
*   *
* * *
*   *
*****
```

Programming Solutions: এবার আমরা programming সমস্যাগুলোর সমাধান দেখবো। মনে রাখবে সমাধান দেয়া হয়েছে তুমি যদি একান্তই নিজে নিজে করতে পারছোনা বলে মনে হয়, তখন যাতে একটু সাহায্য কেবল পেতে পারো তাই। কাজেই পারতপক্ষে সমাধান দেখবে না।

১. নীচের কথাগুলো outputএ (ফলন) দেখানোর জন্য সিপিপিতে একটি program লিখো। দেখতে সুন্দর লাগার জন্য তোমার programএ দরকার মতো ফাঁকা ফাঁকা দিবে আর সহজে বুঝার জন্য দরকার মতো comment (টীকা) লিখবে।

```
your age is nine years.
want to learn programming?
programming is so easy!
```

ফিরিস্তি ১০.৫: Inspiring Program (অণুপ্রেরণার ক্রমলেখ)

```
#include <iostream> //  cout ব্যবহার করার জন্য
#include <cstdlib>  //  EXIT_SUCCESS এর জন্য

using namespace std;// standard namespace ব্যবহার

int main()
{
  //  দরকারী কথাগুলো এoutput দেখাও
  cout << "your age is nine years." << endl;
  cout << "want to learn programming?" << endl;
  cout << "programming is so easy!" << endl;

  return EXIT_SUCCESS;  //  সফল সমাপ্তি
}
```

২. সিপিপিতে একটি program রচনা করো যেটি নীচের নকশার মতো নকশা তৈরী করে। খেয়াল করে দেখো নকশাটি বাংলা অঙ্ক ৪ এর মতো। তুমি চাইলে আরো নানান নকশা, নানান বর্ণ বা অঙ্ক নিজের মতো করে ভেবে নিয়ে সেইমতো নকশা তৈরী করতে পারো। যাইহোক দেখতে সুন্দর লাগার জন্য তোমার programএ দরকার মতো ফাঁকা ফাঁকা দিবে আর সহজে বুঝার জন্য দরকার মতো comment (টীকা) লিখবে।

```
*****
*   *
```

```
* * *
*   *
*****
```

এই programটি কিন্তু অনেক মজার তাই না। তুমি কি বর্ণমালার প্রতিটা বর্ণ আর ০-৯ দশটা অঙ্কের জন্যেই এরকম নকশা তৈরী করতে পারবে? রাস্তাঘাটে বা বিয়ে বাড়িতে অনেক সময় ছোট ছোট বাতি দিয়ে নানান কিছু লেখা হয়, আসলে এই নকশাগুলোর মতো করে নকশা বানিয়েই সেগুলো করা হয়। Computerএর (গণনি) monitorএর (নজরি) পর্দায়ও অনেক কিছু এভাবে দেখানো হয়। আসলে যে কোন ছবিই এরকম অসংখ্য বিন্দুর সমন্বয়ে তৈরী, কিছু বিন্দু জ্বালানো, কিছু বিন্দু নেভানো। যে বিন্দুগুলো জ্বালানো সেগুলো হলো * আর যেগুলো নেভানো সেগুলো ফাঁকা। তো চলো আমরা programটি দেখি।

ফিরিস্তি ১০.৬: Program to Design (নকশা আঁকার ক্রমলেখ)

```
#include <iostream> //  cout ব্যবহার করার জন্য
#include <cstdlib>  //  EXIT_SUCCESS এর জন্য

using namespace std;// standard namespace ব্যবহার

int main()
{
  // দরকার মতো * ও ফাঁকা দিয়ে নকশা
  cout << "*****" << endl;
  cout << "*   *" << endl;
  cout << "* * *" << endl;
  cout << "*   *" << endl;
  cout << "*****" << endl;

  return EXIT_SUCCESS;  //  সফল সমাপ্তি
}
```

১০.৬ Computing Terminology (গণনা পরিভাষা)

- Octothorpe (কাটাকাটি) #
- Program (ক্রমলেখ)
- Computer (গণনি)
- Keyboard (চাপনি)
- Run (চালানো)
- Indentation (ছাড়ন)
- Webpage (জালপাতা)
- Mouse (টিপনি)
- Comment (টীকা)
- Error (ত্রুটি)
- Semicolon (দির্তি) ;
- Monitor (নজরি)
- Namespace (নামাধার)
- Instruction (নির্দেশ)

১০.৬. Computing Terminology (গণনা পরিভাষা)

- Execution (নির্বাহ)
- Executable (নির্বাহযোগ্য)
- Parameter (পরামিতি)
- Para (পরিচ্ছেদ)
- Preprocessor (পুর্ব-প্রক্রিয়ক)
- Integer (পূর্ণক)
- Standard (প্রমিত)
- Output (ফলন)
- Output Device (ফলন যন্ত্র)
- Message (বার্তা)
- Function (বিপাতক)
- Statement (বিবৃতি)
- Software (মন্ত্র)
- Block (মহল্লা)
- Block Comment (মহল্লা টীকা)
- String (মালা)
- Console (যন্ত্রালয়)
- Input (যোগান)
- Body (শরীর)
- Header File (শির নথি)
- Compiler (সংকলক)
- Compile (সংকলন)
- Compile Time (সংকলন কালীন)
- Code (সংকেত)
- Number (সংখ্যা)
- Editor (সম্পাদনা)
- Line Comment (সারি টীকা)

অধ্যায় ১১

Variables and Constants)
(চলক ও ধ্রুবক)

Variableএর value (মান) বদলানো যায়, কিন্তু constantএর value বদলানো যায় না। তবে Programএ data (উপাত্ত) সরাসরি (directly) না লিখে variable বা constant হিসাবে ব্যবহার করলে একরকমের indirection (পরোক্ষতা) তৈরী হয়। ফলে data ঠিক কতো সেটা না ভেবে data কীসের আর তার processing কেমন সেটা ভেবে program তৈরী সহজ হয়।

১১.১ Using Variables (চলকের ব্যবহার)

একটি আয়তের দৈর্ঘ্য ৫ মিটার, প্রস্থ ৩ মিটার। সিপিপি ভাষায় এইরূপ আয়তের ক্ষেত্রফল ও পরিসীমা বের করার program (ক্রমলেখ) রচনা করো। এই programএ তোমাকে variable (চলক) ব্যবহার করতে হবে, সরাসরি সূত্র থেকে output (ফলন) দেয়া যাবে না।

আমরা আগে এই সমস্যার জন্য সংক্ষিপ্ত programটা দেখি যেটাতে variable ব্যবহার না করে একদম সরাসরি সূত্র ব্যবহার করে ক্ষেত্রফল output (ফলন) দেখানো হবে। আমরা জানি দৈর্ঘ্য আর প্রস্থের গুণফল হল ক্ষেত্রফল আর দৈর্ঘ্য ও প্রস্থের যোগফলের দ্বিগুণ হলো পরিসীমা।

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  cout << "area " << 5 * 3
          << " squared-meter" << endl;
  cout << "perimeter " << 2*(5+3)
        << " meter" << endl;

  return EXIT_SUCCESS;
}
```

১১.১. Using Variables (চলকের ব্যবহার)

উপরে আমরা যে program (ক্রমলেখ) লিখলাম আমরা কিন্তু ওইটা চাই না। ওইখানে সংখ্যাগুলো সরাসরি সুত্রে বসিয়ে হিসাব করে output (ফলন) দেখানো হয়েছে। আমরা চাই ক্ষেত্রফল আর পরিসীমার সূত্রগুলো variableএর (চলক) নাম দিয়ে লিখতে আর সূত্র লিখার আগে variableগুলোর মান দিয়ে দিতে। Variable ব্যবহারের নানান সুবিধা আছে। যেমন একটি সুবিধা হলো সুত্রে variableএর নাম থাকায় সূত্র দেখেই সহজে বুঝা যায় কীসের সূত্র, যেমন নীচের program দেখো। আর একটি সুবিধা হলো কেউ যদি বলে ৫ না দৈর্ঘ্য হবে ৬, উপরের programএ কিন্তু দুইখানে 5 বদলাইয়া 6 করতে হবে। ছোট একটা programএই যদি দুইখানে বদলাতে হয়, তাহলে বড় একটি programএর কথা চিন্তা করো, সেটাতে আরো কত জায়গায় যে বদলাতে হবে ইয়ত্তা নাই। আমরা এ কারণে variable ব্যবহার করবো।

ফিরিস্তি ১১.১: Variables in Programs (ক্রমলেখতে চলকের ব্যবহার)

```
#include <iostream>
#include <cstdlib>
using namespace std;

int main()
{
  int length; // এই variable এ থাকবে দৈর্ঘ্য।
  int width;  // এই variable এ থাকবে প্রস্থ।
  int area;   // এই variable এ থাকবে ক্ষেত্রফল।
  int perimeter;  // এই variable এ থাকবে পরিসীমা।

  length = 5; // দৈর্ঘ্যের এই মান বলে দেয়া আছে
  width = 3;  // প্রস্থের এই মান বলে দেয়া আছে।

  // ক্ষেত্রফল বের করার সূত্র হল দৈর্ঘ্য আর প্রস্থের গুণফল।
   area = length * width;

  // পরিসীমা বের করার সূত্র হল দৈর্ঘ্য ও প্রস্থের যোগফলের দ্বিগুন
  perimeter = 2*(length + width);

  // এবার ক্ষেত্রফল আর পরিসীমা output দেয়া হবে
  cout<< "area " << area
        << " squared-meter" << endl;
  cout << "perimeter " << perimeter
        << " meter" << endl;

  return EXIT_SUCCESS;
}
```

ফলন (output)

```
area 15 squared-meter
periment 16 meter
```

উপরের programএ খেয়াল করো আমরা দৈর্ঘ্য, প্রস্থ, ক্ষেত্রফল, আর পরিসীমার জন্য চারটা variable (চলক) নিয়েছি যাদের নাম হলো length, width, area, perimeter। তুমি কিন্তু চাইলে এই নামগুলো বাংলায়ও দিতে পারতে যেমন doirgho, prostho, khetrofol, porishima। তুমি চাইলে আবার শব্দগুলোর প্রথম অক্ষর নিয়ে এক অক্ষরের নামও দিতে পারতে যেমন l, w, a, p। তবে আমরা সবসময় চাই এমন নাম দিতে যাতে নামগুলো দেখলেই বুঝা যায় ওই variableটা কী কাজে ব্যবহার হবে। এক অক্ষরের নাম দিলে অনেক সময় বুঝা যায় কিন্তু একই অক্ষর দিয়ে যদি একাধিক variableএর নাম শুরু হয়, তাহলে মুশকিল হয়ে যায়। অনেকে আবার খালি x, y, z, অথবা a, b, c এই রকম নাম দেয়। ওই রকম নাম দিলে পরে program বুঝতে তোমার নিজের বা অন্য কেউ যে পড়বে তার খুবই সমস্যা হবে। সময় নষ্ট করে বের করতে হবে কোন variable আসলে কী কাজে ব্যবহার করা হয়েছে। কাজেই সবসময় অর্থবোধক আর যথেষ্ট বড় নাম দিতে চেষ্টা করবে, যাতে নাম দেখেই তার উদ্দেশ্য বুঝা যায়। সিপিপিতে variableএর অর্থবোধক (semantic) ও গঠনসিদ্ধ (syntax) নাম দেয়ার বিষয়ে আমরা পরের কোন পাঠে বিস্তারিত আলোচনা করব।

এখন একটা বিষয় খেয়াল করো আমরা এখানে variableগুলোর নামের আগে লিখেছি int যেটা আসলে integer এর সংক্ষিপ্ত। integer হল পূর্ণক বা পূর্ণ সংখ্যা। আমরা variableএর নামের আগে এই রকম int লিখে বুঝিয়েছি যে আমাদের এই variableগুলোর মান হবে integer, আমরা কোন ভগ্নাংশ ব্যবহার করবো না। তুমি যদি ভগ্নাংশ ব্যবহার করতে চাও তাহলে তোমাকে int এর বদলে float লিখতে হবে। float হল একরকমের ভগ্নাংশ। আমরা ভগ্নক বা float নিয়ে আলোচনা পরে আরো বিস্তারিত করবো। তবে int এর বদলে float লিখলে আমাদের programএ কিন্তু আর কোথাও কোন কিছু বদলাতে হবে না, ঠিক কাজ করবে। আমরা আপাতত int রেখেই এই পাঠের আলোচনা চালাই।

তো উপরের programএ আমরা যখন লিখলাম int length; এর মানে হলো length নামের আমাদের একটা variable আছে আর তার মান হবে integer। এইযে int length; লিখে এই বিষয়গুলো বুঝাইলাম এটাকে বলা হয় variable declaration (চলক ঘোষনা)। Variable declare করলে তারপর থেকেই variableটি পরবর্তী যেকোন statementএ (বিবৃতি) ব্যবহার করা যায়, কিন্তু declare করার সাথে সাথে ওইখানে variableএর মান কত সেইটা কিন্তু আমরা জানিনা, সাধারনত variableএ তখন একটা উল্টাপাল্টা মান থাকে। এইটা নিয়ে আমরা পরে আরো আলোচনা করবো। এই programএ আমরা দেখছি এর পরে length = 5; লিখে অর্থাৎ = চিহ্ন ব্যবহার করে আমরা length variableএর value assign (মান আরোপ) করেছি ৫। সুতরাং এরপর থেকে length variableএর মান হবে ৫। একই ভাবে width variableএর মানও আমরা ৩ assign করেছি।

এবার খেয়াল করো, variableএর মান assignment শেষ হলে আমরা ক্ষেত্রফল আর পরিসীমার সূত্রগুলো লিখেছি, সেখানে কিন্তু এবার মানগুলো সরাসরি লিখি নাই, তার বদলে variableগুলো ব্যবহার করেছি। এইখানে হিসাব করার সময় variableএর যে মান থাকবে সেইটাই আসলে ব্যবহার হবে। উপরে যদি length variableএর মান থাকে ৫ তাহলে ৫ ধরে হিসাব হবে, আর যদি পরে length এর মান ৫ এর বদলে ৬ করে দেয়া হয়, তাহলে ৬ ব্যবহার হবে। এই পরিবর্তন কেবল মান assignmentএর ওইখানে করলেই কাজ হয়ে যাবে, সারা programএ করতে হবে না। তবে একটা গুরুত্বপূর্ণ বিষয় বলি এখানে length আর width variable দুটিতে value assignment কিন্তু ক্ষেত্রফল আর পরিসীমার সুত্রের ব্যবহারের আগেই করতে হবে। না করলে compile করার সময় warning message (সতর্ক বার্তা) আসতে পারে, আর program চালানোর সময় উল্টোপাল্টা ফলও আসতে পারে।

সবশেষে খেয়াল করো output (ফলন) দেওয়া হয়েছে যেখানে সেখানে উদ্ধৃতি চিহ্ন "" এর ভিতরে যা আছে তা কিন্তু string (মালা)। কাজেই ওইটা কিন্তু ওইভাবেই outputএ এসেছে এম-

নকি area কথাটাও হুবহু এসেছে যেটা কিনা variableএর নামের হুবহু একই রকম। কিন্তু "" উদ্ধৃতির বাইরে যখন area লেখা হয়েছে একই সারিতে পরের দিকে সেখানে কিন্তু আর area outputএ আসে নি, এসেছে সেটাকে variable ধরলে যে মান হওয়ার কথা সেই 15। কাজেই এটা মনে রাখবে যে variableএর নাম "" উদ্ধৃতির ভিতরে string আকারে থাকলে ওইটা আসলে variableটাকে বুঝায় না। নামটা যখন উদ্ধৃতির বাইরে থাকে তখন ওইটা একটা নাম হয়, এই-ক্ষেত্রে একটা variableএর নাম হয় আর ওইটার মান নিয়ে কাজ হয়। একই অবস্থা perimeter এর ক্ষেত্রেও। উদ্ধৃতি চিহ্নের ভিতরে থাকা perimeter কথাটি হুবহু ফলনে এসেছে কিন্তু উদ্ধৃতির বাইরে থাকা perimeter কথাটির বদলে ওটিকে variable (চলক) ধরলে যে মান পাওয়া যাবে তা outputএ (ফলন) এসেছে।

১১.২ Using Constants (ধ্রুবকের ব্যবহার)

একটি বৃত্তের radius (ব্যাসার্ধ) দেয়া আছে ৫ সেমি, বৃত্তটির area (ক্ষেত্রফল) নির্ণয়ের জন্য সি-পিপিতে একটি program (ক্রমলেখ) রচনা করো। তোমার programএ তুমি ব্যাসার্ধের জন্য একটি integer (পূর্ণক) ব্যবহার করবে। আর ক্ষেত্রফলের জন্য প্রথমে integer ব্যবহার করে দেখবে কী হয়, তারপর fractioner (ভগ্নক) অর্থাৎ floating-point number (সচলবিন্দু সংখ্যা) বা float ব্যবহার করবে। তুমি তো জানো বৃত্তের ক্ষেত্রফল হিসাব করার জন্য আমাদের পাইয়ের মান লাগবে। আমরা ওইটা সরাসরি সংখ্যায় না দিয়ে একটা constant (ধ্রুবক) হিসা-বে ব্যবহার করবো, কারণ পাইয়ের মান তো কখনো বদলাবে না, সব সময় constant থাকবে। পাইয়ের মান যেহেতু fractioner আমাদের constantটি তাই হবে float constant। চলো আমরা এবার তাহলে programটি দেখি।

ফিরিস্তি ১১.২: ক্রমলেখতে ধ্রুবকের ব্যবহার (Constants in Programs)

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  int radius = 5; // এটি বৃত্তের ব্যাসার্ধের জন্য variable
  float const pi = 3.1415;  // পাইয়ের মান constant

  // নীচে আমরা বৃত্তের ক্ষেত্রফলের সূত্র লিখছি
  int area = pi * radius * radius;

  // এবার আমরা এscreen output দেখাবো।
  cout << "area of circle " << area
       << " squared-cm" << endl;

  return EXIT_SUCCESS;
}
```

ফলন (output)

```
area of circle 78 squared-cm
```

তো হয়ে গেল আমাদের বৃত্তের ক্ষেত্রফল নির্ণয়ের program। এই রকম variable (চলক) আর constant (ধ্রুবক) ব্যবহার না করেই তুমি কিন্তু program লিখতে পারতে, তাই না! আমরা কিন্তু সেটা আগের পাঠে এটা আলোচনা করেছি। সেক্ষেত্রে main() functionএর {} bracket দুটোর মধ্যে return EXIT_SUCCESS; এর আগে মাত্র এক সারিতে cout << "area of the circle " << 3.1415 * 5 * 5 << " squared-cm" << endl; এইটুকু লিখলেই আমাদের কাজ হয়ে যেতো। কিন্তু আমরা সেটা না করে কেন variable ব্যবহার করছি সেটাও ওই একই পাঠে আলোচনা করেছি।

এবার আসি আমরা এতক্ষণ যে programটি লিখলাম সেটার বিস্তারিত আলোচনায়। আমরা #include, using namespace, int main(), return ইত্যাদি সম্পর্কে ইত্যমধ্যে জেনেছি আগের পাঠগুলো থেকে, কাজেই আমাদের আলোচনা সীমাবদ্ধ থাকবে main() functionএ আর যা যা লিখেছি সেই বিষয়গুলোতে। তো চলো আমরা এবার সারির পরে সারি ধরে আলোচনা করি।

উপরে যেমন বলা হয়েছে, সেই অনুযায়ী আমরা প্রথমে ব্যাসার্ধের জন্য একটা variable নিয়েছি radius নামে যেটি হবে int ধরনের অর্থাৎ পূর্ণক বা integer। বৃত্তের ব্যাসার্ধ যদি তোমার fractioner হয়, তুমি চাইলে int ব্যবহার না করে float ব্যবহার করতে পারো। আগের পাঠের সাথে এই পাঠে একটা বিষয় খেয়াল করো, আমরা কিন্তু ব্যাসার্ধ radius variable মান আলাদা সারিতে না দিয়ে যেখানে variable declare (চলক ঘোষনা) করেছি সেখানেই = চিহ্ন দিয়ে value assign করেছি অর্থাৎ radius এর মান সরাসরি 5 হয়ে গেছে। এটাকে বলা হয় variableএর initial value assignment (আদি মান আরোপণ)। এটা করার দুটো সুবিধা: একটা হলো আমাদের দুইটা আলাদা সারিতে দুইবার লিখতে হলো না, আরেকটা হলো শুরুতে variableএ উল্টাপাল্টা মান থাকার কারণে programএ ভুল হিসাব করার সম্ভাবনা কমে গেল। জেনে রাখো variable declareএর সাথে সাথে কোন মান না দিয়ে না দিলেও ওখানে উল্টা পাল্টা একটা মান থাকে, কী মান থাকবে আমরা কিন্তু কোন ভাবেই আগে থেকে সেটা জানিনা, পুরাই উল্টাপাল্টা একটা মান হতে পারে সেটা। আর ভুলক্রমে ওই variableএ যদি পরে আর value assign (আরোপ) করা না হয়, অথবা যদি assign করার আগেই অন্য কোন হিসাবে variableটি ব্যবহার করা হয়, তাহলে সঙ্গত কারণেই উল্টাপাল্টা মানটি কাজে লাগিয়ে একটা উল্টাপাল্টা ফলাফল আসবে, যেটা আমরা কখনোই চাই না।

ব্যাসার্ধের জন্য একটা variable (চলক) নেয়ার পরে আমরা পাইয়ের মান রাখার জন্য একটি float const ধরনের constant (ধ্রুবক) নেবো যার নাম pai। পাইয়ের মান যেহেতু ভগ্ন সংখ্যা আমাদের তাই float নিতে হবে, আর পাইয়ের মান যেহেতু সব সময় constant তাই আমরা float এর পরে const লিখে দিতে চাই। তুমি যদি const না লিখো তাহলে কিন্তু এটা একটা variableএর মতো কাজ করবে।

```
int myvar = 15; // variable declare করে মান দিলাম 15
int const myconst = 20; // constant declare মান 20

// এখন পর্যন্ত myvar এর মান 15, নীচে নতুন মান দেবো 23
// আবার value assign না করা পর্যন্ত এরmyvar মান থাকবে 23

variable = 23;     // এটা করা যাবে

// এখন পর্যন্ত myconst এর মান 20, নীচে নতুন মান দেবো 25
```

```
// কিন্তু program compile করলে আমরা error message পাবো।
// cpp.sh দিয়ে compile করলে error টিmessage নিম্নরূপ হবে
// error: assignment of read-only variable 'myconst'

myconst = 25;    // এটা করা যাবে না, error message আসবে
```

উপরের program খেয়াল করো। Variable (চলক) আর constantএর (ধ্রুবক) মধ্যে তফাৎ হলো variableএর মান declare করার সময় একবার assign করা যায়, আর তারপরেও যতবার ইচ্ছা ততবার নতুন নতুন মান assign (আরোপ) করা যায়। কিন্তু constantএ (ধ্রুবক) একটা মান কেবল declare করার সময় বলে দেওয়া যায়, programএ পরে আর কোথাও ওই constantএর মান বদলে নতুন মান assign (আরোপ) করা যায় না। যদি করো তাহলে compiler (সংকলক) error message (ত্রুটি বার্তা) দেখাবে। তো আমরা যেহেতু জানি যে পাইয়ের মান সবসময় constant, এটার মান আমাদের কখনো বদল হবে না, আমরা তাই এটাকে variable হিসাবে declare না করে constant হিসাবে declare করবো।

আশা করা যায় variable আর constantএর পার্থক্য পরিস্কার হয়েছে। এবার দেখো আমাদের বৃত্তের ক্ষেত্রফলের programএ আমরা ক্ষেত্রফলের জন্য area নামে একটা variable নিয়েছি, যার type হল int বা পূর্ণক. যদিও আমরা জানি পাইয়ের মান fractioner হওয়ার কারণে আমাদের ফলাফল আসলে একটি fractioner হবে। এইটা আমরা মুলত পরীক্ষামুলক করছি। তো int নেয়ার কারণে আমরা আমাদের programএর output দেখতে পাবো 78 আসলে হওয়ার কথা 78.5375। এইটা কেন হলো কারণ হলো প্রথমে 78.5375 ঠিক মতো ভিতরে ভিতরে হিসাব হয়ে যাবে, কিন্তু যখন area variableএর মধ্যে মানটা assign (আরোপ) হবে তখন যেহেতু integer বলে fractionটুকু ঢুকানো যাবে না, তাই ওইটা বাদ পরে যাবে (truncation)। আর মান যেটা assign হবে সেটা হলো বাঁকী পূর্ণাংশটুকু বা 78। তো ভগ্নাংশ সহ সঠিক ক্ষেত্রফল পাওয়ার জন্য area এর সামনে int না লিখে float লিখে দাও তাহলে দেখবে ঠিক ঠিক 78.5375 ই output হিসাবে চলে আসবে।

উপরের আলোচনায় আমরা তিনটা ব্যাপার শিখলাম: ১) আমরা variable (চলক) ব্যবহার করবো, না constant (ধ্রুবক) ব্যবহার করবো সেটা; তারপর ২) declare করার সাথে সাথে একটা initial value দিয়ে দেয়া যাকে বলা হয় initial assignment (আদি মান আরোপণ), আর ৩) কোন variable বা constantএর type কেমন হবে, int না float হবে, পূর্ণক না ভগ্নক হবে সেটা আগে থেকে ধারনা করতে পারতে হবে, আর সেই অনুযায়ী variable বা constantএর প্রকার বলে দিতে হবে, না হলে সঠিক ফলাফল নাও পাওয়া যেতে পারে, যেমন ভগ্নক 78.5375 এর বদলে পূর্ণক 78 পাওয়া যেতে পারে।

১১.৩ Variable Declarations (চলক ঘোষনা)

এই পাঠে সিপিপিতে একাধিক variable আমরা কী ভাবে সহজে declare করতে পারি তা আলোচনা করবো। আমরা আগে দেখেছি প্রতিটি variable আলাদা আলাদা করে, এমনকি প্রতিটি আলাদা আলাদা সারিতে ঘোষনা করতে। সুবিধার জন্য আমরা চাইলে একাধিক variable এক সারিতেই একটা statementএই declare করতে পারি, যদি তাদের সকলের data type (উপাত্ত প্রকরণ) একই হয়, যেমন ওই variableগুলোর সবই যদি int ধরনের হয় অথবা float ধরনের হয়। উদাহরণ দিয়ে ব্যাপারগুলো পরিস্কার করা যাক। ধরো length, width, perimeter নামে আমরা তিনটি variable (চলক) নিলাম, তিনটা variableএর typeই int অর্থাৎ পূর্ণক।

```
int length; // দৈর্ঘ্যের জন্য চলক যা int ধরনের অর্থাৎ পূর্ণক
int width;  // প্রস্থের জন্য চলক যা int ধরনের অর্থাৎ পূর্ণক
int perimeter;  // পরিসীমার জন্য চলক যা int ধরনের অর্থাৎ পূর্ণক
```

উপরের তিনটি variableই যেহেতু int ধরনের, কাজেই আমরা ওই তিনটি variableকে চাইলে একটি statementএই ঘোষনা করতে পারি। সেক্ষেত্রে আমাদের int একবার লিখতে হবে, আর variableগুলোর নাম একটার পর একটা comman , (বির্তি) দিয়ে লিখতে হবে।

```
int length, width, perimeter;// সবগুলোই int ধরনের
```

এবার আর একটি উদাহরণ দেখি, যেখানে বৃত্তের ব্যাসার্ধ আর ক্ষেত্রফল বের করতে হবে। তো ব্যাসার্ধ যদি int ধরনের বা পূর্ণক হয় আর ক্ষেত্রফল তো float ধরনের বা ভগ্নক হবেই। কাজেই আমরা এ দুটোকে একটা statement (বিবৃতি) দিয়ে declare করতে পারবো না।

```
int radius;   // ব্যাসার্ধের জন্য variable int ধরনের
float area; // ক্ষেত্রফলের জন্য variable float ধরনের
```

কিন্তু যদি perimeter এর মতো radius টাও float বা ভগ্নক ধরনের হতো তাহলে আমরা এক statement দিয়েই দুটোকে এক সাথে declare করতে পারতাম।

```
float radius, area; // ব্যাসার্ধ ও ক্ষেত্রফলের চলক
```

তাহলে একটা programএই (ক্রমলেখ) যদি আমরা আয়তের পরিসীমা আর বৃত্তের ক্ষেত্রফল বের করতে চাই, আমরা দরকারী সবগুলো variable নীচের মতো করে ঘোষনা করতে পারি, যেখানে int variableগুলো একটা statementএ (বিবৃতি) থাকবে আর float variableগুলো আলাদা আরেকটা statementএ থাকবে। মনে রেখো আমরা কিন্তু এই পাঁচটি variableএর (চলক) প্রত্যেককে আলাদা আলাদা statementএ লিখতেই পারতাম। এখানে আমরা বরং সেটা না করা নিয়েই আলোচনা করছি, উল্টো একসাথে করা নিয়ে আলোচনা করছি।

```
int length, width, perimeter; // দৈর্ঘ্য, প্রস্থ, পরিসীমা
float radius, area; // ব্যাসার্ধ ও ক্ষেত্রফল
```

১১.৪ Initial Assignment (আদিমান আরোপণ)

আগের পাঠে আমরা একাধিক variable (চলক) declare (ঘোষনা) নিয়ে আলোচনা করেছি। এখন আমরা এদের initial value (আদি মান) assign (আরোপ) করার দিকে নজর দেই। Intial value হল প্রথমবারের মতো যে মান দিয়ে দেওয়া হয় সেই মানটি। ঘোষনা দেয়ার পরে variableএ আলাদা করে আদি মান assign করতে চাইলে আমরা নীচের মতো করে করবো।

```
length = 6;
width = 3;
radius = 5;
```

অথবা চাইলে এক statementএ এক সাথেও করা সম্ভব, comma , (বির্তি) দিয়ে।

```
length = 6, width = 3, radius = 5;
```

আমরা কিন্তু চাইলে আদিমানগুলো নীচের মতো ঘোষনার সাথে সাথেই দিতে পারতাম।

```
int length = 6, width = 3, perimeter;
float radius = 5, area;
```

ঘোষনার সাথে সাথে variableএর initial value (আদিমান) assign (আরোপ) করলে programএর (ক্রমলেখ) দক্ষতা অল্প একটু বাড়তে পারে। কারণ ঘোষনার সাথে সাথে আদিমান না দিলেও একটা উল্টাপাল্টা মান তো ভিতরে ভিতরে দেয়াই হয়, পরে যখন আমরা আবার মান দেই, তখন আরেকবার দেওয়া হলো, মানে প্রথমবারেই দেওয়া হলো না। আর ঘোষনার সাথে সাথে আদিমান দিলে, একদম প্রথমবারেই মানটি variableএ দেওয়া হয়ে গেলো। Constantএর ক্ষেত্রে কিন্তু আদি ও একমাত্র মান (initial and only value) ঘোষনার সাথে সাথেই দিতে হবে, পরে দেয়ার কোন সুযোগ নাই, compiler (সংকলক) error message (ত্রুটি বার্তা) দেখাবে।

কোন variable declare করার সাথে সাথে তাতে কোন initial value না দিলেও যে উল্টাপাল্টা মান থাকে সেটা কত তা যদি জানতো চাও তবে পরীক্ষা করে দেখতে পারো। ধরো তোমার variable length। এখন ঘোষনার পরেই cout << "length is " << length << endl; লিখে program compile (সংকলন) করে run করে দেখতে পারো। কিন্তু প্রতিবার চালালে যে একই মান আসবে তার কোন নিশ্চয়তা নাই, যদি আসে সেটা নেহায়েত কাকতাল।

আমরা আগেই জানি বৃত্তের ক্ষেত্রফল নির্ণয়ের জন্য আমাদের পাইয়ের মান দরকার হবে, যেটি একটি constant (ধ্রুবক) আর পাইয়ের মান আসলেই ভগ্নক বা float। কিন্তু float হওয়া সত্ত্বেও আমরা কিন্তু পাইয়ের জন্য pai নামক variableটিকে radius আর area এর সাথে একই statementএ ঘোষনা করতে পারবো না। কারণ radius ও area হল variable (চলক) যাদের মান পরে যতবার ইচ্ছা বদলানো যাবে আর pai হল constant (ধ্রুবক) যার মান একবার দেওয়ার পরে আর বদলানো যাবে না। পাইয়ের মান তাই আলাদা করে ঘোষনা করতে হবে।

```
int length = 6, width = 3, perimeter;
float radius = 5, area;
float const pai = 3.1415; // পাইয়ের মানের জন্য ধ্রুবক
```

আমাদের যদি একাধিক float constant থাকে সেগুলোকে আবার এক statementএই ঘোষনা করতে পারবো, যেমন পাই আর জি এর মান ঘোষনা করছি নীচে। তোমরা জানো g হল মাধ্যাকর্ষনের ত্বরণের মান, যা নির্দিষ্ট স্থানে মোটামুটি একটা ধ্রুবক।

```
float const pai = 3.1415, g = 9.81;
```

পরিসীমা আর ক্ষেত্রফলের জন্য আমাদের সূত্র লিখতে হবে, সেগুলোকে comma , (বির্তি) দিয়েই এক statementএ লেখা সম্ভব, যেমন নীচে লিখলাম।

```
int length = 6, width = 3
int perimeter = length * width;
float radius = 5;
float area = pai * radius * radius;
float const pai = 3.1415;
```

উপরে যা লিখলাম তাতে কিন্তু একটা error (ত্রুটি) আছে, compile (সংকলন) করতে গেলেই error ধরা পড়বে। Errorটি হল আমরা pai declare করেছি পঞ্চম statementএ, কিন্তু pai ব্যবহার করেছি চতুর্থ statementএ area এর সূত্র লিখতে গিয়েই। কোন variable declare করার আগে সেটা ব্যবহার করা যাবে না, compiler যখন run করে তখন সে একে

একে statementগুলো উপর থেকে নীচে আর বামে থেকে ডানে পড়তে থাকে। তো compiler কোন variable বা constant declare করার আগেই যদি তাদের ব্যবহারটা পড়ে ফেলে যেমন pai, তাহলে সে বুঝতে পারবে না pai টা কী জিনিস, এইটা কি variable নাকি constant, এটা কি int ধরনের নাকি float ধরনের। আমাদের তাই declare অবশ্যই আগে করতে হবে, ব্যবহার করতে হবে পরে। তো চলো নীচে আমরা পাইয়ের declaration আগে করি।

```
int length = 6, width = 3;
int perimeter = length * width;
float const pai = 3.1415; // ঘোষনা আগে করা হলো
float radius = 5;
float area = pai * radius * radius;
```

লক্ষ্য করো length, width, radiusএর জন্য কিন্তু উপরের ওই error ঘটে নি, কারন সুত্রে ব্যবহারের আগেই তো ওগুলো declare হয়েছে, যদিও একই সারিতে কিন্তু বামের বিষয়গুলো যেহেতু ডানেরগুলোর থেকে আগে, তাই ঘোষনা আগেই হয়েছে। আমরা অবশ্য উপরের মতো করে সূত্রও একই statementএ না দিতে বলবো। তাতে পড়ারও সুবিধা হয়, আবার আগে লিখবো না পরে লিখবো সেই সমস্যাও দুর হয়। তাহলে পুরো ব্যাপারটি দাঁড়াচ্ছে নীচের মতো:

```
int length = 6, width = 3, perimeter;
float const pai = 3.1415;
float radius = 5, area;

perimeter = length * width;
area = pai * radius * radius;
```

১১.৫ Exercise Problems (অনুশীলনী সমস্যা)

Conceptual Questions: নীচে কিছু conceptual প্রশ্ন রয়েছে। প্রশ্নগুলোর উত্তর নিজে নিজে বের করবে।

১. Programএ (ক্রমলেখ) variable ও constant কেন ব্যবহার করা হয়?

২. Programএ (ক্রমলেখ) variable declaration বলতে কী বুঝ? উদাহরণ দেখাও।

৩. Variableএ (চলক) value assign (মান আরোপণ) বলতে কী বুঝ? ব্যাখ্যা করো।

৪. কখন তুমি variable (চলক) ব্যবহার না করে constant (ধ্রুবক) ব্যবহার করবে?

৫. সিপিপিতে কী ভাবে variable ও constant declare করতে হয়। উদাহরণ দেখাও।

৬. সিপিপিতে কী ভাবে int ও float ধরনের variable declare করতে হয়?

৭. সিপিপিতে এক সারিতে কখন একাধিক variable declare করা যায়? উদাহরণ দেখাও।

৮. Variableএ (চলক) initial value assignment (আদিমান আরোপণ) কী?

৯. Variableএ (চলক) initial value assign না করলে সম্ভাব্য কী ফলাফল ঘটতে পারে?

১০. Constantএ কেন initial value assign করতে হয়, কিন্তু পরে assign করা যায় না?

১১. ফলাফল float (ভগ্নক) কিন্তু int (পূর্ণক) ধরনের variableএ assign করলে কী ঘটে?

Programming Problems: নীচে আমরা কিছু programming problem দেখবো। এই problemগুলো আগে ধৈর্য্য ধরে নিজে নিজে সমাধান করতে চেষ্টা করবে। যখন একেবারেই পারছো না বলে মনে হয় তখনই কেবল সমাধান দেখে নিতে পারো। সমাধানগুলো programming problemগুলোর শেষে আছে।

১. এমন একটি program রচনা করো যেটি দুটি int ধরনের, আর একটি float ধরনের variable declare করে। Programটি তারপর variable তিনটির মান যথাক্রমে 10, 15, 12.6 assign করে। পরিশেষে programটি variableগুলোর মান পর্দায় দেখায়।

২. ধরো দুটো পূর্ণ সংখ্যা 89 আর 56। এই দুটিকে তুমি দুটো variableএ নিবে, আর তারপর দুইটি variableএ তাদের যোগফল, বিয়োগফল নির্ণয় করবে। সবশেষে সবগুলো varaibleএর মান outputএ দেখাবে। সব মিলিয়ে এমন একটি program লিখো।

৩. যদি তাপমাত্রা সেলসিয়াসে c ডিগ্রী হয় আর ফারেনহাইটে হয় f ডিগ্রী, তাহলে আমরা লিখতে পারি $f = 9c/5 + 32$। ধরো তাপমাত্রা সেলসিয়াসে দেয়া আছে, তাহলে ফারেনহাইটে এটি কত হবে? তোমার programএ তুমি float ধরনের variable ব্যবহার করবে।

৪. যদি তাপমাত্রা ফারেনহাইটে হয় f ডিগ্রী আর সেলসিয়াসে হয় c ডিগ্রী, তাহলে লেখা যায় $c = 5(f - 32)/9$। ধরো তাপমাত্রা ফারেনহাইটে দেয়া আছে, তাহলে সেলসিয়াসে এটি কত হবে? তোমার programএ তুমি float ধরনের variable ব্যবহার করবে।

৫. ধরো একটা কাজ করতে তোমার 7 ঘন্টা 15 মিনিট 39 সেকেন্ড লেগেছে। এই সময়কে সেকেন্ডে রূপান্তর করো। তোমার programএ তুমি 60 সেকেন্ডে এক মিনিট আর 60 মিনিটে এক ঘন্টা এই দুটি বিষয় বুঝানোর জন্য দুটো constant ব্যবহার করবে।

Programming Solutions: এবার আমরা programming problemগুলোর সমাধান দেখবো। মনে রাখবে সমাধানগুলো দেয়া হয়েছে তুমি যদি একান্তই নিজে নিজে করতে পারছোনা বলে মনে হয়, তখন কেবল একটু সাহায্য যাতে পেতে পারো তাই। কাজেই পারতপক্ষে সমাধান দেখবে না, নিজে নিজে সমাধানের সবটা চেষ্টা আগে করে দেখবে।

১. এমন একটি program রচনা করো যেটি দুটি int ধরনের, আর একটি float ধরনের variable declare করে। Programটি তারপর variable তিনটির মান যথাক্রমে 10, 15, 12.6 assign করে। পরিশেষে programটি variableগুলোর মান পর্দায় দেখায়।

ফিরিস্তি ১১.৩: Program Declaring Variables (চলক ঘোষনার ক্রমলেখ)

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
```

```
{
  int intvar1, intvar2; //  পূর্ণক দুটি এক সাথে ঘোষনা
  float floatvar;       //  ভগ্নকটি আলাদা ঘোষনা

  intvar1 = 10, intvar2 = 15; // মান আরোপণ
  floatvar = 12.6;            //  মান আরোপণ

  cout << "integers are ";     // endl দেই নাই
  cout << intvar1 << " " << intvar2 << endl;
  cout << "float is " << floatvar << endl;

  return EXIT_SUCCESS;  //  সফল সমাপ্তি
}
```

ফলন (output)

```
purnok duti 10 15
vognok holo 12.6
```

২. ধরো দুটো পূর্ণ সংখ্যা 89 আর 56। এই দুটিকে তুমি দুটো variableএ নিবে, আর তারপর দুইটি variableএ তাদের যোগফল, বিয়োগফল নির্ণয় করবে। সবশেষে সবগুলো varaibleএর মান outputএ দেখাবে। সব মিলিয়ে এমন একটি program লিখো।

আমরা এই programএ কেবল দরকারী অংশটুকু দেখাচ্ছি। ধরে নিচ্ছি যে তুমি দরকারী header file (শির নথি) include (অন্তর্ভুক্ত) করা, namespace (নামাধার) std ব্যবহার, main funciton (বিপাতক) লেখা ও মান return করা ইত্যমধ্যে ভালো করে শিখে ফেলেছো। তো তুমি যদি সত্যি নীচের লেখা program compile করে run করতে চাও, তোমাকে কিন্তু আগে include, namespace, main, return ওইগুলো লিখে নিতে হবে, তারপর main functionএর ভিতরে return এর আগে তুমি আমাদের নীচের অংশগুলো লিখে নিবে। তারপর compile করে program চালাবে।

ফিরিস্তি ১১.৪: Arithmetic Program (পাটিগণিতের অণুক্রিয়ার ক্রমলেখ)

```
int first = 89, second = 56;

int sum = first + second;
int diff = first - second;

cout << "first is " << first;
cout << " second is " << second;
cout << endl;

cout << "sum is " << sum;
cout << " diff is " << diff;
cout << endl;
```

৩. যদি তাপমাত্রা সেলসিয়াসে c ডিগ্রী হয় আর ফারেনহাইটে হয় f ডিগ্রী, তাহলে আমরা লিখতে পারি $f = 9c/5 + 32$। ধরো তাপমাত্রা সেলসিয়াসে দেয়া আছে, তাহলে ফারেনহাইটে এটি কত হবে? তোমার programএ তুমি float ধরনের variable ব্যবহার করবে।

ধরে নিচ্ছি প্রথম সমস্যার সমাধান দেখে তুমি programএর কাঠামো দাঁড় করতে পারবে।

ফিরিস্তি ১১.৫: Celcius to Fahrenheit (সেলসিয়াস থেকে ফারেনহাইটে রূপান্তর)

```
float c = 30, f = 9 * c / 5 + 32;
```

৪. যদি তাপমাত্রা ফারেনহাইটে হয় f ডিগ্রী আর সেলসিয়াসে হয় c ডিগ্রী, তাহলে লেখা যায় $c = 5(f - 32)/9$। ধরো তাপমাত্রা ফারেনহাইটে দেয়া আছে, তাহলে সেলসিয়াসে এটি কত হবে? তোমার programএ তুমি float ধরনের variable ব্যবহার করবে।

ধরে নিচ্ছি প্রথম সমস্যার সমাধান দেখে তুমি programএর কাঠামো দাঁড় করতে পারবে।

ফিরিস্তি ১১.৬: Fahrenheit to Celcius (ফারেনহাইট থেকে সেলসিয়াসে রূপান্তর)

```
float f = 76, c = 5*(f - 32) / 9;
```

৫. ধরো একটা কাজ করতে তোমার 7 ঘন্টা 15 মিনিট 39 সেকেন্ড লেগেছে। এই সময়কে সেকেন্ডে রূপান্তর করো। তোমার programএ তুমি 60 সেকেন্ডে এক মিনিট আর 60 মিনিটে এক ঘন্টা এই দুটি বিষয় বুঝানোর জন্য দুটো constant ব্যবহার করবে।

ধরে নিচ্ছি প্রথম সমস্যার সমাধান দেখে তুমি programএর কাঠামো দাঁড় করতে পারবে।

ফিরিস্তি ১১.৭: Convert Time to Seconds (সময়কে সেকেন্ডে রূপান্তর)

```
int hour = 7, minute = 15, second = 39;
int const hourMinute = 60, minuteSecond = 60;

int totalMin = hour * hourMinute + minute;
int totalSec = totalMin * minuteSecond + second;
```

১১.৬ Computing Terminologies (গণনা পরিভাষা)

- Initial Value (আদি মান)
- Assign (আরোপণ)
- Declaration (ঘোষনা)
- Variable (চলক)
- Constant (ধ্রুবক)
- Fraction (ভগ্নক)
- Value (মান)
- Value Assign (মান আরোপণ)
- Floating-Point (সচলবিন্দু)
- Var Declaration (চলক ঘোষণা)

অধ্যায় ১২

Naming Identifiers
(শনাক্তকের নামকরণ)

নামে কী আসে যায় কর্মে পরিচয়। আপনার কাজই নির্ধারণ করে দেবে আপনার পরিচয়। আপনার নাম পরিচয় হবে আপনার কাজের কারণেই। Program (ক্রমলেখ) লিখতে গিয়ে আমরা তাই variable (চলক), constant (ধ্রুবক), function (বিপাতক) সহ যে কোন কিছুর নাম দেই তাদের কী কাজে লাগানো হবে সেটা মাথায় রেখে।

১২.১ Well-formed Names (সুগঠিত নাম)

সিপিপিতে variable ও constantএর ব্যবহার তুমি ইত্যমধ্যে শিখে ফেলেছো। আর সাথে সাথে variableএর নাম কী রকম দিতে হবে সেটাও আগে একটু জেনেছো। এখন আমরা বিস্তারিত ভাবে শিখব সিপিপিতে কী ভাবে variable বা constantএর নাম দিতে হয়, বিশেষ করে নামের syntax (গঠনরীতি) কেমন অর্থাৎ নামে কী রকম অক্ষর থাকতে পারবে অথবা পারবে না। আমরা আপাতত কেবল main function (বিপাতক) নিয়ে কাজ করছি। কিন্তু ভবিষ্যতে আমরা যখন নিজেদের জন্য নানান function তৈরী করবো, তখন functionএর নামকরণের জন্যেও constant বা variableএর নাম তৈরীর নিয়মগুলোই কাজে লাগবে। Variable বা constant বা function যাইহোক নাম কে বলা হয় identifier (শনাক্তক)।

সিপিপিতে কোন identifierএর (শনাক্তকের) নামে কেবল ১) ইংরেজী বর্ণমালার বড় হাতের অক্ষর A-Z, ২) ইংরেজী বর্ণমালার ছোট হাতের অক্ষর a-z, ৩) ইংরেজী অংক 0-9 আর ৪) underscore (নিম্নদাগ) _ থাকতে পারবে। তবে identifierএর নামের প্রথম অক্ষর আবার অংক 0-9 হতে পারবে না, প্রথম অক্ষর ছাড়া অন্য যে কোন অক্ষর হিসাবে অংকগুলো ব্যবহার করা যাবে। সুতরাং বোঝাই যাচ্ছে প্রথম অক্ষর যে কোন বর্ণ A-Z বা a-z অথবা underscore (নিম্নদাগ) _ হতে পারবে। আর তারপরের যে কোন অক্ষর বর্ণ বা অংক বা underscore হতে পারবে। সিপিপিতে identifierএর নামের length (দৈর্ঘ্য) নিয়ে কোন বিধিনিষেধ নেই তবে program (ক্রমলেখ) compileএ (সংকলন) কী compiler (সংকলক) ব্যবহার করা হচ্ছে তার ওপর এটা নির্ভর করতে পারে। cpp.sh দিয়ে compile করলে কোন বিধি নিষেধ নেই, মাইক্রোসফট c++ দিয়ে compile করলে ২০৪৮ অক্ষর পর্যন্ত হতে পারে। যাইহোক আমরা এখানে syntax অনুযায়ী legal ও illegal কিছু নাম দেখবো।

অবৈধনাম	কারণ
12	নামের সবগুলোর অক্ষর অংক হতে পারবে না
12cholok	নামের প্রথম অক্ষর অংক হতে পারবে না
amar cholok	নামের মাঝখানে কোন ফাঁকা (space) থাকতে পারবে না
ama;cho+k	বর্ণ, অংক, নিম্নদাগ ছাড়া অন্য কোন প্রতীক থাকতে পারবে না

Programএ (ক্রমলেখ) অবৈধ নাম ব্যবহার করলে কী হয়? করে দেখো কী হয়! Compiler (সংকলক) error message (ত্রুটিবার্তা) দিবে, আর তোমাকে নামটি ঠিক করতে হবে। তাহলে এখন থেকে তোমার programএ নাম দেওয়ার সময় নামের এই গঠনরীতি গুলো মেনে চলবে।

বৈধনাম	কারণ
p	একটাই অক্ষর সেটি ছোট হাতের বর্ণ
P	একটাই অক্ষর সেট বড় হাতের বর্ণ
abc	তিনটা অক্ষর সব ছোট হাতের বর্ণ
ABC	তিনটা অক্ষর সব বড় হাতের বর্ণ
Abc	তিনটা অক্ষর ছোটহাতের বড়হাতের মিশানো
bAc	তিনটা অক্ষর ছোটহাতের বড়হাতের মিশানো
a1bc	তিনটা ছোটহাতের অক্ষর ও একটা অংক, অংকটি শুরুতে নয়
a1Bc	তিনটা ছোটবড় হাতের অক্ষর ও একটা অংক যেটি শুরুতে নয়
a_bc	তিনটা ছোটহাতের অক্ষর ও একটি নিম্নদাগ (underscore)
_abc	তিনটা ছোট হাতের অক্ষর ও তিনটি নিম্নদাগ
_A_b_c	তিনটা ছোটবড় হাতের অক্ষর ও তিনটি নিম্নদাগ
amar_cholok	ছোটহাতের অক্ষর ও নিম্নদাগ, নামটি অধিক বোধগম্য
_amar_Cholok	ছোটবড় হাতের অক্ষর ও নিম্নদাগ, অধিক বোধগম্য
_amarCholok123	ছোটবড় হাতের অক্ষর, নিম্নদাগ, ও অংক যেটি শুরুতে নয়
amar125cholok	ছোটহাতের অক্ষর ও অংক, অংকটি শুরুতে নয়।

১২.২ Meaningful Names (অর্থবোধক নাম)

সিপিপিতে identifierএর (শনাক্তক) নাম কেমন হতে পারে আর কেমন হতে পারে না, আমরা তা আগের পাঠে দেখেছি। এই পাঠে আমরা দেখবো নামের অর্থবোধকতা (semantic)। আমরা যখন কোন নাম দেবো, তখন নামটি অবশ্যই অর্থবহ হওয়া চাই। আমরা আগের একটি পাঠে অল্প একটু আলোচনা করেছি নামের অর্থবোধকতা নিয়ে। এখন আরো বিস্তারিত আলোচনা করছি নামগুলো কেমন হলে ভালো হয় সে সম্পর্কে। Variable (চলক) বা constant (ধ্রুবক) বা function (বিপাতক) নাম সবসময় তার কাজ ও ব্যবহারের দিকে খেয়াল রেখে অর্থবোধক হওয়া উচিৎ। অর্থবোধক না হলে program (ক্রমলেখ) বোঝা আমাদের জন্য কঠিন হয়ে যায়।

অনেকে অতিরিক্ত আগ্রহে যত্রতত্র নিজের বা প্রিয় কারো নামে identifierএর নাম দিয়ে থাকে যেমন gonimia1, gonimia2, ইত্যাদি। তো এই variable দুটোর একটা যদি ব্যাসার্ধের জন্য আরেকটা যদি ক্ষেত্রফলের জন্য ব্যবহার করা হয়, তাহলে variableএর নাম থেকে মোটেও বুঝা যাবে না কোন নামটি কী কাজে ব্যবহৃত হচ্ছে। ব্যাসার্ধের জন্য বরং radius বা bashardho অথবা নিদেনপক্ষে r বা b ব্যবহার করা যেতে পারে। এক অক্ষরের নাম দেয়া অনেকে পছন্দ করে, কারণ তাড়াতাড়ি লেখা যায়, কিন্তু একই আদ্যাক্ষর যুক্ত একাধিক variable থাকলে তখন মুশকিল হয়ে যায়। সেক্ষেত্রে ওই অক্ষরের সাথে আরো অক্ষর লাগিয়ে অথবা সংখ্যা লাগিয়ে প্রতিটি নামকে আলাদা করতে হবে, যাতে অন্তত বুঝা যায় কোন variableটি কী উদ্দেশ্যে ব্যবহার করা হয়েছে।

আমরা যদি দুটো বৃত্ত নিয়ে কাজ করি তাহলে তাদের ব্যাসার্ধের জন্য দুটি variable হতেই পারে radius1 আর radius2 তাতে কোন সমস্যা নাই। ব্যাপারটা দীপু নম্বর ২ চলচ্চিত্রের মতো, একজনের নাম দীপু নম্বর ১ আর একজন দীপু নম্বর ২। অথবা কেউ চাইলে নাম দিতে পারে radiusA আর radiusB। এভাবে একই ধরনের কাজে ব্যবহার হবে এরকম variable অনেকগুলো লাগলে আমরা সংখ্যা লাগিয়ে বা বর্ণ লাগিয়ে আলাদা আলাদা নাম তৈরী করে নিবো। এর জন্য অবশ্য array (সাজন) নামে আলাদা একটা ধারণা আছে, যেটা আমরা পরে জানবো। Array ব্যবহার করে আমরা সংখ্যা লাগিয়ে যত ইচ্ছা ততগুলো একই ধরনের নাম পাই। অনেকে আলসেমি করে অথবা কোন কারণে identifierএর (শনাক্তক) নাম করণ করতে থাকে a, b, c, p, q, r , i , j , k, x, y, z ইত্যাদি একের পর এক অক্ষর দিয়ে। এটা খুবই বাজে অভ্যাস। এইরকম identifier মোটেও অর্থবোধক নয়। এগুলো থেকে বুঝার কোন উপায় নেই কোন variableটি ঠিক কী কাজে ব্যবহার করা হচ্ছে। সবসময় এরকম নামকরণ থেকে দুরে থাকবে।

এখানে প্রশ্ন করতে পারো: নামকরণে কি সবসময় একটা মাত্র শব্দই ব্যবহার করবো? একের অধিক শব্দ ব্যবহার করবো না? উত্তর হচ্ছে অর্থবোধক করার জন্য তুমি দরকার মতো একাধিক শব্দ অবশ্যই ব্যবহার করবে, এইটা খুবই ভালো অভ্যাস। আর সেক্ষেত্রে যাতে প্রতিটি শব্দ খুব সহজে বোঝা যায় সে জন্য তোমার কিছু কৌশল অবলম্বন করতে হবে। একটা কৌশল হলো দুটি শব্দের মাঝে একটি underscore (নিম্নদাগ) _ দেওয়া যেমন my_var। আরেকটি কৌশল হল প্রতিটি শব্দের প্রথম অক্ষরটি বড়হাতের দেওয়া আর অন্যগুলো ছোট হাতের, যেমন MyVar তবে চাইলে একদম প্রথম শব্দের প্রথম অক্ষরটি ছোটহাতেরও রাখতে পারো যেমন myVar। নীচের সারণীতে আমরা কিছু অর্থবোধক নামের উদাহরণ দেখবো।

নাম	যথোপযুক্ততার কারণ
sum	যোগফলের জন্য sum চলকের ইংরেজী নাম
jogfol	যোগফলের জন্য jogfol চলকের বাংলা নাম
bijor_songkhar_jogfol	নিম্নদাগ _ দিয়ে অর্থবোধক শব্দ আলাদা হয়েছে
odd_number_sum	নিম্নদাগ _ দিয়ে অর্থবোধক শব্দ আলাদা হয়েছে
Bijor_Shongkhar_Jogfol	নিম্নদাগ _ দিয়ে আলাদা, বড়হাতের আদ্যাক্ষর
BijorShongkharJogfol	বড়হাতের প্রথম অক্ষর দিয়ে আলাদা আলাদা
bijorShongkharJogfol	এটি অনেক প্রচলিত ও অনেকেরই পছন্দের

১২.৩ Case Sensitivity (লিপি সংবেদনশীলতা)

সিপিপি ভাষা একটি case sensitive (লিপি সংবেদনশীল) ভাষা। এই কথার অর্থ কী? সিপিপিতে বড়হাতের ছোটহাতের অক্ষর কি ভিন্নভিন্ন ধরা হয়, নাকি ইংরেজীর মতো একই ধরা হয়?

```
barek is going home
BAREK IS GOING HOME
Barek Is Going Home
```

আগের কয়েকটি পাঠে variable (চলক) বা constant (ধ্রুবক) বা functionএর (বিপাতক) নাম, এককথায় identifierএর (শনাক্তক) নামকরণ নিয়ে আমরা আলোচনা করেছি। নামকরণের নিয়মগুলো আলোচনা করার সময় দেখেছি যে কোন শনাক্তকের নামকরণে আমরা চাইলে বড়হাতের বর্ণ A-Z, ছোটহাতের বর্ণ a-z, অংক 0-9, আর নিম্নদাগ _ ব্যবহার করতে পারবো। একই নামে বড়হাতের ছোটহাতের অক্ষর মিশিয়েও নামকরণ করতে পারবো। এইঅবস্থায় প্রশ্ন হচ্ছে কোন নাম ইচ্ছামতো একবার বড়হাতের অক্ষরে অথবা ছোট হাতের অক্ষরে অথবা আরেকবার কিছু অক্ষর

ছোটহাতের কিছু অক্ষর বড় হাতের এইভাবে লিখতে পারবো কিনা। বিশেষ করে আমরা জানি ইংরেজীতে আমি ছোট হাতেরই লিখি আর বড় হাতেরই লিখি শব্দটা আসলে একই থাকে, সিপিপিতেও কি তাই? আমরা বরং উদাহরণ দিয়ে ব্যাপারটা দেখি। ইংরেজীতে ছোট হাতের বড় হাতের অক্ষর আলাদা হলেও ওগুলো কেবলই সৌন্দর্যবর্ধন মুলক। উপরের তিনটে ইংরেজী বাক্য তাই একই।

এবার আমরা সিপিপি ভাষায় ছোট হাতের বড় হাতের অক্ষরের ব্যবহার দেখি। নীচের নামগুলোর প্রত্যকেটি সিপিপি ভাষায় আলাদা আলাদা নাম হিসাবে ধরা হবে।

```
myvariable, myVariable, MyVariable, my_variable,
My_Variable, myVariable, MyVaRiAbLe
```

সিপিপিতে উপরের একটা নাম দিয়ে যে variable বা constant বা functionকে বুঝানো হবে অন্য নাম দিয়ে ওইটাকে বুঝানো যাবে না, বরং অন্য একটা বুঝানো হয়ে যাবে। মোট কথা দুটো নামের একটা অক্ষরেও যদি এদিক সেদিক থাকে তাহলে নামদুটো আসলে আলাদা। দুটোকে একই জিনিসের নাম হিসাবে ধরে নেয়া যাবে না। সুতরাং program (ক্রমলেখ) লেখার সময় খেয়াল রাখবে যাতে একটা variableকে বুঝাতে গিয়ে কেবল বড়হাতের ছোটহাতের বর্ণের ভিন্নতার কারণে আরেকটাকে বুঝিয়ে না ফেলো, তাতে সব ভজঘট লেগে যাবে। তোমার programও উল্টাপাল্টা ফলাফল দিবে। আবার ধরো তোমার একটাই variable যার নাম myvariable, কিন্তু পরে তুমি লিখেছো myVariable। এই অবস্থায় compile (সংকলন) করলে তোমাকে "myVariable is not declared" এইরকম error message (ত্রুটিবার্তা) দিবে। তোমাকে তখন myVariable এর বদলে myvariable লিখে ঠিক করতে হবে। Program তৈরীর সময় আমরা প্রায়শই এইরকম ভুল করে থাকি। আর তখন আমাদের এইভাবে ঠিক করে নিতে হয়।

উপরের এই নিয়ম জানার পরে তুমি হয়তো মনে করবে এইটা তো ভালোই। আমার যদি দুইটা বৃত্তের ব্যাসার্ধের জন্য চলক লাগে একটার নাম দিবো radius আর একটার নাম দিবো Radius। হ্যাঁ, তুমি সেটা দিতেই পারো। সিপিপি যেহেতু দুইটাকে আলাদা আলাদা variable হিসাবে ধরে নিবে, তাই এই দুটো হলো দুটো বৈধ আলাদা নাম। তবে অর্থবোধকতার দিক ভেবে তুমি হয়তো এরকম নাম করণ থেকে দূরে থাকার চেষ্টা করবে। একটা অক্ষর বড় বা ছোটহাতের কেবল এই অল্প একটুখানি ভিন্নতা দিয়ে আসলে তেমন বেশী অর্থবোধক পার্থক্য তৈরী করা যায় না, ফলে program (ক্রমলেখ) পড়া কঠিন হয়। আর একটা ব্যাপার: variableএর নামকরণে বড়হাতের ছোটহাতের অক্ষর মিশাতে তো পারোই যেমন MyVariable, কিন্তু এমন ভাবে মিশিও না যে পড়াটা খুব কঠিন হয়ে যায়, যেমন MyVaRiAbLe, এই রকম নাম চট করে পড়া আসলে সম্ভব না, বরং এইরকম নাম যন্ত্রনাদায়ক। কাজেই সবমিলিয়ে সহজ ও সুন্দর নাম দিবে, কেমন!

১২.৪ Reserved & Key Words (সংরক্ষিত ও চাবি শব্দ)

Reserved word (সংরক্ষিত শব্দ) বা key word (চাবি শব্দ) কী? আমি কি variable (চলক), constant (ধ্রুবক) বা functionএর (বিপাতক) এর identifier হিসাবে reserved word বা key word ব্যবহার করতে পারবো? সিপিপিতে reserved বা key word কোনগুলো?

Reserved word বিষয়ে আলোচনার আগে আমরা একটা গল্প বলে নেই। এক বাড়িতে থাকে জামাই-বউ আর তাদের সাথে থাকে বড় কুটুম অর্থাৎ বউয়ের ভাই বা জামাইয়ের শ্যালক। তো সেই শ্যালকের নাম হল দুলাল। একদিন জামাই বেচারা তার বউয়ের কষ্ট লাঘব করার জন্য একজন কাজের ছেলে নিয়ে আসে। বউ জিজ্ঞেস করে "এই ছেলে তোমার নাম কী?" কাজের ছেলে বলে তার নাম দুলাল। বউ তখন জামাইকে বলে ছেলেটির নাম বদলাতে হবে। জামাই অবাক, অবাক কাজের ছেলেটিও। তার নাম দুলাল, ভালোই তো নামটি, সেটা বদলাতে হবে কেন। বউ

জামাইকে বকতে থাকে "তুমি জানো না আমার ভাই অর্থাৎ তোমার শ্যালকের নাম দুলাল"। যে বাসায় শ্যালকের নাম দুলাল, সেই বাসার কাজের ছেলের নাম দুলাল হয় কেমনে, শ্যালক হলো বড় কুটুম, তার কী এত বড় অসম্মান করা যায়! আর জামাইয়ের নাম হলো কাদের। তো বউ আরো এক কাঠি বাড়িয়ে বলতে থাকে ঠিক আছে কাজের ছেলের নাম বদলে কাদের রাখা হউক, দেখি জামাইয়ের কেমন লাগে। তারপর জামাইয়ের সামনেই কাজের ছেলেকে বলে "এই এখন থেকে তোর নাম দিলাম কাদের।" তারপর হেঁড়ে গলায় ডাকতে থাকে "কাদের, এই কাদের, এই দিকে আয়।" কেমন একটা বেড়াছেড়া অবস্থা। শেষ পর্যন্ত ঠিক হয় এক বাসায় দুইটা দুলাল তো হতে পারেনা, একজনের নাম বদলাতে হবে। আর বাসার বড় কুটুমের নাম তো আর বদলানো যাবে না কোন ভাবেই, ওটা সংরক্ষিত নাম, কাজেই বদলাতে হবে কাজের ছেলের নাম। সুতরাং কাজের ছেলের নাম দেয়া হয় দুলাল্যা। তাহলে শ্যালকের নাম দুলাল, আর কাজের ছেলের নাম দুলাল্যা।

সিপিপি ভাষায় গঠন কাঠামো ঠিক রাখার জন্য কিছু সুনির্দিষ্ট শব্দ আছে। আমরা ইত্যমধ্যে এরকম কিছু শব্দ ব্যবহার করেছি। যেমন return, int, float। এই শব্দগুলোর অর্থ সিপিপি ভাষাতে আগে থেকে সুনির্দিষ্ট, যেমন return মানে যখন function (বিপাতক) শেষ হয়, int আর float হল variableএর মান কেমন পূর্ণক বা পূর্ণ সংখ্যা না ভগ্নক বা ভগ্ন সংখ্যা এইরকম। এই তিনটি ছাড়াও আরো অনেকগুলো এই রকম শব্দ আছে। এই শব্দগুলো চাইলে আমরা নিজেরা আমাদের variable (চলক) বা constant (ধ্রুবক) বা functionএর (বিপাতক) নাম হিসাবে ব্যবহার করতে পারবো না। এইগুলো হচ্ছে reserved word (সংরক্ষিত শব্দ)। এই শব্দগুলোকে অন্য কথায় key wordও (চাবি শব্দ) বলা হয়। তাহলে তোমার ক্রমলেখতে তুমি এইরূপ reserved word বা key word identifierএর (শনাক্তক) নাম হিসাবে ব্যবহার করবে না। কারণ ওগুলো বড়কুটুম দুলালের নামের মতো। যদি একান্তই দরকার হয় তাহলে দুলাল কে দুলাল্যা বানানোর মতো কিছু যোগ-বিয়োগ করে ভিন্ন শব্দ বানিয়ে ব্যবহার করবে। যেমন return না ব্যবহার করে returnValue ব্যবহার করলে, এইরকম। নীচে আমরা সিপিপির reserved wordগুলোর তালিকা দিচ্ছি।

- Structure programmingএ (সংগঠিত পরিগণনায়) ব্যবহৃত শব্দ:
  break, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, int, long, register, short, signed, sizeof, static, struct, switch, typedef, union, unsigned, return, void, volatile, while

- Object-oriented (বস্তুমুখী পরিগণনায়) ব্যবহৃত শব্দ:
  class, explicit, delete, friend, inline, mutable, namespace, new, opeator, private, protected, public, this, using, virtual

- Error handlingএর (ত্রুটি সামলানো) জন্য শব্দ:
  catch, noexcept, throw, try

- Logical and bit-wise operators (যুক্তি ও বিটপ্রতি অণুক্রিয়ার) শব্দ:
  bool, and, and_eq, bitand, bitor, compl, false, not, not_eq, or, or_eq, true, xor, xor_eq

- Data type (উপাত্ত প্রকরণ) সংক্রান্ত শব্দ:
  auto, const_cast, decltype, nullptr, dynamic_cast, reinterpret_cast, static_cast, typeid

- Template (ছাঁচ) সংক্রান্ত শব্দ:
  export, template, typename
- Compile-timeএ (সংকলন সময়) ব্যবহৃত হওয়া শব্দ:
  static_assert, constexpr
- Preprocessorএর (পূর্ব প্রক্রিয়ক) জন্য শব্দ:
  if, elif, else, endif, defined, ifdef, ifndef, define, undef, include, line, error, pragma
- বিভিন্ন আকারের অক্ষরের জন্য শব্দ: char16_t, char32_t, wchar_t
- বিবিধ শব্দ: alignas, alignof, asm, concept, requires, thread_local

১২.৫ Exercise Problems (অনুশীলনী সমস্যা)

Conceptual Questions: নীচে কিছু conceptual প্রশ্ন রয়েছে। প্রশ্নগুলোর উত্তর নিজে নিজে বের করবে।

১. Identifier (শনাক্তক) কী? Programএ identifierএর ভুমিকা কী?

২. সিপিপিতে identifierএর (শনাক্তক) নাম করণের নিয়মাবলী বর্ণনা করো।

৩. Programএ (ক্রমলেখ) গঠনগত ভাবে অবৈধ নাম ব্যবহার করলে কী ঘটে?

৪. অর্থবোধক নাম কী? Programএ অর্থবোধক নাম ব্যবহার করা উচিৎ কেন?

৫. সিপিপি একটি case sensitive (লিপি সংবেদনশীল) ভাষা, এর মানে কী ?

৬. Reserved ও key word কী? এগুলো কেন identifier হিসাবে ব্যবহার করা যায় না?

Exercise Questions: নীচের শব্দগুলো syntactically (গঠনগত ভাবে) identifierএর (শনাক্তক) নাম হিসাবে বৈধ নাকি অবৈধ? যদি বৈধ হয় তাহলে অর্থবোধক (meaningful) নাকি অর্থবোধক নয়? অথবা কোন শব্দ কি reserved বা key word (সংরক্ষিত বা চাবি শব্দ)? প্রথমে নিজে নিজে উত্তর বের করার চেষ্টা করবে, একান্ত না পারলে নীচের সমাধান দেখবে।

১. void
২. MAX_ENTRIES
৩. double
৪. time
৫. G
৬. Sue's
৭. return
৮. cout
৯. xyz123
১০. part#2
১১. "char"
১২. #include
১৩. a_long_one
১৪. _xyz
১৫. 9xyz
১৬. main
১৭. mutable
১৮. max?out

১৯. Name	২৪. _SUM	২৯. int
২০. name	২৫. sum_of_numbers	৩০. pow
২১. name_1	২৬. firstName	৩১. $sum
২২. Int	২৭. Identifier	৩২. num^2
২৩. INT	২৮. printf	৩৩. num 1

Exercise Answers: উপরের প্রশ্নগুলোর উত্তর এখানে দেয়া হচ্ছে। প্রথমে নিজে নিজে উত্তর বের করার চেষ্টা করবে, একান্ত না পারলে এই সমাধান দেখবে।

১. void : Reserved word (সংরক্ষিত শব্দ), কোন প্রকারেরই না এমন বুঝানো হয়

২. MAX_ENTRIES : বৈধ identifier (শনাক্তক), অর্থবোধক

৩. double : Reserved word (সংরক্ষিত শব্দ), বড় আকারের ভগ্নকের জন্য

৪. time : বৈধ identifier (শনাক্তক), কিন্তু ভাষালয়ে (library) বিদ্যমান, তাই পরিত্যাজ্য

৫. G : বৈধ identifier (শনাক্তক), কিন্তু পারিপার্শ্বিকতা ছাড়া অর্থ বুঝা যাচ্ছে না

৬. Sue's : অবৈধ identifier (শনাক্তক) কারণ নামে ' ব্যবহার করা যায় না

৭. return : Reserved word (সংরক্ষিত শব্দ), বিপাতক থেকে ফেরত গমন

৮. cout : বৈধ identifier (শনাক্তক), কিন্তু ভাষালয়ে (library) বিদ্যমান, তাই পরিত্যাজ্য

৯. xyz123 : বৈধ identifier (শনাক্তক), কিন্তু অর্থবোধক কিনা পরিস্কার না

১০. part#2 : অবৈধ identifier (শনাক্তক) কারণ নামে # ব্যবহার করা যায় না

১১. "char" : অবৈধ identifier (শনাক্তক) কারণ নামে " ব্যবহার করা যায় না

১২. #include : Preprocessorএর (পূর্ব-প্রক্রিয়ক) জন্য reserved word

১৩. a_long_one : বৈধ identifier (শনাক্তক), কিন্তু অর্থ সেই ভাবে পরিস্কার নয়।

১৪. _xyz : বৈধ (identifier) শনাক্তক, কিন্তু অর্থ সেই ভাবে পরিস্কার নয়

১৫. 9xyz : অবৈধ (identifier) শনাক্তক, নামের শুরুতে অঙ্ক থাকতে পারে না

১৬. main : Reserved word নয়, কিন্তু প্রত্যেক programএই থাকে বলে পরিত্যাজ্য

১৭. mutable : সংরক্ষিত শব্দ, কোন constantও বিশেষ অবস্থায় পরিবর্তন যোগ্য হলে

১৮. max?out : অবৈধ identifier (শনাক্তক), নামে ? চিহ্ন থাকতে পারবে না

১৯. Name : বৈধ identifier (শনাক্তক), অর্থবোধক, কীসের নাম সেটা পরিস্কার নয়

২০. name : বৈধ identifier (শনাক্তক), অর্থবোধক, কীসের নাম সেটা পরিস্কার নয়

২১. name_1 : বৈধ identifier (শনাক্তক), অর্থবোধক, কীসের নাম সেটা পরিস্কার নয়

২২. Int : বৈধ identifier, তবে সংরক্ষিত শব্দ int এর সাথে বিভ্রান্তি দেখা দিতে পারে

২৩. INT : বৈধ identifier, তবে সংরক্ষিত শব্দ int এর সাথে বিভ্রান্তি দেখা দিতে পারে

২৪. _SUM : বৈধ identifier (শনাক্তক), অর্থবোধক, যোগফলের জন্য

২৫. sum_of_numbers : বৈধ identifier (শনাক্তক), অর্থবোধক

২৬. firstName : বৈধ identifier (শনাক্তক), অর্থবোধক, অনেকের পছন্দের

২৭. Identifier : বৈধ identifier (শনাক্তক), অর্থবোধক, কীসের শনাক্তক পরিস্কার নয়

২৮. printf : বৈধ identifier, অর্থবোধক, ভাষালয়ে (library) বিদ্যমান, পরিত্যাজ্য

২৯. int : Reserved word (সংরক্ষিত শব্দ), পূর্ণক উপাত্ত ধারনের জন্য উপাত্ত প্রকরণ

৩০. pow : বৈধ identifier, অর্থবোধক, কিন্তু ভাষালয়ে (library) বিদ্যমান, তাই পরিত্যাজ্য

৩১. $sum : অবৈধ identifier (শনাক্তক), নামে $ চিহ্ন ব্যবহার করা যায় না

৩২. num^2 : অবৈধ identifier (শনাক্তক), নামে ^ চিহ্ন ব্যবহার করা যায় না

৩৩. num 1 : অবৈধ identifier (শনাক্তক), নামে ফাঁকা ব্যবহার করা যায় না

১২.৬ Computing Terminologies (গণনা পরিভাষা)

- Identifier (শনাক্তক)
- Underscore (নিম্নদাগ)
- Array (সাজন)
- Structured (সংগঠিত)
- Programming (পরিগণনা)
- Structured programming (সংগঠিত পরিগণনা)
- Object-oriented (বস্তুমুখী)
- Object-oriented programming (বস্তুমুখী পরিগণনা)
- Error handling (ত্রুটি সামলানো)
- Logical operators (যুক্তি অণুক্রিয়া)
- Bit-wise operators (বিটপ্রতি অণুক্রিয়া)
- Data type (উপাত্ত প্রকরণ)
- Template (ছাঁচ)
- Compile-time (সংকলন সময়)

অধ্যায় ১৩

Input and Assignment (যোগান ও আরোপণ)

Programএ (ক্রমলেখ) data (উপাত্ত) কোথা থেকে আসে? হয় আমরা programএর ভিতরে সরাসরি লিখে দেই, যেমনটি আগের পাঠগুলোতে করেছি, আর না হয় আমরা data ব্যবহারকারীদের কাছে থেকে input (যোগান) নেই। Data input নিয়ে সেটিকে ধারন করার উদ্দেশ্যে আমরা variableএ (চলক) assign (আরোপণ) করি যাতে ওই data পরে কাজে লাগানো যায়।

১৩.১ Data Input (উপাত্ত যোগান)

সিপিপিতে এমন একটি program লিখো যেটি যে কোন আয়তের ক্ষেত্রফল ও পরিসীমা নির্ণয় করতে পারে। তোমার program তুমি মাত্র একবারই compile (সংকলন) করতে পারবে, আর প্রত্যেক আলাদা আয়তের জন্য তুমি programটি বারবার কেবল চালাতে পারবে, কিন্তু programএর ভিতরে দৈর্ঘ্য ও প্রস্থ বদলে দিয়ে বারবার compile করতে পারবে না। তারমানে তোমাকে দৈর্ঘ্য ও প্রস্থ input (যোগান) হিসাবে userএর কাছে থেকে নিতে হবে।

উক্ত program লেখার আগে চলো আমরা কিছু দরকারী আলোচনা সারি। আমরা যখন কোন computing problem (গণনা সমস্যা) সমাধান করতে চাই, যেমন আলোচ্য ক্ষেত্রে আমরা আয়তের দৈর্ঘ্য ও প্রস্থ জেনে তার ক্ষেত্রফল ও পরিসীমা নির্ণয় করতে চাই, তখন আমরা মুলত একটি program (ক্রমলেখ) ব্যবহার করবো, মানে আমরা programটি run করবো। এখন এই program হয়তো আমরা নিজেরা তৈরী করবো অথবা অন্য কেউ আমাদের তৈরী করে দিবে। বেশীর ভাগ ক্ষেত্রে programটি অন্যের তৈরী করা দেয়া, আমরা কেবল user।

ভেবে দেখো program তৈরী করা (write) আর run করা (চালানো) আসলে দুটো ভিন্ন ঘটনা। এই দুটো ঘটনা পরপর একসাথে ঘটবে এরকম সবসময় হয় না। বরং বেশীর ভাগ সময়ে এই ঘটনা দুটো আসলে ভিন্ন দুটি স্থানে ভিন্ন দুটি সময়ে ভিন্ন দুই ব্যক্তির দ্বারা সংঘটিত হয়। তাছাড়া program যে চালাবে সে হয়তো কেবল একটা আয়তের ক্ষেত্রফল ও পরিসীমা নির্ণয় করতে চায় না বরং তার হাতে হয়তো অনেক অনেক আয়ত আছে আর সে সবগুলো আয়তের জন্যই ক্ষেত্রফল ও পরিসীমা নির্ণয় করতে চায়। সুতরাং প্রতিটা আয়তের জন্য তার একটা করে আলাদা program লাগবে যদি programএর ভিতরে আয়তের দৈর্ঘ্য ও প্রস্থ দিয়ে দেয়া হয়। অথবা তার এমন একটা program লাগবে যেটা কোন না কোন ভাবে সবগুলো আয়তের জন্যই কাজ করবে, আর সঠিক ভাবেই করবে অর্থাৎ programটি মুলত formulaএর (সূত্র) ওপর নজর দেবে, dataএর (উপাত্ত) ওপর নয়। Data বদলালেও formula তো সবসময় একই থাকবে।

১৩.১. Data Input (উপাত্ত যোগান)

আমরা উপরে যেসব অবস্থা আলোচনা করলাম সেই সব অবস্থায় programmer (ক্রমলেখক) program (ক্রমলেখ) তৈরী করার সময় জানবেন না আয়তের দৈর্ঘ্য ও প্রস্থ কী হবে, সেটি জানা সম্ভব হবে পরে user যখন programটি চালাবেন কেবল তখন। প্রশ্ন হচ্ছে এমতাবস্থায় programmer data (উপাত্ত) ছাড়া কী ভাবে program তৈরী করবেন। সত্যি বলতে উত্তর তো গণিতেই আছে: variable (চলক) ব্যবহার করে। আর আমরা তো ইত্যমধ্যে programএ variable ব্যবহার করেছিই। আমাদের কেবল যেটা করা দরকার তা হলো programএর ভিতরে দৈর্ঘ্য বা প্রস্থ সরাসরি লিখে না দিয়ে ওইটা যাতে user program চালানোর সময় দিয়ে দিতে পারে সেই ব্যবস্থা করা। নীচের programএ আমরা তাই করেছি। আমরা userএর কাছে থেকে variableএর মান data হিসাবে input (যোগান) নিয়েছি। এবার আমরা ওই programটির সংশ্লিষ্ট অংশটুকু বিশ্লেষণ করি।

ফিরিস্তি ১৩.১: Programs with Data Input (উপাত্ত যোগানের ক্রমলেখ)

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  int length; // আয়তের দৈর্ঘ্যের জন্য চলক
  cin >> length;  // দৈর্ঘ্য যোগান হিসাবে নেওয়া হবে

  int width;  // আয়তের প্রস্থের জন্য চলক
  cin >> width; // প্রস্থ যোগান হিসাবে নেওয়া হবে

  // ক্ষেত্রফল ও পরিসীমার সূত্র
  int area = length * width;
  int perimeter = 2*(length + width);

  //  ক্ষেত্রফল ও পরিসীমা ফলন
  cout <<"area is "<< area << endl;
  cout <<"perimeter is "<< perimeter << endl;

  return EXIT_SUCCESS;  // সফল ভাবে ফেরত
}
```

যোগান-ফলন (input-output)

```
13
12
area is 156
perimeter is 50
```

উপরের programএ খেয়াল করো আমরা দৈর্ঘ্যের জন্য একটি variable length ঘোষনা করেছি, কিন্তু সাথে সাথে তার কোন initial value assign (আদিমান আরোপ) করি নাই। কারণ আগেই যেমন আলোচনা করলাম, আমরা যখন program লিখছি তখন আমরা আসলে জানিনা যে length এর মান কতো। আমরা বরং ওইটা userএর কাছে থেকে নেবো। আর সে কারণে আমরা cin >> length; লিখেছি। এখানে cin হল console in। সাধারণত input device (যোগান যন্ত্র) keyboard (চাপনি) ও mouse (টিপনি) আর output device (ফলন যন্ত্র) monitor (নজরি) মিলিয়ে হল আমাদের console বা যন্ত্রালয়। তো console in বলতে আমরা এখানে input device বিশেষ করে keyboard (চাপনি) থেকে input (যোগান) নেয়া বুঝাচ্ছি। তাহলে cin userএর কাছে থেকে keyboardএর মাধ্যমে সংখ্যাটা নিয়ে সেটা length variableএর ভিতরে দিয়ে দিবে। এতে ওই variableএ একরকমের value assign হয়ে যাবে।

Userএর কাছে থেকে দৈর্ঘ্য নেবার পরে আমাদের প্রস্থও নিতে হবে। উপরের programএ খেয়াল করো আমরা দৈর্ঘ্যের মতো করে প্রস্থের জন্যও width নামে একটা int ধরনের variable ঘোষনা করেছি আর তার পরের সারিতে cin ব্যবহার করে width এর মান userএর কাছে থেকে নেয়ার কথা লিখেছি। উপরের programএর বাঁকী অংশটুকু তো আগের পাঠের programগু-লোতে যেমন দৈর্ঘ্য ও প্রস্থ ব্যবহার করে ক্ষেত্রফল ও পরিসীমার সূত্র লিখা হয়েছে আর তারপরে output (ফলন) দেখানো হয়েছে ঠিক তেমনই। আমরা সেগুলো আর আলোচনা করছি না।

এবার আমরা আর একটু আলোচনা করি উপরের programটি compile (সংকলন) করে চালালে কী ঘটবে তা নিয়ে। উপরের programটি চালালে আমরা দেখব screenএ (পর্দা) কিছু আসছে না, cursorটা (চটুল) কেবল লাফালাফি করছে। আমরা এই অবস্থায় দৈর্ঘ্যের মান, ধরা যাক 13 চেপে enter (ভুক্তি) চাপবো। ভিতরে ভিতরে cin ওই মান নিয়ে length variableএর মধ্যে রেখে দিবে। Cursorটা (চটুল) তারপরও লাফালাফি করবে। আমরা তখন 12 দিয়ে enter (ভুক্তি) চাপবো, cin ওইটা width variableএ রেখে দিবে। তারপর screenএ আমরা output দেখতে পাবো। প্রথম সারিতে থাকবে area is 156 আর পরের সারিতে perimeter is 50।

উপরে programএ আমরা চাইলে কিছু সংক্ষিপ্তকরণ করতে পারি। যেমন দৈর্ঘ্য ও প্রস্থ এর variable দুটি declare ও input নেয়া চার সারিতে না করে আমরা ওগুলোকে মাত্র দুই সারিতে সারতে পারি। প্রথম সারিতে আমরা variable দুটো declare করবো। আর পরের সারিতে আমরা variable দুটোর input নিবো। নীচের programএ এইগুলো দেখানো হলো।

```
int length, width;      // আয়তের দৈর্ঘ্যে ও প্রস্থের জন্য চলক
cin >> length >> width; // দৈর্ঘ্য ও প্রস্থ যোগান নেওয়া হবে
```

আর সেক্ষেত্রে programটি চালানোর সময় input নেয়ার অংশ নিম্নরূপ হবে। লক্ষ্য করবে cursor (চটুল) যখন input নেবার জন্য লাফাতে থাকবে, আমরা তখন 13 ও 12 সংখ্যা দুটি ফাঁকা দিয়ে এক সাথে দিয়েই enter (ভুক্তি) চাপতে পারবো, অথবা চাইলে 13 লিখে enter চেপে তারপর 12 লিখে আবার enter চাপতে পারবো। আর output তো আগের মতোই হবে।

```
13 12
```

কেউ যদি চায় তাহলে কিন্তু output অংশেও এরকম সংক্ষিপ্তকরণ করতে পারে। যেমন ক্ষে-ত্রফল ও পরিসীমা চাইলে এক সারিতেই output দিতে পারে।

```
cout << "area and perimeter are " << area << " " <<
 perimeter << endl; // cout হতে এই পর্যন্ত পুরোটা আসলে এক
 সারিতে
```

তবে সবকিছু একবার cout দিয়ে দেওয়ার চেয়ে আমরা হয়তো দুইবারে দিতে চাইবো।

```
cout << "area and perimeter are ";
cout << area << " " << perimeter << endl;
```

উপরের উভয় ক্ষেত্রে screenএ output কিন্তু একসারিতেই আসবে।

```
area and perimeter are 156 50
```

১৩.২ Input Prompt (যোগান যাচনা)

সিপিপিতে এমন একটি program (ক্রমলেখ) রচনা করো যেটি যে কোন আয়তের ক্ষেত্রফল ও পরিসীমা নির্ণয় করতে পারে। তোমার program আয়তের দৈর্ঘ্য ও প্রস্থ ব্যবহারকারীর কাছে থেকে input (যোগান) নিবে। আর দৈর্ঘ্য ও প্রস্থ যোগান নেবার আগে তোমার program অবশ্যই ব্যবহারকারীকে দৈর্ঘ্য ও প্রস্থের মান জিজ্ঞেস করবে অর্থাৎ prompt (যাচনা) করবে।

ফিরিস্তি ১৩.২: Program with Input Prompt (যোগান যাচনার ক্রমলেখ)

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
  int length;        // আয়তের দৈর্ঘ্যের জন্য চলক
  cout << "length is? ";  // মান যাচনা করা হচ্ছে
  cin >> length;         // দৈর্ঘ্য যোগান হিসাবে নেওয়া হবে

  int width;         // আয়তের প্রস্থের জন্য চলক
  cout << "width is? "; // মান যাচনা করা হচ্ছে
  cin >> width;    // প্রস্থ যোগান হিসাবে নেওয়া হবে

  // ক্ষেত্রফল ও পরিসীমার সূত্র
  int area = length * width;
  int perimeter = 2*(length + width);

  //  ক্ষেত্রফল ও পরিসীমা ফলন দেয়া হবে
  cout <<"area is "<< perimeter << endl;
  cout <<"perimeter is "<< perimeter << endl;

  return EXIT_SUCCESS;  // সফল ভাবে ফেরত
}
```

যোগান-ফলন (input-output)

```
length is? 13
width is? 12
area is 156
perimeter is 50
```

আগের পাঠের programএ আমরা variableএর মান userএর কাছে থেকে নেয়ার জন্য cin ব্যবহার করেছি। ওই programটি যখন আমরা চালাই তখন দেখি screenএ (পর্দা) কিছু নাই আর cursorটা (চটুল) কেনো যেনো লাফালাফি করছে। সেই অবস্থায় আমরা প্রথমে দৈর্ঘ্যের মান 13 দিয়ে enter (ভুক্তি) চেপেছি। Cursorটা (চটুল) তারপরও লাফালাফি করছিল। আমরা তখন 12 দিয়ে enter চেপেছি। তারপর screenএ output এসেছিল প্রথম সারিতে area is 156 আর পরের সারিতে perimeter is 50। তো এই যে cursorটা (চটুল) লাফালাফি করছিল দৈর্ঘ্য ও প্রস্থের মান নেয়ার জন্য এইটা আমরা বুঝতে পারি কারণ আমরা নিজেরাই এক্ষেত্রে programটি তৈরী write) করেছি আর নিজেরাই সেটা compile (সংকলন) করে চালাচ্ছি (run)। আমরা এক্ষেত্রে জানি যে আমাদের programটি প্রথমে দৈর্ঘ্য চাচ্ছে আর সেটা দেবার পর প্রস্থ চাচ্ছে। এবার ভেবে দেখো আমাদের লেখা program যদি আমরা ছাড়া অন্য কেউ চালায় (run) তাহলে সে কী ভাবে জানবে cursorটি (চটুল) ওই অবস্থায় কেন লাফাচ্ছে। সে কি আসলেই দৈর্ঘ্য বা প্রস্থ নেয়ার জন্য অপেক্ষা করছে নাকি ভিতরে ভিতরে ঘটনা অন্য কিছু, সে হয়তো অন্য কিছু করছে।

তো ওপরের সমস্যা সমাধানের জন্য আমরা যেটি করবো সেটি হলো আমাদের programএ cin >> length; লেখার আগে আমরা একটা message দেখাবো যে আমরা দৈর্ঘ্যের মান চাই। উপরের program খেয়াল করো cin >> length; লেখার আগে আমরা cout << "length is? "; লিখে আসলে সেটাই করতে চাইছি। এই program যখন চালানো হবে তখন প্রথমে screenএ length is? দেখা যাবে। আর cout এর শেষে আমরা যেহেতু endl অর্থাৎ end line দেই নাই, cursorটা (চটুল) সেহেতু ওই একই সারিতে লাফাইতে থাকবে, লাফাইতে থাকবে মুলত cin >> length; এর কারণে length এর মান নেয়ার জন্য। আমরা তখন length এর মান দিয়ে enter (ভুক্তি) চাপবো। তাহলে "চটুল কেন লাফায়?" আমরা এই সমস্যার সমাধান করে ফেললাম কেমন! এই যে input (যোগান) নেবার আগে একটা message দিয়ে userকে জানানো যে আমরা কী input চাই, এই ব্যাপারটিকে বলা হয় input prompt (যোগান যাচনা)। উপরের programএ খেয়াল করো আমরা প্রস্থের জন্যেও একই ভাবে input (যোগান) নেবার আগে "width is? " message দিয়ে input prompt (যোগান যাচনা) করেছি। তাহলে এখন থেকে তোমার programএ input নেবার আগে অবশ্যই input prompt করবে, কেমন?

উপরে programএ আমরা চাইলে কিছু সংক্ষিপ্তকরণ করতে পারি। যেমন দৈর্ঘ্য ও প্রস্থ দুটি declare (ঘোষনা), input prompt (যোগান যাচনা) করা, ও input (যোগান) নেয়া ছয় সারিতে না করে আমরা ওগুলোক মাত্র তিন সারিতে সারতে পারি। প্রথম সারিতে আমরা variable দুটো declare করবো। আর পরের সারিতে আমরা input prompt করবো তারপরে সারিতে variable দুটোর মান input নিবো। নীচে programএ (ক্রমলেখ) এইগুলো দেখানো হলো।

```
int length, width;        // দৈর্ঘ্যে ও প্রস্থের জন্য চলক
cout << "length & width are? ";// একসাথে যাচনা
cin >> length >> width;      // দৈর্ঘ্য ও প্রস্থ যোগান
```

আর সেক্ষেত্রে programটি চালানোর সময় input নেয়ার অংশ নিম্নরূপ হবে। অর্থাৎ এই program চালালে length & width are? দেখানোর পরে cursorটা (চটুল) input নেবার জন্য লাফাতে থাকবে। আমরা 13 ও 12 সংখ্যা দুটি ফাঁকা দিয়ে এক সাথে দিয়েই enter (ভুক্তি)

চাপতে পারবো, অথবা চাইলে 13 লিখে enter চেপে তারপর 12 লিখে আবার enter চাপতে পারবো। আর outputএর অংশ আগের মতোই হবে, কাজেই আমরা সেটা আর দেখাচ্ছি না।

```
length & width are? 13 12
```

১৩.৩ Value Assignment (মান আরোপণ)

Programএ (ক্রমলেখ) variable নিয়ে তাতে value assign করলে আসলে কী ঘটে? আবার Variableএ একটা মান আগে থেকে আছেই, এমতাবস্থায় আরেকটা value assign করলে কী ঘটে? একটা variable থেকে আরেকটা variableএ value assign করলেই বা কী ঘটে?

```
int amar;
int tomar = 5;
```

উপরে আমরা দুটো variable declare (চলক ঘোষনা) করলাম: একটার নাম amar আর আরেকটার নাম tomar, দুটোই int ধরনের অর্থাৎ পূর্ণক, একটাতে initial value (আদিমান) দিয়ে দিলাম আর একটাতে দিলাম না। আমরা যখন variable declare করি তখন আসলে আমরা computerএর (গণনি) memoryতে (স্মরনি) কিছু জায়গা দখল করি। ধরে নিতে পারো memory হল একটা রাস্তার পাশে অনেকগুলো একই রকম বাড়ী। কোন variable declare করার সময় আমরা আসলে ওই বাড়ীগুলোর একটা দখল করে সেই বাড়ীটার নাম দিয়ে দেই আমাদের variableএর নামে। তোমরা নিশ্চয় দেখেছো অনেকেরই বাড়ীর নাম থাকে যেমন "শান্তি নীড়"। আমাদের varaible বাড়ীগুলোর নাম amar ও tomar। তো আমরা যখন উপরের দুটো variable declare করলাম তখন memoryতে ওই রকম দুটো জায়গা নিয়ে তাদের নাম দিয়ে দিলাম amar আর tomar। এখন কথা হচ্ছে memoryতে (স্মরণি) ওই জায়গায় আমরা আসলে রাখবো কী? উত্তরটাতো সহজ আমরা রাখবো variableটির মান। যখন আমরা initial value দিয়ে দিলাম তখন ওই জায়গাতে আমাদের দেয়া মানটা থাকবে, আর যখন initial value দিবো না, তখনও ওই জায়গাটিতে আগে থেকে যাই ছিল তাই থাকবে।

```
amar = tomar;
```

এবার আমরা যদি উপরের মতো করে tomar এর মান amar এ assign করি তাহলে কী ঘটবে? আসলে উপরের এই statement (বিবৃতি) চালানোর পরে amar এর আগের মান মুছে গিয়ে সেটার নতুন মান হয়ে যাবে tomar এর মানের সমান অর্থাৎ amar এর মানও হবে 5। এখানে একটা গুরুত্বপূর্ণ বিষয় বলে রাখতে হবে যে এই যে tomar থেকে amar এ মান assign করা হলো এতে কিন্তু tomar এর মানে কোন পরিবর্তন হবে না। অর্থাৎ tomar এর মান আগের মতো 5-ই থাকবে। Assignmentএ (আরোপণ) সমান চিহ্নের বামে যা থাকে সেটাকে target (লক্ষ্য) আর ডানে যেটা থাকে সেটাকে source (উৎস) বলা হয়, কারণ source থেকে মান নিয়ে targetএ assign করা হয়। উপরের assignmentএ amar = চিহ্নের বামে তাই এটি target আর tomar ডানপাশে তাই এটি source। Assignmentএর ফলে targetএর মান বদলে কিন্তু sourceএর মান বদলে না, একই থাকে।

উপরের program (ক্রমলেখ) আর output (ফলন) লক্ষ্য করো। আমরা প্রথমে variable x declare করে তার initial value (আদি মান) দিয়েছি 3, তারপর variable y declare করে তার initial value দিয়েছি x+5 অর্থাৎ 3 + 5 = 8। এই পর্যায়ে output দেখানো হয়েছে x আর y দুটোর মানেরই। Outputএ আমরা দেখতে পাচ্ছি x 3 y 8। তারপর programএ

আমরা লিখেছি x = y * 3; ফলে x এর মান হবে এখন 8 * 3 = 24, আর y এর মান কিন্তু একই থাকবে, কারণ y এর মান আমরা কেবল ব্যবহার করেছি, y এ তো কোন মান assign করি নাই। Programএ পরের statementএ (বিবৃতি) আমরা x ও y এর তখনকার মান দেখিয়েছি, Outputএ সেটা ঠিকই x 24 y 8 দেখাচ্ছে। Programএ এরপরের বাক্যে আমরা আবার x এ মান assign করেছি x = y + 3 * 4; তো এর ফলে আগের মতোই y এর মান বদল হবে না, কিন্তু x এর নতুন মান হয়ে যাবে 8 + 3 * 4 = 20, যা পরের cout এর মাধ্যমে outputএ ঠিকই দেখানো হয়েছে x 20 y 8। Programএ এরপর আমরা দেখি y = x * 2; এর ফলে y এর নতুন মান হবে y = 20 * 2 = 40, আর x এর মান এবার আগে যা ছিলো তাই থাকবে, কারণ x এর মান কেবল ব্যবহার করা হয়েছে, x এ কোন মান assign করা হয় নি।

```
int x = 3;       // initial value assign আদি মান আরোপ
int y = x + 5;   // initial value assign আদি মান আরোপ
cout << "x " << x << " y " << y << endl; // মান দেখাও

x = y * 3;       // assign value again পুনরায় মান আরোপ
cout << "x " << x << " y " << y << endl; // মান দেখাও

x = y + 3 * 4;   // assign value again পুনরায় মান আরোপ
cout << "x " << x << " y " << y << endl; // মান দেখাও

y = x * 2;       // assign value again পুনরায় মান আরোপ
cout << "x " << x << " y " << y << endl; // মান দেখাও
```

ফলন (output)

```
x 3 y 8
x 24 y 8
x 20 y 8
x 20 y 40
```

সবমিলিয়ে একটা বিষয় দেখা যাচ্ছে assignmentএ (আরোপণ) = চিহ্নের বামের target variableএর (লক্ষ্য চলক) মান কেবল পরিবর্তন হয়, আর = চিহ্নের ডানে থাকা variable (চলক) বা expressionএর (রাশি) এর মান কোন পরিবর্তন হয় না। আরেকটি ব্যাপার হলো কোন variableএ পরে কোন নতুন মান assign না হওয়া পর্যন্ত আগেরবার যে মান assign করা হয়েছিল সেটাই থাকে।

১৩.৪ Value Swapping (মান অদল-বদল)

ধরো তোমার দুটো variable (চলক) আছে x আর y আর তাদের মান যথাক্রমে 12 ও 13। তো তোমাকে এমন কিছু statement লিখতে হবে যাতে ওই statementগুলো চালানোর (run) পরে আমরা x আর y এর মান যথাক্রমে 13 আর 12 পাই অর্থাৎ মানদুটো অদল-বদল হয়ে যায়।

```
int x = 12; // x এর মান assign করা হলো
int y = 13; // y এর মান assign করা হলো
```

১৩.৪. Value Swapping (মান অদল-বদল)

উপরে আমরা কেবল variable x আর y declare করে তাদের initial value হিসাবে 12 ও 13 দিয়ে দিলাম। এখন আমরা এমন কিছু করবো যাতে x আর y মান swap হয়ে যায়। প্রথমেই আমরা একটা চটুল সমাধান করি। তোমাদের মধ্যে যারা দুষ্টু ধরনের আর চটপটে তারা সাধারণত এই সমাধানটি করতে চাইবে। নীচের বিবৃতি দুটো লক্ষ্য করো: আমরা স্রেফ x এর মধ্যে সরাসরি 13 assign করেছি আর y এর মধ্যে 12 assign করেছি। ব্যস হয়ে গেল x আর y এর মান অদল-বদল! আসলে আমরা কী এইটে চেয়েছিলাম? এখানে তো variable দুটোর মধ্যে একটা থেকে আরেকটাতে মান নেয়ার মতো কোন ঘটনা ঘটে নি, কাজেই কোন অদল বদলের কিছু ঘটে নি!

```
x = 13; // x এর মান assign করা হলো
y = 12; // y এর মান assign করা হলো
```

অদল-বদল বুঝার জন্য চিন্তা করো তোমার দুটি পেয়ালা আছে: কাঁচের পেয়ালা আর কাঁসার পেয়ালা। কাঁচের পেয়ালায় আছে আঙুরের রস আর কাঁসার পেয়ালায় কমলার রস। এখন তুমি এই পেয়ালা দুটোতে থাকা ফলের রস অদল-বদল করতে চাও যাতে কাঁচের পেয়ালায় থাকে কমলার রস আর কাঁসার পেয়ালায় থাকে আঙুরের রস। তো এখন তুমি কী করবে। তুমি তো আর সরাসরি এক-টার ফলের রস আরেকটাতে ঢেলে দিতে পারো না। তোমাকে যেটা করতে হবে তা হলো আরেকটা পেয়ালা নেয়া। ধরো সেটা কাঠের পেয়ালা। এই কাঠের পেয়ালাটি তুমি একটা থেকে আরেকটাতে ঢালাঢালির কাজে ব্যবহার করবে। তাহলে এই অতিরিক্ত কাঠের পেয়ালা কাজে লাগিয়ে কীভাবে তোমার কাঁচ আর কাঁসার পেয়ালার ফলের রস অদল-বদল করা যায়, আমরা নীচে তা দেখি।

১. একদম শুরুতে কাঁচের পেয়ালায় রয়েছে আমাদের আঙুরের রস আর কাঠের পেয়ালা খালি। সুতরাং কাঁচের পেয়ালা থেকে আঙুরের রস কাঠের পেয়ালায় ঢালো। ফলে কাঠের পেয়ালায় থাকলো আঙুরের রস আর কাঁচের পেয়ালা খালি হয়ে গেলো।

২. কাঁচের পেয়ালা যেহেতু এখন খালি আর কাঁসার পেয়ালায় আছে কমলার রস, আমরা তাই কাঁসার পেয়ালার কমলার রস কাঁচের পেয়ালায় ঢালবো। ফলে কাঁচের পেয়ালায় থাকলো কমলার রস আর কাঁসার পেয়ালা খালি হয়ে গেলো।

৩. কাঁসার পেয়ালা যেহেতু এখন খালি আর কাঠের পেয়লায় আছে আঙুরের রস, আমরা তাই কাঠের পেয়ালার আঙুরের রস কাঁসার পেয়ালায় ঢালবো। ফলে কাঁসার পেয়ালায় থাকলো আঙুরের রস আর কাঠের পেয়ালা খালি হয়ে গেলো।

উপরের ধাপ তিনটি সম্পন্ন করলেই আমাদের এক পেয়ালার ফলের রস আরেক পেয়ালায় অদল-বদল হয়ে যাবে। তো পেয়ালা দুটোর রস অদল-বদলের মতোই আসলে আমাদের variable দুটোর মান অদল-বদল করতে হবে। একটা অতিরিক্ত পেয়ালার মতো আমাদের এখানেও লাগবে একটা অতিরিক্ত variable (চলক)। ধরে নেই আমাদের সেই অতিরিক্ত variable হলো z। আম-রা তাহলে এই অতিরিক্ত variable কাজে লাগিয়ে আমাদের x আর y variable দুটোর value swap (মান অদল-বদল) করে ফেলি।

```
z = x;  // z হলো 12 আর x আছে 12, y আছে 13
x = y;  // x হলো 13 আর y আছে 13, z আছে 12
y = z;  // y হলো 12 আর z আছে 12, x আছে 13
```

তো উপরের তিনটি statement চালালেই আমাদের x আর y variable দুটোর মান অদল-বদল হয়ে গেলো। তবে পেয়ালা আর ফলের রসের অদল বদলের সাথে variable আর valueএর অদল-বদলের কিন্তু কিছুটা তফাৎ আছে। তফাৎটা হলো ফলের রস এক পেয়ালা থেকে আরেক

পেয়ালয় ঢাললে যেটা থেকে ঢালা হলো সেই পেয়ালা খালি হয়ে যায়। কিন্তু variableএর ক্ষেত্রে z = x; করলে variable xএর মান variable zএ assign হয় ঠিকই, কিন্তু variable x কিছুতেই খালি হয় না, বরং তার যে মান ছিলো সেটাই থাকে। Variableএর মান বদলে যায় কেবল যখন এতে নতুন মান assign করা হয়।

১৩.৫ Assignment Left and Right (আরোপণের বাম ও ডান)

কোন variableএর (চলক) l-value (বাম-মান) ও r-value (ডান-মান) বলতে কী বুঝো? কোন variableএ মান assign করতে গেলে আমরা assignment (আরোপণ) = চিহ্ন দিয়ে বামে ও ডানে কিছু লিখি যেমন y = x;। এখানে বামেরটিকে বলা হয় target (লক্ষ্য) আর ডানেরটিকে বলা হয় source (উৎস)। Assignmentএর (আরোপণ) ফলে ডান পাশের source থেকে মান বাম পাশের targetএ assign হয়। কথা হচ্ছে assignment = চিহ্নের বামে আমরা কী কী দিতে পারবো বা পারবো না, আর ডানেই বা কী কী দিতে পারবো বা পারবো না? তাছাড়া একটা variableএর নাম assignment = চিহ্নের বাম বা ডানপাশে লিখলে এই দুই ক্ষেত্রে variableএর ভুমিকায় আসলে কোন তফাৎ হয় কিনা?

এই আলোচনায় যাওয়ার আগে আমরা একটু পরের উদ্ধৃতাংশটুকু বিবেচনা করি। "ঢাকার মামা হালিম বিখ্যাত। চল আমরা মামা হালিম খাই। তুমি খাবে এক বাটি, আমি খাব এক বাটি। আমার বাটিটা পরিস্কার নয়, তোমার বাটিটা পরিস্কার।" তো এইখানে বাটি মানে কখন আসলে হালিম আর কখন আসলে সেটা পাত্র? আমরা বুঝতে পারি "তুমি খাবে এক বাটি, আমি খাব এক বাটি" এই কথাগুলোতে বাটি বলতে আসলে সত্যি সত্যি পাত্রটাকে কামড়ে কামড়ে খাওয়ার কথা বলা হচ্ছে না, বরং তুমি এক বাটি পরিমান হালিম খাবে আর আমি এক বাটি পরিমান হালিম খাবো তাই বুঝানো হচ্ছে। এক বাটি হালিম মানে একটা বাটিতে থাকা হালিম। বিষয়গুলোকে চলক আর তার মানের সাথে মিলাও। বাটি ঠিক যেন চলকের মতো আর হালিম হল তার মানের মতো। আবার "আমার বাটিটা পরিস্কার নয়, তোমার বাটিটা পরিস্কার।" এই অংশে বাটি মানে আসলে বাটি নামের পাত্রটা, সেই পাত্রে ঢালা হালিম নয় কোন ভাবেই। তাহলে দেখা যাচ্ছে বাটি বলতে কখনো কখনো আসলে পাত্রটাকে বুঝানো হয় আর কখনো কখনো পাত্রটাতে থাকা হালিমকে বুঝানো হয়। একই ভাবে variableএর নাম উল্লেখ করলে কখনো কখনো variableটির মানকে বুঝানো হয়, কখনো কখনো আসলে varaibleটির জন্য memoryতে (স্মরণি) বরাদ্দ জায়গাটুকু বুঝানো হয়।

x = 3; এখানে variable x বলতে আমরা আসলে variable x এর জন্য memoryতে (স্মরনি) নেয়া জায়গাটুকু বুঝি যেখানে মান 3 কে রাখা হবে। এখানে কোন ভাবেই variable x এ আগে থেকে বিদ্যমান মানকে বুঝানো হচ্ছে না। খেয়াল করো এখানে variable x assign = চিহ্নের বাম পাশে আছে। যখন variable x আসলে memoryতে বরাদ্দকৃত জায়গাকে বুঝায় তখন এটাকে আমরা স্রেফ variable না বলে আরো স্পষ্ট করে বলবো varaibleএর l-value (বাম-মান)। তাহলে মনে রেখো variableএর l-value দিয়ে আমরা বুঝাবো variableএর জন্য memoryতে নেয়া জায়গাটুকু।

y = x; এখানে variable y বলতে আমরা variable y এর জন্য memoryতে বরাদ্দ পাওয়া জায়গাটুকুকে বুঝি। আর variable y assign = চিহ্নের বামে আছে তাই এখানে variable y এর বাম-মান ব্যবহৃত হয়েছে। তবে variable x বলতে এখানে আমরা কেবল তার মানটাকে বুঝি। খেয়াল করো variable x এর মানটাইতো variable y এর memoryএর জায়গাটাতে জমা হবে, variable x এর জন্য বরাদ্দ জায়গাতো আর গিয়ে variable y

এর জায়গায় লেখা হবে না। আমরা দেখছি এখানে variable x আরোপ = চিহ্নের ডানে রয়েছে। যখন variable x আসলে তার মানটাকে বুঝায় তখন আমরা এটাকে বলব variable r-value (ডান-মান)। Variableএর r-value দিয়ে আমরা তাহলে বুঝাবো variableএর যে মান সেটিকে, memoryতে থাকা জায়গাটিকে নয়।

উপরের আলোচনা থেকে আমরা একটা বিষয়ই পরিস্কার করতে চেয়েছি সেটা হলো, আরোপ = চিহ্নের বামে আমরা কেবল এমন কিছু দিতে পারবো যার জন্য memoryতে জায়গা দখল করা আছে, অর্থাৎ যার l-value (বাম-মান) আছে। আর assign চিহ্নের ডান পাশে আমরা এমন কিছু দিতে পারবো যার মান আছে অর্থাৎ r-value (ডান-মান) আছে। একটা বিষয় খেয়াল করো যার l-value আছে অর্থাৎ memoryতে যার জায়গা আছে তার একটা মানও থাকবেই অর্থাৎ তার r-value থাকবেই, যেমন যে কোন variableএর। কথা হচ্ছে এমন কিছু কি আছে যার r-value আছে কিন্তু l-value নাই। উত্তর ধরে নিতে পারো আছে। যেমন x = 3; এইখানে 3 এর r-value আছে কিন্তু l-value নাই। কাজেই কেউ চাইলে 3 = x; লিখতে পারবে না, compile (সংকলন) করার সময় error দেখাবে, বলবে "error: lvalue required as left operand of assignment"। একই ভাবে কেউ চাইলে assignment হিসাবে y+3 = x;ও লিখতে পারবে না, একই error (ত্রুটি) দেখাবে, কারণ variable y এর l-value সম্ভব হলেও y + 3 করলে ওইটা আর variable y থাকে না হয়ে যায় একটা expressioni (রাশি) যার মান হবে y এর মান যোগ 3, কাজেই সেটার কেবল মান থাকে, তার জন্য memoryতে কোন জায়গা থাকে না। বুঝাই যাচ্ছে অন্যদিকে assignment হিসাবে x = y + 3; লিখা যাবে কারণ y + 3 এর r-value আছে অপর দিকে variable x এর l-value আছে।

১৩.৬ Self-Referential Assignment (আত্ম-শরন আরোপণ)

Program (ক্রমলেখ) দেখলে আমাদের সাধারণত x = x + 1; বা এই জাতীয় অদ্ভুত কিছু বিষয় নজরে আসে। মুল কথা হলো এই সব ক্ষেত্রে একই variable (চলক) assignment (আরোপ) = চিহ্নের বামেও রয়েছে আবার ডানেও রয়েছে। আমরা সকলে গণিত জানি কম বা বেশী। সেখানে সমীকরণ নিয়ে আমাদের যে ধারনা আছে সেই অনুযায়ী তো x কখনো x + 1 এর সমান হতে পারে না। তাহলে programএ x = x + 1; এর মতো অর্থহীন বিষয় কেন থাকে?

```
x = x + 1;  // চিহ্ন = গণিতের সমান চিহ্ন নয়, এটি গণনার আরোপণ।
```

আসলে = চিহ্নটি গণিতে আমরা ব্যবহার করি দুটো সংখ্যা তুলনা করে যদি দেখি তারা একে অপরের সমান তাহলে। আমরা তাই ওটাকে গণিতে equal (সমান) চিহ্ন বলে থাকি। কিন্তু গণনার জগতে = চিহ্নটিকে সমান চিহ্ন হিসাবে ব্যবহার না করে বরং assignment (আরোপণ) চিহ্ন হিসাবে ব্যবহার করা হয়। কাজেই কোন programএ আমরা যখন x = x + 1; দেখি তখন আসলে ওটা কোন ভাবেই গণিতের সমীকরণ নয়, বরং ওইটা গণনার জগতের assignment। সুতরাং গণিতের জগতে ওইটা কোন অর্থ তৈরী না করলেও গণনার জগতে ওটার সুনির্দিষ্ট অর্থ আছে।

আমরা assignment (আরোপণ) নিয়ে আগেই আলোচনা করেছি। ওই assignmentগুলোর সবগুলোতে বাম ও ডান উভয় পাশে variable থাকলেও আলাদা আলাদা variable ছিল। আর x = x + 1ও assignment তবে এখানে একই variable assign চিহ্নের বামেও আছে ডানেও আছে। এইরকম assignment যেখানে একই variable বামেও আছে ডানেও আছে সেটাকে আমরা বলবো self-referential assignment (আত্মশরন আরোপণ) অর্থাৎ যেখানে একটা variable নিজের মানের জন্য নিজেরই শরনাপন্ন হয়। Self-referential (আত্মশরণ) assignmentএ ডানপাশে variableটির r-value (ডান-মান) ব্যবহৃত হয়, আর বাম-

পাশে variableটির l-value (বাম-মান) ব্যবহৃত হয়। এই রকম assignmentএ আসলে কী ঘটে?

```
int x = 3;          // চলক x এ আদি মান আরোপ করা হলো
x = x + 1;          // এখানে আত্ম–শরণ আরোপণ করা হচ্ছে
cout << x << endl;  // চলক x এর মান ফলন দেওয়া হচ্ছে
```

এই রকম assignment বুঝতে গেলে আমরা x = x + 1; statementটিকে দুইটি ঘটনায় বিভক্ত করে নিতে পারি। একটা ঘটনা হল ডান পাশে x + 1 হিসাব করা অর্থাৎ x+1 এর মান বা আরো পরিস্কার করে বললে r-value হিসাব করা। আর অন্য ঘটনাটা হল বাম পাশে x এর l-value অর্থাৎ memoryতে (স্মরনি) x এর জন্য বরাদ্দ করা জায়গায় ডান পাশ থেকে পাওয়া মানটি লিখে দেওয়া। তো এই দুটো ঘটনার প্রথমটি আগে ঘটবে আর দ্বিতীয়টি পরে ঘটবে। উপরে আমরা x এর initial value নিয়েছি 3। এরপর যখন x= x+1; execute (নির্বাহ) হবে তখন প্রথম ঘটনাটি ঘটবে আগে অর্থাৎ x + 1 মান হিসাব হবে। x এর মান যেহেতু এই অবস্থায় 3 তাই x + 1 হবে 4। মনে করে দেখো এই 4 এর কিন্তু কেবল r-value আছে এর জন্য memoryতে কোন জায়গা দখল করা নেই বা এর কোন l-value নেই। অর্থাৎ এই 4 কোন ভাবেই x variableএর জায়গায় নেই, অন্য কোথাও আছে। যাইহোক এমতাবস্থায় এরপর ঘটবে দ্বিতীয় ঘটনাটি অর্থাৎ এই 4 মানটি গিয়ে লেখা হয়ে যাবে x এর জন্য বরাদ্দ জায়গাতে। আমরা তাই x এর পুরনো মান 3 বদলে সেখানে পাবো এর নতুন মান 4। তাহলে x = x + 1; self-referential assignmentএর (আত্ম-শরণ আরোপণ) ফলে variableএর মান এক বেড়ে গেলো।

Self-referential assignmentএর আরো নানান জটিল অবস্থা আছে যেমন x = x ∗ 3; বা x = x∗x + x + 1;। এগুলোর প্রতিটি ক্ষেত্রে আগে ডানপাশের মান হিসাব করা হবে আর তারপর সেই মান বাম পাশে লিখে দেয়া হবে, ফলে variableটিতে নতুন একট মান থাকবে।

১৩.৭ Exercise Problems (অনুশীলনী সমস্যা)

Conceptual Questions: নীচে কিছু conceptual প্রশ্ন রয়েছে। প্রশ্নগুলোর উত্তর নিজে নিজে বের করবে।

১. সরাসরি programএর (ক্রমলেখ) ভিতরে data দিয়ে দিলে সমস্যা কী?

২. Data (উপাত্ত) কেনো input (যোগান) নিতে হবে? সুবিধা-অসুবিধা কী কী?

৩. Input prompt (যোগান যাচনা) কী? Input নেয়ার আগে কেন prompt করা উচিৎ?

৪. Variableএ (চলক) মান assignmentএ source ও targetএ কী ঘটে বর্ণনা করো।

৫. Variableএর l-value আর r-value বলতে কী বুঝো? উদাহরণ দিয়ে ব্যাখ্যা করো।

৬. Assignmentএ = চিহ্নের বামে কেন এমন কিছু দেয়া যায় না যার কেবল r-value আছে?

৭. Self-referential (আত্ম-শরণ) assignment কী উদাহরণ সহ ব্যাখ্যা করো।

৮. দুটি variableএ (চলক) থাকা মান বদলাবদলি করবে কেমনে ব্যাখ্যা করো।

১৩.৭. Exercise Problems (অনুশীলনী সমস্যা)

Programming Problems: নীচে আমরা কিছু programming সমস্যা দেখবো। এই সমস্যাগুলো আগে ধৈর্য্য ধরে নিজে নিজে সমাধান করতে চেষ্টা করবে। যখন একেবারেই পারছো না বলে মনে হয় তখনই কেবল সমাধান দেখে নিতে পারো। সমাধানগুলো programming প্রশ্নগুলোর শেষে আছে।

১. এমন একটি program (ক্রমলেখ) রচনা করো যেটি একটি পূর্ণক (int) ও একটি ভগ্নক (float) input (যোগান) নিয়ে সেগুলো আবার outputএ (ফলন) দেখায়।

২. এমন একটি program (ক্রমলেখ) রচনা করো যেটি দুটি ভগ্নক (float) সংখ্যা input (যোগান) নিয়ে সংখ্যা দুটি ও তাদের যোগফল outputএ ফলন দেখায়।

৩. এমন একটি program (ক্রমলেখ) রচনা করো যেটি তিনটি পূর্ণক (int) input (যোগান) নিয়ে তাদেরকে যে ক্রমে input নেয়া হয়েছে সেই ক্রমে আবার উল্টোক্রমে দেখাবে। যেমন ভুক্ত সংখ্যা তিনটি যদি হয় পর পর 2 3 1 তাহল সিধা ক্রমে দেখাবে 2 3 1 আবার তাদের উল্টাক্রমে দেখাবে 1 3 2। খেয়াল করো আমরা কিন্তু মানের ক্রম বলছি না।

৪. এমন একটি program (ক্রমলেখ) রচনা করো যেটি একদম ঠিক ঠিক নীচের মতো input (যোগান) ও output (ফলন) উৎপন্ন করে। তুমি কিন্তু পরীক্ষার নম্বরগুলো input নিবে, আর আমরা একেকবার চালানোর সময় এক এক রকম সংখ্যা input দিবো।

```
program to calculate result
- - - - - - - - - - - - - - -
number in first exam? 90
number in second exam? 75
number in third exam? 91
- - - - - - - - - - - - - - -
total number obtained 256
```

Programming Solutions: এবার আমরা programming সমাধানগুলো দেখবো। মনে রাখবে সমাধানগুলো দেয়া হয়েছে তুমি যদি একান্তই নিজে নিজে করতে পারছোনা বলে মনে হয়, তখন কেবল একটু সাহায্য যাতে পেতে পারো তাই। কাজেই চেষ্টা করবে সমাধান না দেখতে।

> ধরে নিই তুমি দরকারী header file (শির নথি) include করা, namespace (নামাধার) std ব্যবহার করা, main functionএর skeleton লেখা আর সেটার শেষে return EXIT_SUCCESS; লিখে মান ফেরত দেয়া ইত্যমধ্যে ভালো করে শিখে ফেলেছো। তো তুমি যদি নীচে লেখা programগুলো compile (সংকলন) করে চালাতে (run) চাও, তোমাকে কিন্তু আগে include, namespace, main, return এগুলো লিখে নিতে হবে, তারপর main functionএর ভিতরে return এর আগে তুমি আমাদের নীচের অংশগুলো লিখে নিবে। তারপর compile করে program চালাবে। আমরা এখন থেকে মোটামুটি এইভাবে program দেখাবো।

১. এমন একটি program (ক্রমলেখ) রচনা করো যেটি একটি পূর্ণক (int) ও একটি ভগ্নক (float) input (যোগান) নিয়ে সেগুলো আবার outputএ (ফলন) দেখায়।

ফিরিস্তি ১৩.৩: যোগান ও ফলনের ক্রমলেখ (Input Output Program)

```
int integer;
float fraction;

cout << "integer is? ";
cin >> integer;

cout << "fraction is? ";
cin >> fraction;

cout << "puronok is " << integer << endl;
cout << "fraction is " << fraction << endl;
```

২. এমন একটি program (ক্রমলেখ) রচনা করো যেটি দুটি ভগ্নক (float) সংখ্যা input (যোগান) নিয়ে সংখ্যা দুটি ও তাদের যোগফল outputএ ফলন দেখায়।

ফিরিস্তি ১৩.৪: Input Process Output (যোগান প্রকিয়ন ফলন)

```
float first, second;

cout << "two numbers are? ";
cin >> first >> second;

float sum = first + second;

cout << "two numbers are "; //  কোন endl নাই
cout << first << " " << second << endl;

cout << "their sum is " << sum << endl;
```

৩. এমন একটি program (ক্রমলেখ) রচনা করো যেটি তিনটি পূর্ণক (int) input (যোগান) নিয়ে তাদেরকে যে ক্রমে input নেয়া হয়েছে সেই ক্রমে আবার উল্টোক্রমে দেখাবে। যেমন ভুক্ত সংখ্যা তিনটি যদি হয় পর পর 2 3 1 তাহল সিধা ক্রমে দেখাবে 2 3 1 আবার তাদের উল্টাক্রমে দেখাবে 1 3 2। খেয়াল করো আমরা কিন্তু মানের ক্রম বলছি না।

ফিরিস্তি ১৩.৫: Input Order Reverse Order (যোগানের সিধা ক্রম উল্টা ক্রম)

```
int first, second, third;

cout << "three numbers are? ";
cin >> first >> second >> third;

cout << "given order " << first << " ";
cout << second << " " << third << endl;
```

```
cout << "reverse order " << third << " ";
cout << second << " " << first << endl;
```

৪. এমন একটি program (ক্রমলেখ) রচনা করো যেটি একদম ঠিক ঠিক নীচের মতো input (যোগান) ও output (ফলন) উৎপন্ন করে। তুমি কিন্তু পরীক্ষার নম্বরগুলো input নিবে, আর আমরা একেকবার চালানোর সময় এক এক রকম সংখ্যা input দিবো।

```
program to calculate result
- - - - - - - - - - - - - - -
number in first exam? 90
number in second exam? 75
number in third exam? 91
- - - - - - - - - - - - - - -
total number obtained 256
```

ফলাফল processingএর programটি আমরা নীচে দেখাচ্ছি।

ফিরিস্তি ১৩.৬: Result Processing Program (ফলাফল প্রক্রিয়ার ক্রমলেখ)

```
int first, second, third;

cout << "result processing program" << endl;

cout << "- - - - - - - - - - - - - - -" << endl;

cout << "number in first exam? ";
cin >> first;

cout << "number in second exam? ";
cin >> second;

cout << "number in third exam? ";
cin >> third;

cout << "- - - - - - - - - - - - - - -" << endl;

int sum = first + second + third;

cout << "total number is ";
cout << sum << endl;
```

১৩.৮ Computing Terminologies (গণনা পরিভাষা)

- Formula (সূত্র)
- Prompt (যাচনা)
- Swap (অদল-বদল)
- l-value (বাম-মান)
- r-value (ডান-মান)
- Self-reference (আত্ম-শরণ)

অধ্যায় ১৪

Mathematical Processing (গাণিতিক প্রক্রিয়াকরণ)

Mathematical processingএ expressionএ (রাশি) বিভিন্ন operators (অণুক্রিয়া) ও functions (বিপাতক) কী ভাবে হিসাব করা হয় আমাদের তা জানতে হবে।

১৪.১ Unary Operators (একিক অণুক্রিয়া)

সিপিপিতে unary (একিক) operator positive (ধনাত্মক) + আর negative (ঋণাত্মক) − কী ভাবে কাজ করে? যথাযথ program লিখে উদাহরণ সহ বুঝিয়ে দাও। Unary (একিক) operator (অণুক্রিয়ক) একটা operandএর (উপাদান) ওপর প্রযুক্ত হয়ে ফলাফল তৈরী করে।

ফিরিস্তি ১৪.১: Arithmetic Positive Negative (পাটিগণিতের ধনাত্মক ও ঋণাত্মক)

```
int a = 5;     int const b = -9; // a চলক b ধ্রুবক
cout << "+7 = " << +7 << "    -7 = " << -7 << endl;
cout << "+a = " << +a << "    -a = " << -a << endl;
cout << "+b = " << +b << "    -b = " << -b << endl;
cout << endl;
cout << "+(a*b) = " << +(a*b);     //  a*b হল রাশি
cout << "    -(a*b) = " << (a*b) << endl;
cout << "+abs(b) = " << +abs(b);  //abs() বিপাতক
cout << "    -abs(b) = " << -abs(b) << endl;
```

ফলন (output)

```
+7 = 7    -7 = -7
+a = 5    -a = -5
+b = -9    -b = 9

+(a*b) = -45    -(a*b) = -45
+abs(b) = 9    -abs(b) = -9
```

কোন number (সংখ্যা), variable (চলক), constant (ধ্রুবক), function (বিপাতক), বা expressionএর (রাশি) সামনে positive চিহ্ন থাকলে তার যে মান সেটিই থাকে, কিন্তু negative চিহ্ন থাকলে তার চিহ্ন বদলে যায় অর্থাৎ আগে positive থাকলে পরে negative হয়ে যায় আর আগে negative থাকলে পরে positive হয়ে যায়। Variable ও constant আগেই জানো। Function (বিপাতক) হলো এমন একটা জিনিস যে কিছু input (যোগান) নিয়ে কিছু output (ফলন) দেয়। যেমন cstdlib নামক header fileএ (শির নথি) abs(x) নামে একটা function আছে যেটি একটি সংখ্যা input নিয়ে তার চিহ্নটুকু বাদ দিয়ে কেবল মানটুকু output হিসাবে return করে। অর্থাৎ abs(3) হলো 3 আবার abs(-3)ও 3। একই ভাবে abs(3.5) হলো 3.5 আবার abs(-3.5)ও 3.5। expression (রাশি) হল সংখ্যা, constant, variable, operator, function মিলে যখন একটা জিনিস তৈরী হয় যার মান হিসাব করা যায় যেমন 3 + x $*$ abs(y) একটি রাশি যেখানে x আর y হল চলক।

১৪.২ Binary Operators (দুয়িক অণুক্রিয়া)

সিপিপিতে binary (দুয়িক) operatorগুলো যোগ +, বিয়োগ $-$, গুণ $*$, কী ভাবে কাজ করে? যথাযথ program লিখে উদাহরণ সহ বুঝিয়ে দাও। binary operator (দুয়িক অণুক্রিয়ক) দুটো operandএর (উপাদান) ওপর প্রযুক্ত হয়ে ফলাফল উৎপন্ন করে।

ফিরিস্তি ১৪.২: Arithmetic Plus Minus Times (পাটিগণিতের যোগ বিয়োগ গুণ)

```
cout << "5 + 3 = " << 5 + 3 << endl;
cout << "5.1 + 3 = " << 5.1 + 3 << endl;
cout << "5.1 + 3.2 = " << 5.1 + 3.2 << endl;
cout << endl;

cout << "5 - 3 = " << 5 - 3 << endl;
cout << "5.1 - 3 = " << 5.1 - 3 << endl;
cout << "5.1 - 3.2 = " << 5.1 - 3.2 << endl;
cout << endl;

cout << "5 * 3 = " << 5 * 3 << endl;
cout << "5.1 * 3 = " << 5.1 * 3 << endl;
cout << "5.1 * 3.2 = " << 5.1 * 3.2 << endl;
cout << endl;
```

উপরের program (ক্রমলেখ) খেয়াল করো। আর তার সাথে নীচের output (ফলন) মিলিয়ে নাও। লক্ষ্য করো আমরা তিনটি করে যোগ, বিয়োগ, আর গুণ করেছি। যোগ, বিয়োগ, বা গুণ আমরা ভালোই জানি, নতুন করে শেখার কিছু নাই। তবে একটি বিষয় খেয়াল করতে হবে। সেটি হলো data type কেমন? আর এ কারণেই আমরা প্রতিটি operatorএর (অণুক্রিয়া) জন্যে তিনটি করে উদাহরণ নিয়েছি। প্রতিটি operatorএর উদাহরণগুলোর প্রথম সারিতে খেয়াল করো। সেখানে operand (উপাদান) হিসাবে আমরা দুটো পূর্ণকের যোগ, বিয়োগ বা গুণ করেছি, যেমন 5 + 3, 5 $-$ 3 আর 5 $*$ 3। ফলাফল হিসাবে যা পেয়েছি তাও একটি পূর্ণক, যেমন 8, 2, আর 15। এবার প্রতিটি operatorএর জন্য তৃতীয় সারিতে খেয়াল করো। সেখানে operand (উপাদান) হিসাবে আমরা দুটো ভগ্নক যোগ, বিয়োগ বা গুণ করেছি, যেমন 5.1 + 3.2, 5.1 $-$ 3.2 আর

5.1 * 3.2। ফলাফল হিসাবেও আমরা পেয়েছি একটি ভগ্নক যেমন 8.3, 1.9, আর 16.32। তারপর প্রতিটি operatorএর জন্য দ্বিতীয় সারিতে খেয়াল করো। Operand হিসাবে একটি ভগ্নক ও একটি পূর্ণক যোগ, বিয়োগ বা গুণ করা হয়েছে যেমন 5.1 + 3, 5.1 − 3, আর 5.1 * 3। আর ফলাফল এসেছে একটি ভগ্নক যেমন 8.1, 2.1, আর 15.3, যেগুলোর কোনটিই পূর্ণক নয়। Operandদুটো একটা ভগ্নক হওয়ায় ফলাফলও ভগ্নক হয়ে গেছে।

ফলন (output)

```
5 + 3 = 8
5.1 + 3 = 8.1
5.1 + 3.2 = 8.3

5 − 3 = 2
5.1 − 3 = 2.1
5.1 − 3.2 = 1.9

5 * 3 = 15
5.1 * 3 = 15.3
5.1 * 3.2 = 16.32
```

তাহলে উপরের আলোচনা থেকে আমরা দেখলাম কোন operatorএর (অণুক্রিয়া) যদি দুটি operandই (উপাদান) একরকম হয় তাহলে ফলাফলও সেই রকমই হয়। যেমন operand দুটোই int হলে ফলাফলও int; operand দুটোই float হলে ফলাফলও float। আর যদি দুটো operand দুরকম হয় যেমন একটি পূর্ণক বা int আর একটি ভগ্নক বা float তাহলে ফলাফল হবে ভগ্নক বা float। গণিতে আমরা জানি পূর্ণক সংখ্যাগুলো একই সাথে ভগ্নকও যেমন 3 আসলে 3.0, কিন্তু একটি ভগ্নক কিন্তু পূর্ণক নাও হতে পারে যেমন 5.1 ভগ্নক কিন্তু একে পূর্ণক হিসাবে লেখা সম্ভব নয়। আর এ কারণে কোন operator (অণুক্রিয়া) প্রয়োগের পূর্বে operand (উপাদান) দুটো দুরকম হলে প্রথমে পূর্ণকটিকে ভিতরে ভিতরে ভগ্নকে রূপান্তর করে নেয়া হয়, আর তারপর যোগ, বিয়োগ বা গুণ করা হয় দুটোকে ভগ্নক হিসাবে নিয়েই। এই যে ভিতরে ভিতরে পূর্ণকটি ভগ্নকে রূপান্তর করা হয় এটা এক রকমের type casting (উপাত্ত প্রকারান্তর)। Type casting নিয়ে আমরা পরে আরো বিস্তারিত জানবো, আপাতত int থেকে floatএ casting মনে রাখো।

১৪.৩ Division and Remainder (ভাগফল ও ভাগশেষ)

সিপিপিতে binary (দুয়িক) operator ভাগফল (division) ও ভাগশেষ (remainder) কী ভাবে কাজ করে? যথাযথ program লিখে উদাহরণ সহ বুঝিয়ে দাও। তুমি ইত্যমধ্যে জেনেছো binary operator দুটো operandএর (উপাদান) ওপর প্রযুক্ত হয়ে ফলাফল উৎপন্ন করে।

ফিরিস্তি ১৪.৩: Arithmetic Division Operation (পাটিগণিতের ভাগফল অণুক্রিয়া)

```
cout << "13 / 5 = " << 13 / 5 << endl;
cout << "13.0 / 5 = " << 13.0 / 5 << endl;
cout << "13 / 5.0 = " << 13 / 5.0 << endl;
cout << "13.0 / 5.0 = " << 13.0 / 5.0 << endl;
```

১৪.৩. Division and Remainder (ভাগফল ও ভাগশেষ)

ভাগফলের data type (উপাত্ত প্রকরণ) কেমন হবে সেই নিয়ম আসলে যোগ, বিয়োগ, বা গুণের মতো একই। যদি দুটো operandই (উপাদান) এক রকমের হয় তাহলে ফলাফলও হবে সেই রকমেরই। কিন্তু operand দুটোর একটি যদি হয় পূর্ণক বা int আরেকটি ভগ্নক বা float তাহলে ফলাফল হবে একটি ভগ্নক বা float। এখানেও ভিতরে ভিতরে int প্রথমে float এ type casting (প্রকারান্তর) হয়ে যাবে, ভাগের কাজটি হবে উপাত্ত casting হবার পরে। Data type casting ছাড়াও ভাগের ক্ষেত্রে ভাগশেষ থাকবে কি থাকবে না সেটার একটা ব্যাপার আছে।

ফলন (output)

```
13 / 5 = 2
13.0 / 5 = 2.6
13 / 5.0 = 2.6
13.0 / 5.0 = 2.6
```

উপরের output লক্ষ্য করো, যদি ভাগের operand দুটোর যেকোন একটিও ভগ্নক হয়, যেমন শেষের তিন সারি, তাহলে কিন্তু ভাগশেষের কোন ব্যাপার থাকে না, ফলে আমরা সেক্ষেত্রে ভাগফল পাই 2.6। কিন্তু ভাগের ক্ষেত্রে যদি দুটো operandই পূর্ণক হয়, যেমন প্রথম সারি তাহলে ভাগটি কিন্তু একটু আলাদা। যেমন 13 / 5 করলে আমরা ফলাফল পাই 2 কারণ আমরা জানি এক্ষেত্রে 3 অবশিষ্ট থাকে। ভাগের ক্ষেত্রে আরো একটি গুরুত্বপূর্ণ বিষয় আছে তা হলো operandএর পূর্ণকগুলো ধনাত্নক না ঋণাত্নক। কারণ ঋণাত্নক সংখ্যার ভাগ একটু বিটকেলে হতে পারে। সব মিলিয়ে পূর্ণ সংখ্যার ভাগ আরো বিস্তারিত করে আমরা ভাগশেষের সাথে মিলিয়ে নীচে আলোচনা করবো। তবে একটা কথা মনে রাখবে ভাগের ক্ষেত্রে যদি ভাজক শুন্য হয় যেমন 13 / 0 তাহলে তোমার program চালানোর (run) সময় divide by zero বা শুন্য দিয়ে ভাগ নামে error message (ত্রুটিবার্তা) দেখিয়ে বন্ধ হয়ে যাবে। এই রকম error compileএর (সংকলন) সময় ধরা পড়ে না, কেবল চালানোর (run) সময় বা নির্বাহ (execute) করার সময় ধরা পড়ে, তাই এদেরকে বলা হয় run-time (চলা-কালীন) বা execution-time (নির্বাহ-কালীন) error।

ফিরিস্তি ১৪.৪: Arithmetic Remainder Operation (পাটিগণিতের ভাগশেষ অণুক্রিয়া)

```
cout << "13 / 5 = " << 13 / 5 << "    ";
cout << "13 % 5 = " << 13 % 5 << endl;

cout << "13 / -5 = " << 13 / -5 << "    ";
cout << "13 % -5 = " << 13 % -5 << endl;

cout << "-13 / 5 = " << -13 / 5 << "    ";
cout << "-13 % 5 = " << -13 % 5 << endl;

cout << "-13 / -5 = " << -13 / -5 << "    ";
cout << "-13 % -5 = " << -13 % -5 << endl;

// নীচের সারিগুলো সংকলন (compile) হবে না, ভগ্নকে ভাগশেষ হয় না
// cout << "13.0 % 5 = " << 13.0 % 5 << endl;
// cout << "13.0 % 5.0 = " << 13.0 % 5.0 << endl;
// cout << "13.0 / 5.0 = " << 13.0 / 5.0 << endl;
```

যাইহোক সবশেষে আমরা ভাগশেষ (remainder) দেখি। ভাগের ক্ষেত্রে আমরা আলোচনা করেছি ভগ্নক বা float এর জন্য ভাগশেষের কোন ব্যাপার নেই। কাজেই ভাগশেষ operatorএর (অণুক্রিয়া) operand (উপাদান) দুটোর যে কোন একটিও যদি ভগ্নক হয়, তাহলে ভাগশেষ মোটামুটি অর্থহীন হয়ে যায়। কাজেই এমন কিছু আমাদের programএ (ক্রমলেখ) লিখলে compile (সংকলন) করার সময় error (ত্রুটি) আসবে। উপরের programএর শেষের তিনটি সারি দেখতে পারো যেগুলো comment (টীকা) হিসাবে রাখা আছে। ওইগুলো comment না করে সামনের // হেলানো দাগ দুটো তুলে দিলে programএর অংশ হয়ে যাবে, আর তখন compile (সংকলন) করলে error (ত্রুটি) আসবে, করে দেখতে পারো।

একটা বিষয় খেয়াল করেছো, এখানে আমরা কিন্তু commentএর (টীকা) হেলানো // চিহ্ন দুটোর একরকমের অপব্যবহার করেছি। উপরের programএর শেষ তিনটি সারি আসলে কোন ভাবেই প্রকৃত comment নয়। ওগুলোতো বাংলায় বা ইংরেজীতে লেখা নয়, ওগুলো সিপিপিতে লেখা আর comment চিহ্ন তুলে নিলেই ওগুলো programএর অংশ হয়ে যাবে সহজেই। তবু কেন এখানে আমরা ওগুলোকে commentএর ভিতরে রাখলাম? এটা আসলে একটা খুবই উপকারী কৌশল। Commentএর ভিতরে রাখলে যেহেতু সেটা programএর ঠিক অংশ থাকে না, compile হয় না, কোন error আসার ব্যাপার নাই, আমরা তাই মাঝে মাঝে কিছু কিছু সিপিপিতে লেখা অংশও commentএর ভিতরে রাখি। Program (ক্রমলেখ) লেখার সময় আমরা নানান কিছু পরীক্ষা নিরীক্ষা করি, এভাবে করি, ওভাবে করি। তখন যে অংশগুলো ওই সময় দরকার নাই, চাইলে সেগুলো তো মুছে ফেলা যায়, কিন্তু মুছে ফেললেই তো তোমাকে পরে আবার কষ্ট করে লিখতে হতে পারে। এমতাবস্থায় তুমি যদি ওই অদরকারী অংশটুকুতে comment (টীকা) করে দাও, ব্যস হয়ে গেলো। কোন ঝামেলা নাই, পরে ওই অংশটুকু আবার দরকার হলেই uncomment (টীকা তুলে) করে দিবে। কী চমৎকার কৌশল তাই না! আমরা সবাই এটি হরদম ব্যবহার করি। এখন থেকে এই কৌশল কাজে লাগাবে, কেমন!

ফলন (output)

```
13 / 5 = 2      13 % 5 = 3
13 / −5 = −2     13 % −5 = 3
−13 / 5 = −2     −13 % 5 = −3
−13 / −5 = 2     −13 % −5 = −3
```

এবারে ভাগশেষের ফলাফলের দিকে নজর দেই। ভাগফল সহ আলোচনার সুবিধার জন্য উপরের program (ক্রমলেখ) আর outputএ (ফলন) আমরা ভাগশেষের সাথে সাথে ভাগফলও দেখিয়েছি। আমরা আগেই আলোচনা করেছি ভাগশেষ করা যায় কেবল পূর্ণকের জন্য। ভাগ করলে যা অবশেষ থাকে তাই ভাগশেষ। কিন্তু পূর্ণক তো ধনাত্মকও (positive) হতে পারে, ঋণাত্মকও (negative) হতে পারে। আসলে ঋণাত্মক সংখ্যার ভাগশেষ নিয়েই যতো জটিলতা সৃষ্টি হয়। ঋণাত্মক সংখ্যার ভাগশেষ নিয়ে নানান রকম নিয়ম আছে, আমরা এখানে আলোচনা করছি cpp.sh এ যে নিয়মে ভাগশেষ হয়, সেটা নিয়ে। তুমি যে compiler (সংকলক) দিয়ে program compile (সংকলন) করবে, জেনে নিও সেখানে কেমন হয়। কারো কাছে থেকে জেনে নিতে পারো। অথবা নিজেই উপরের program (ক্রমলেখ) এর মতো করে program তৈরী করে চালিয়ে দেখে নিতে পারো। তেমন কঠিন কিছু নয়।

যাইহোক উপরের output খেয়াল করো। সেখানে কিন্তু কোন ভগ্নক নেই, সবগুলোই পূর্ণক, তবে ধনাত্মক ও ঋণাত্মক আছে। খেয়াল করো ভাগফল ও ভাগশেষ উভয় ক্ষেত্রে মানটা ঠিক পাওয়া যায় চিহ্ন বিবেচনা না করলে। যেমন চারটা ব্যাপারের সবগুলোতেই চিহ্ন বাদ দিলে ভাজক (divisor) আর ভাজ্য (dividend) হয় কেবল 5 আর 13। 13 কে 5 দিয়ে ভাগ করলে ভাগফল হয় 2 আর ভাগশেষ হয় 3। এই পর্যন্ত সবগুলো ব্যাপারেই ঠিক আছে, কিন্তু গোলমাল বাঁধে কেবল চিহ্ন

নিয়ে, ভাগফল বা ভাগশেষ কখন ধনাত্মক + হবে আর কখন ঋণাত্মক − হবে। ভাগফলের ক্ষেত্রে খেয়াল করো যখনই সংখ্যা দুটোর চিহ্ন একই রকম তখন ভাগফল ধনাত্মক যেমন প্রথম ও চতুর্থ সারি, আর যখনই তারা বিপরীত চিহ্নের তখনই ভাগফল ঋণাত্মক যেমন দ্বিতীয় ও তৃতীয় সারি। ভাগশেষের ক্ষেত্রে চিহ্ন নির্ভর করে ভাজ্য (dividend) এর ওপর, ভাজকের ওপর নয়। ভাজ্য যখনই ধনাত্মক যেমন 13, ভাগশেষ তখন ধনাত্মক + হয়েছে। আর ভাজ্য যখন ঋণাত্মক যেমন -13 তখন ভাগশেষ ঋণাত্মক − হয়েছে। ভাগশেষের চিহ্ন 5 বা -5 এর চিহ্নের ওপর নির্ভর করে নাই। একটা বিষয় আগেই বলেছি, ভাগফল ও ভাগশেষের ক্ষেত্রে ভাজক যদি শুন্য হয় তাহলে তোমার program চালানোর সময় divide by zero বা শুন্য দিয়ে ভাগ নামে error message দেখিয়ে বন্ধ হয়ে যাবে। এই রকম error compileএর (সংকলন) সময় ধরা পড়ে না, কেবল চালানোর (run) সময় ধরা পড়ে, তাই এদেরকে বলা হয় run-time error (চলা-কালীন ত্রুটি)।

উপরের উদাহরণগুলোতে আমরা যদিও কেবল সংখ্যাই সরাসরি ব্যবহার করেছি, তুমি কিন্তু চাইলে কোন variable (চলক) বা constant (ধ্রুবক) ব্যবহার করতে পারতে। তুমি চাইলে কোন expression (রাশি) বা functionও (বিপাতক) ব্যবহার করতে পারতে। আসলে r-value (ডান-মান) আছে এরকম যে কোন কিছুই এখানে ব্যবহার করা যেতে পারে। এই আলোচনাগুলো unary operatorএর সময়ই আলোচনা করা হয়েছে, তবুও আবার বলি। function (বিপাতক) এমন একটা জিনিস যে কিছু input (যোগান) নিয়ে কিছু output (ফলন) দেয়। যেমন cstdlib নামক header fileএ (শির নথি) abs(x) নামে একটা function আছে যেটি একটি সংখ্যা input নিয়ে তার চিহ্নটুকু বাদ দিয়ে কেবল মানটুকু output হিসাবে return করে। অর্থাৎ abs(3) হলো 3 আবার abs(-3)ও 3। একই ভাবে abs(3.5) হলো 3.5 আবার abs (-3.5)ও 3.5। expression (রাশি) হল number, constant, variable, operator, function মিলে যখন একটা কিছু তৈরী করা হয় যার মান আছে সেটি, যেমন 3 + x ∗ abs(y) একটি expression যেখানে x আর y হল variable।

```
int a = 4, b = -3;
int const c = 5;

a + 3, c / b, b * c;      //চলক, ধ্রুবক, সংখ্যা
a = c % abs(b);           // abs(b) হল বিপাতক
a = a - ( b * c);         // b * c হল রাশি
```

১৪.৪ Assignment Operator (আরোপণ অণুক্রিয়া)

Assignmentএ (আরোপণ) variableএর জন্য memoryতে (স্মরণি) বরাদ্দকৃত স্থানে মান ভরে দেয়ার ব্যাপারটা আমরা আগে দেখেছি। কিন্তু assignment আসলে একটা operatorও (অণুক্রিয়া) বটে। Assignment একটা operator এই কথার মানে কী? আমরা assignment নিয়া কী কী করতে পারবো?

Assignment (আরোপণ) নিজেও একটা operator (অণুক্রিয়া) এই কথার মানে হলো assignment কিছু operandএর (উপাদান) ওপর প্রযুক্ত হয়ে একটি ফলাফল উৎপন্ন করে। সত্যি বলতে গেলে যোগ, বিয়োগ, গুণ বা ভাগের মতো assignmentও আসলে একটা binary (দুয়িক) operator। কাজেই এটি দুটি operandএর (উপাদান) ওপর প্রযুক্ত হয়। খেয়াল করো Assignmentএর বাম পাশে একটা operand থাকে যার l-value থাকতে হবে অর্থাৎ যার জন্য memoryতে (স্মরণি) জায়গা বরাদ্দ থাকতে হবে, যেমন variable। আর assignmentএর ডা-

নে থাকতে হবে এমন কিছু যার r-value বা মান আছে, যেমন variable (চলক), constant (ধ্রুবক), function (বিপাতক) বা expression (রাশি)। কথা হচ্ছে assignmentএর ফলে উৎপন্ন হওয়া ফলাফলটা কী? আসলে যে মানটি assignmentএর বামপাশের variableএ assign হয় সেই মানটিই assignment operatorএর ফলাফল হিসাবেও বিবেচনা করা হয়।

```
int v = 3, w = -5, x, y, z; //  ভগ্নকও নেয়া যেতে পারে
x = v + 5;     //  চলক x এর মান 8, আরোপণের ফলাফলও 8
y = abs(w);    //  চলক y এর মান 5, আরোপণের ফলাফলও 5
z = x + y;     //  চলক z এর মান 13, আরোপণের ফলাফলও 13
```

উপরে programএ v + 5 বা 3 + 5 অর্থাৎ 8 assign হয়েছে x এ। তারপর, abs(w) function w বা -5 এর মান হতে চিহ্ন ছাড়া 5 ফেরত দিয়েছে যা assign হয়েছে y variableএ। আর শেষে x + y বা 8 + 5 অর্থাৎ 13 assign হয়েছে z variableএ।

তাহলে অন্যান্য operatorএর মতো assignment operatorএরও যেহেতু একটি ফলাফল আছে কাজেই সেই ফলাফলটি অন্য কোন varaible যার l-value আছে তাতে আবারও assignment করা সম্ভব!

```
int v = 3, w = -5, x, y, z; //  ভগ্নকও নেয়া যেতে পারে
x = (v + w);// যোগ অণুক্রিয়ার ফলাফল একটি চলকে আরোপণ
z = (y = x);// ডানের আরোপণের ফলাফল বামেরটিতে আরোপণ
z = v * w;  // গুণ আগে হবে, গুণফল আরোপণ তারপরে হবে
z = y = x;  // ডানের আরোপন আগে, সেই ফল নিয়ে বামের আরোপন
```

সুতরাং কেউ যেমন অনেকগুলো যোগ পরপর লিখতে পারে x + y + z + 3, ঠিক তেমনি চাইলেই কেউ অনেকগুলো assignmentও (আরোপণ) পরপর লিখতে পারে যেমন z = y = x = w। তবে কোন বন্ধনী নাই ধরে নিলে, যোগের ক্ষেত্রে সাধারণত সবচেয়ে বামের যোগটি থেকে শুরু হয়ে যোগগুলো পরপর বাম থেকে ডানে একে একে হতে থাকে। আর assignmentএর (আরোপণ) ক্ষেত্রে সবচেয়ে ডানের assignment হতে শুরু করে assignmentগুলো ডান থেকে বামে একে একে হতে থাকে।

```
int x = 1, y = 2; z = 3;  // আদি মান আরোপণ

x + (y = 3);     // y হলো 3, ফলাফল 1 + 3 বা 4
y = x + (z = 4); // z হলো 4, y হলো 1 + 4 বা 5
z = 5 + (y = z -3); // y হলো 4 - 3 বা 1, z হলো 5 + 1
```

উপরের উদাহরণের শেষ তিনটি সারি খেয়াল করো। Variable declareএর পরের সারির statementএ (বিবৃতি) x + (y = 3); প্রথমে বন্ধনীর ভিতরে y এর মান 3 assign (আরোপণ) হবে আর assignmentএর (আরোপণ) ফলাফলও হবে 3, যা x এর মান 1 সাথে যোগ হয়ে যোগফল হবে 4। এই 4 হলো পুরো রাশিটির মান। এরপরের statementএ y = x + (z = 4); প্রথমে বন্ধনীর ভিতরে z এর মান assign হবে 4 আর ফলাফল ও 4, আর তারপর 4 ও x এর মান 1 এর সাথে যোগ হয়ে হবে 5 যা গিয়ে y চলকে assign হবে। এবারে আসি শেষ statementএ z = 5 + (y = z -3); প্রথমে বন্ধনীর ভিতরে z - 3 হিসাব হবে, z এর মান ঠিক আগের সারিতে হয়েছে 4 সাথে 3 বিয়োগ হলে হয় 1 যা y এ assign হবে আর assignmentএর ফলাফলও (result) হবে 1। এরপর সেই 1 আর 5 যোগ হয়ে ফল হবে 6 যা z এর ভিতরে assign হবে।

১৪.৫ Compound Assignment (যৌগিক আরোপণ)

Compound assignment কী? সিপিপিতে compound assignment কী ভাবে একটি assignmentএর সাথে অন্য একটি operatorএর composition (যোজন) ঘটায়? Self-referential assignmentএর সাথে compound assignmentএর সম্পর্ক কী?

Compound assignment হলো assignmentএর সাথে আর একটি operatorএর composition (যোজন)। Assignment = এর সাথে যোগ + এর যোজন ঘটানোর ফলে নতুন যে operator তৈরী হয় সেটি plus-assignment (যোগ-আরোপণ) +=। ঠিক একই ভাবে assignment = ও বিয়োগ − যুক্ত হয়ে তৈরী হয় minus-assignment (বিয়োগ-আরোপণ) −=, তারপর একই ভাবে times-assignment (গুণ-আরোপণ) ∗=, division-assignment (ভাগফল-আরোপণ) /= আর remainder-assignment (ভাগশেষ-আরোপণ) %=।

```
x += 13;       // এর মানে আসলে x = x + 13;
x −= 7;        //  এর মানে আসলে x = x − 7;
y ∗= x;        // এর মানে আসলে y = y ∗ x;
z /= x + y;    // এর মানে আসলে z = z / (x + y);
z %= abs(3);   // এর মানে আসলে z = z % abs(3);
```

তাহলে উপরের উদাহরণগুলো থেকে দেখা যাচ্ছে প্রতিটি compund assignment আসলে এক একটি self-referential assignment (আত্ম-শরণ আরোপন)। এখানে Compound assignmentএর বাম পাশে যে variableটি থাকে সেটির মানের সাথে সংশ্লিষ্ট পাটিগণিতীয় operator যেমন যোগ, বিয়োগ, গুণ, ভাগফল, বা ভাগশেষ হিসাব করা হয়, আর তারপর ফলাফলটি ওই variableএই assign করা হয়। আসলে compund assignmentগুলো তৈরী করা হয়েছে program রচনার সময় কষ্ট কিঞ্চিত কমানোর জন্য। অনেক সময় assignmentএর বাম পাশে যেটি থাকবে সেটি সহজ সরল variable না হয়ে অন্য কিছু হতে পারে যেটি হয়তো খুবই বড়, সেটির অবশ্যই l-value (বাম-মান) আছে অর্থাৎ তার জন্য memoryতে (স্মরণি) জায়গা দখল করা আছে। যেমন ধরো নীচের উদাহরণে আমরা array (সাজন) ব্যবহার করছি, class (শ্রেণী) ব্যবহার করছি, এগুলো কী এখনই তা জানতে চেয়ো না, আমরা পরে বিস্তারিত করে শিখবো ওগুলো। খালি খেয়াল করো প্রথম দু সারিতে কী ভাবে লম্বা একটা জিনিস assign = চিহ্নের বাম ও ডান উভয় পাশেই আছে। আর খেয়াল করো শেষের সারির statementটি: compund assignment ব্যবহার করে ওই একই বিষয় কত চমৎকার করে সংক্ষেপে লেখা গেছে।

```
this−>amarSajonCholok[suchok] =
    this−>amarSajonCholok[suchok] + amarbriddhi;

this−>amarSajonCholok[suchok] += amarbriddhi;
```

তাহলে দেখলে তো একই জিনিস assign = চিহ্নের বাম পাশে একবার আবার পরক্ষণেই assign = চিহ্নের ডানপাশেও একবার লিখতে হবে, এটি বেশ বিরক্তিকর, আর দেখতেও কত বিরক্তিকর লাগে। তারচেয়ে compund assignment সংক্ষিপ্ত আর বুঝাটাও সহজ। ফলাফলের হিসাবে উভয় ক্ষেত্রে কিন্তু আমরা একই ফলাফল পাবো। তবে মনে রেখো program (ক্রমলেখ) চালাতে সময় কম লাগবে নাকি বেশী লাগবে সেইক্ষেত্রে কিন্তু compound assignmentএর কোন ভুমিকা নেই।

১৪.৬ Increment and Decrement (হ্রাস ও বৃদ্ধি অণুক্রিয়া)

সিপিপিতে লেখা programএ (ক্রমলেখ) আমরা ++ বা −− প্রায়ই দেখতে পাই। এইগুলো কী? একটা যোগ বা বিয়োগ চিহ্ন দেখেছি কিন্তু দুটো যোগ বা বিয়োগ একসাথে তো আজব ব্যাপার! দুটো যোগ বা বিয়োগ এক সাথে দেয়ার সুবিধা-অসুবিধা কী? Program কি এতে দ্রুত চলে?

```
int x = 6, y; //  দুটো চলক একটার আদিমান আছে, আরেকটার নাই
++x;        // এক বেড়ে x হলো 7, y জানিনা কারণ আদিমান নেই
x++;        // এক বেড়ে x হলো 8, y জানিনা কারণ আদিমান নেই
y = ++x;    // এক বেড়ে x হলো 9, তারপর y এ 9 আরোপিত হলো
y = x++;    // প্রথমে y হলো x এর সমান বা 9, পরে x হলো 10
```

উপরের program (ক্রমলেখ) খেয়াল করো। দুটো variable (চলক) নেয়া হয়েছে x আর y। Variable x এর initial value (আদিমান) দেয়া হয়েছে 6, কিন্তু y এর initial value দেয়া হয় নি। এরপর দ্বিতীয় আর তৃতীয় statementএ রয়েছে ++x; আর x++;, খেয়াল করো উভয় ক্ষেত্রে x এর মান এক করে বেড়েছে, এ কারণে অবশ্য ++ কে বলা হয় increment operator (বৃদ্ধি অণুক্রিয়ক)। Increment operator ++ variableএর আগেই দেয়া হউক আর পরেই দেয়া হউক ফলাফল কিন্তু একই। অবশ্য increment ++ আগে ব্যবহার করলে এটিকে pre-increment (পূর্ব-বৃদ্ধি) আর পরে ব্যবহার করলে post-increment (উত্তর-বৃদ্ধি) বলা হয়।

তবে বলে রাখি increment operator (বৃদ্ধি অণুক্রিয়কের) সাথে কিন্তু এমন কিছু ব্যবহার করতে হবে যার l-value (বাম-মান) রয়েছে অর্থাৎ memoryতে (স্মরণি) জায়গা দখল করা আছে। Variableএর (চলক) যেহেতু l-value আছে তাই আমরা variable x ব্যবহার করতে পারলাম। কিন্তু তুমি যদি চাও ++3 বা 3++ লিখবে যাতে 4 পাওয়া যায় অথবা লিখবে (x+3)++ বা ++(x+3), তা লিখতে পারবে না, compile (সংকলন) error হবে। Error হওয়ার কারণ 3 number (সংখ্যা) বা x+3 expressionএর (রাশি) r-value (ডান-মান) তথা value (মান) আছে কিন্তু তাদের l-value (বাম-মান) তথা memoryতে (স্মরণি) জায়গা দখল করা নেই। দরকার নেই তবুও বলে রাখি, তুমি কিন্তু ++ এর সাথে variable x এর বদলে constant জাতীয় কিছু তো এমনিতেই ব্যবহার করতে পারবে না, কারণ constantএর তো মান বদলানো যায় না।

যাইহোক ++ আগেই দেই আর পরেই দেই ++x বা x++ আসলে x+=1; অর্থাৎ x = x+1; এর সমতুল এবং সংক্ষিপ্ত রূপ বলতে পারো। লক্ষ্য করো incrementএ ++ যে 1 বৃদ্ধি ঘটে সেই ব্যাপারটা কিন্তু উহ্য থাকে। ফলে ++ কেবল একটা operandএর (উপাদান) ওপর প্রযুক্ত হয় বলে মনে হয়। আর তাই ++ কে একটি unary (একিক) operator (অণুক্রিয়ক) বলা হয়। কথা হচ্ছে এই unary operatorএর ফলাফলটা কী? ফলাফল তো আমরা আগেই দেখেছি, মান এক বেড়ে যাওয়া। সেটা ঠিক, কিন্তু তাছাড়াও increment operatorএর (বৃদ্ধি অণুক্রিয়া) ফলাফলে কিছু গুরুত্বপূর্ণ বিষয় আছে যে কারণে pre-increment (পূর্ব-বৃদ্ধি) আর post-increment (উত্তর-বৃদ্ধি) একের থেকে অন্যে বেশ খানিকটা আলাদা।

Pre-increment (পূর্ব-বৃদ্ধি) আর post-increment (উত্তর বৃদ্ধি) যে আলাদা তা পরিস্কার হবে উপরের programএর (ক্রমলেখ) শেষের সারি দুটো দেখলে। যখন y = ++x; করা হয়েছে তখন x এর মান আগে বেড়ে হয়েছে 9 আর তারপর xএর সেই বেড়ে যাওয়া মান 9ই y এ assign (আরোপ) হয়েছে। কিন্তু যখন y = x++; তখন কিন্তু খেয়াল করো আগে x এর মান y এ assign হয়েছে ফলে y হয়েছে 9 আর তারপর x এর মান বেড়েছে 1 ফলে হয়েছে 10। আচ্ছা y = ++x; আর y = x++; এ দুটোকে যদি আমরা বৃদ্ধি ++ ব্যবহার না করে লিখতাম তাহলে কেমন হতো? আমাদের অবশ্যই দুটো করে statement লিখতে হতো। নীচে লক্ষ্য করো y = ++x; এ আগে

মান বাড়ানো পরে assignment, আর y = x++; এ আগে assignment পরে মান বাড়ানো। আশা করা যায় pre- (পূর্ব-) ও post-increment (উত্তর-বৃদ্ধির) তফাৎ পরিস্কার হয়েছে।

```
x = x + 1;   //  y= ++x; এ x এর মান বৃদ্ধি আগে ঘটবে
y = x;       //  y= ++x; এ y তে x এর মান আরোপন পরে
y = x;       //  y= x++; এ y তে x এর মান আরোপন আগে
x = x + 1;   //  y= x++; এ x এর মান বৃদ্ধি তার পরে
```

Pre-increment (পূর্ব-বৃদ্ধি) আর post-increment (উত্তর-বৃদ্ধির) আরো একটা পার্থক্যও জানা দরকার অবশ্য। সেটা হলো pre-incrementএর ফলাফল আসলে একটা l-value (বাম-মান) এক্ষেত্রে variableটির l-value, অন্যদিকে post-incrementএর ফলাফল আসলে একটা r-value (ডান-মান)। আগেই বলেছি increment operatorএর (পূর্ব-বৃদ্ধি) সাথে ব্যবহৃত operandএর (উপাদান) অবশ্যই l-value থাকতে হবে। ফলে post-incrementএর ফলাফলের ওপরে আবার কোন বৃদ্ধিই চালানো যায় না, কিন্তু pre-incrementএর ফলাফলের ওপর চালানো যায়। তুমি যদি পরীক্ষা করতে চাও তাহলে ++++x; বা (++x)++; চেষ্টা করো, compile (সংকলন) হয়ে যাবে, কিন্তু x++++ বা ++(x++) চেষ্টা করো, compile হবে না, error (ত্রুটি) আসবে পরের বৃদ্ধিটার জন্য "l-value required"। তুমি যদি স্রেফ ++x++; লিখো, এটা কিন্তু compile হবে না, error দেখাবে, কারণ হলো pre- ও post-incrementএর মধ্যে post-incrementএর precedence (অগ্রগণ্যতা) আগে, ফলে ++x++ আসলে ++(x ++) এর সমতুল। Precedence order (অগ্রগণ্যতার ক্রম) হলো কোন operator আগে হবে কোনটা পরে হবে তার নিয়ম। এ বিষয়ে আমরা পরে বিস্তারিত জানবো।

এবারে আমরা increment ব্যবহারে programএর গতির ওপর প্রভাব নিয়ে একটু আলোচনা করি। increment (বৃদ্ধি) ++x বা x++ সাধারণত x+=1 বা x=x+1 এর চেয়ে দ্রুতগতির, এর কারণ মুলত একদম যন্ত্র পর্যায়ে x++ বা ++x বিশেষ ভাবে নির্বাহিত হয় কিন্তু x+=1 বা x=x+1 সাধারণ যোগের মতো করে নির্বাহিত হয়। সাধারণতো pre-increment (পূর্ব-বৃদ্ধি) আর post-incrementএর (উত্তর-বৃদ্ধি) মধ্যে পূর্ব-বৃদ্ধি দ্রুত গতির। কারণ হলো, post-incrementএর ফলাফল যেহেতু x এর মান বৃদ্ধি করবার আগের মান, তাই ওই আগের মানটি প্রথমে কোথাও ক্ষণস্থায়ী (temporarily) ভাবে রেখে দিতে হয়, আর x এর মান বৃদ্ধিটা তারপর ঘটে, আর তারপর ক্ষণস্থায়ী ভাবে রাখা মানটা ফলাফল হিসাবে আসে যেটি y = x++; এর ক্ষেত্রে y এ assign হয়। কিন্তু পূর্ব-বৃদ্ধির ক্ষেত্রে মান বৃদ্ধি আগে ঘটে আর ফলাফলটাও সেই বৃদ্ধিপ্রাপ্ত মানই, কাজেই ক্ষণস্থায়ী ভাবে আগের মান রেখে দেওয়ার কোন বোঝা (overhead) এখানে নেই। মোটকথা pre-increment সরাসরি l-valueএর ওপরই কাজ করে অর্থাৎ ++x এ সরাসরি variableটার ওপরই কাজ করে, আর কোন ক্ষণস্থায়ী কিছু দরকার হয় না। এ কারণে pre-increment ++x; post-increment x++; এর চেয়ে বেশী দ্রুতগতির হয়ে থাকে। কাজেই তুমি পারতো পক্ষে ++x ব্যবহার করবে, x++ ব্যবহার করবে না।

Programএ increment ব্যবহারে এবারে একটা পরামর্শ দেই। Pre-increment ও post-increment নিয়ে অনেক রকম খেলা যায়, যেমন তুমি চাইলে x = (++x)++ + ++x; এর মতো অনেকগুলো + চিহ্ন দিয়ে কিছু একটা লিখতে পারো। এই রকম জটিল statementগুলো হয়তো compile (সংকলন) হবে। এর ফলে ফলাফলও কিছু একটা আসবে, যেটা চাইলে বুঝা সম্ভব, কিন্তু বুঝতে গেলে মাথা বেশ গরম হয়ে যায়। আমার পরামর্শ হলো এইরকম জটিল statement পারতো পক্ষে লেখবে না। সবসময় এমন ভাবে code (সংকেত) লিখবে যাতে পরে তুমি বা অন্য কেউ তেমন কোন কষ্ট ছাড়াই তোমার code দেখে বুঝতে পারে। মনে রাখবে code যত জটিল, তার ভুল বের করাও তত কঠিন।

উপরের পুরো আলোচনাতে আমরা কেবল increment (বৃদ্ধি) নিয়ে আলোচনা করেছি। আসলে decrement (হ্রাস) −− নিয়ে আলোচনাটা একদম একই রকম। আমরা তাই পুনরাবৃত্তি করবো না। কেবল জেনে রাখো decrementএর (হ্রাস) ফলে মান 1 কমে যায়। তাই −−x বা x−− হলো x −= 1 বা x = x − 1 এর সমতুল। আমরা −−x কে pre-decrement (পূর্ব হ্রাস) আর x−− কে post-increment (উত্তর-হ্রাস) বলি। Pre-decrementএর তুলনায় post-decrementএর precedence (অগ্রগণ্যতা) বেশী। গতির দিক বিবেচনায় pre-decrement, post-decrementএর চেয়ে শ্রেয়তর।

১৪.৭ Comma Operator (বির্তি অণুক্রিয়া)

সিপিপিতে comma operator (বির্তি অণুক্রিয়া) কয়েকটি expression (রাশি) কে এক সাথে পরপর লেখায় সাহায্য করে। Comma (বির্তি) operatorএর বামপাশের operandএর (উপাদান) মান সব সময় void (নর্থক) হয় আর উপেক্ষিত হয়। এর অর্থ হচ্ছে ডান পাশের operandটির (উপাদান) মানই comma operatorএর (বির্তি অণুক্রিয়া) ফলাফল হয়।

একটা উদাহরণ দেখি x = (y=3, y+1); এই statementএর ফলে বন্ধনীর ভিতরে প্রথমে commarএর বাম পাশের expression হিসাবে y এর মান assign (আরোপ) হবে 3। যদিও assignmentএর কারণে আমরা y এ 3 assignmentএর পাশাপাশি ফলাফলও পাই 3, কিন্তু commaএর (বির্তি) কারণে সেই ফলাফল বাদ গিয়ে ফলাফল হয়ে যাবে void (নর্থক)। যাইহোক এরপর commaএর (বির্তি) ডান পাশের expression হিসাবে y+1 এর মান 3+1 বা 4 হবে যেটি আসলে যোগেরও + ফলাফল। আর যোগের এই ফলাফল 4 ই শেষ পর্যন্ত x varaibleএ assign হবে। এখানে বন্ধনী দরকার কারণ comma (বির্তি) , সাধারণত assignmentএর (আরোপণ) = পরে হিসাব করা হয়। আমরা বন্ধনীর ভিতরের assignmentটি y = 3 commaএর (বির্তি) আগে করতে চাইলেও বন্ধনীর বাইরের variable x এ assignmentটি commaএর পরে করতে চাই, আর এ কারণে বন্ধনী জরুরী। ব্যাপারটি আরো পরিস্কার বুঝতে চাইলে একই জিনিস বন্ধনী ছাড়া কী হবে দেখো x = y = 3, y + 1;। এখানে দুটো assignmentই (আরোপণ) commaএর (বির্তি) আগে execute (নির্বাহ) হবে। ফলে প্রথমে y এর মান assign হবে 3, তারপর x এও মান 3ই assign হবে, তারপর y+1 হিসাব হবে 4। এই 4 commaএর ফলাফল হলেও সেটি কিন্তু এখানে কিছুতে assign হয় নি।

Comma (বির্তি) operator (অণুক্রিয়া) হিসাবে ব্যবহার হলেও এর আরো নানান ব্যবহার আছে সিপিপিতে। যেমন একাধিক variable (চলক) একসাথে declare (ঘোষণা) করতে আমরা comma (বির্তি) দিয়ে লিখি int x, y, z = 3; Commaএর (বির্তি) এই রকম ব্যবহার আসলে operator হিসাবে নয়, বরং তালিকার separator (পৃথকী) হিসাবে ব্যবহার। আমরা যখন পরে for loop (জন্য ঘূর্ণী) ও parameter (পরামিতি) নিয়ে আলোচনা করবো তখনও list separator (তালিকা পৃথকী) হিসাবে commaএর (বির্তি) ব্যবহার দেখতে পাবো।

১৪.৮ Precedence Order (অগ্রগণ্যতার ক্রম)

Precedence order (অগ্রগণ্যতার ক্রম) কী? সিপিপিতে এ পর্যন্ত পরিচিত হওয়া operatorগুলোর (অণুক্রিয়া) precedence order (অগ্রগণ্যতার ক্রম) আলোচনা করো।

ধরো তুমি 3 + 4∗5 + 6 এর মান হিসাব করবে। আগেকার দিনে এক রকম সস্তা calculator (কলনি) পাওয়া যেতো যেটি করতো কী, বাম থেকে হিসাব করতো একের পর এক। ফলে সেটা প্র-

থম 3 ও 4 যোগ করে 7 বের করতো, তারপর তার সাথে 5 গুণ করে বের করতো 35 আর শেষে তার সাথে 6 যোগ করে ফল দিতো 41। তুমি চাইলে উল্টো আরেক রকমের অবস্থা ভাবতে পারো, যেখানে ডান দিক থেকে একের পর এক হিসাব হবে। সুতরাং 5 ও 6 যোগ করে 11, তারসাথে 4 গুণ করে 44, শেষে 3 যোগ করে 47। কিন্তু ছোটবেলা থেকে সরলের নিয়ম আমরা শিখে এসেছি: গুণ আগে হবে যোগ পরে হবে। আমরা তাই হিসাব করি 4 ও 5 এর গুণ আগে ফল 20 তার সাথে বামের যোগ আগে, তাই 3 আগে যোগ হলো 23, শেষে ডানের যোগ তাই 6 যোগ করে হলো 29, যেটাকে আমরা সঠিক হিসাব বলে ধরে নেই। এই যে বাম থেকে ডানে বা ডান থেকে বামে হিসাব না করে গুণ যোগের আগে করতে হবে, আবার দুটো যোগ পর পর থাকলে বামের যোগ আগে করতে হবে। এই নিয়মগুলোকে precedence order (অগ্রগণ্যতার ক্রম) বলা হয়।

সরল অংকে precedence order ছিল: বন্ধণী, এর, ভাগ, গুণ, যোগ, বিয়োগ। সবেচেয়ে ভিতরের বন্ধনী সবেচেয়ে আগে। ভাগ আর গুণ আসলে বাম থেকে যেটা আগে আসে। একই ভাবে যোগ ও বিয়োগ বাম থেকে যেটা আগে আসে। সিপিপিতে আমরা এ পর্যন্ত অনেকগুলো operatorএর (অণুক্রিয়া) সাথে পরিচিত হয়েছি। এগুলো হলো unary + − ++ −− binary + − ∗ / % = += −= ∗= /= %= , তো এদের মধ্যে unary operatorএর (একিক অণুক্রিয়া) ক্রম সবার আগে, তারপর binary operatorগুলোর (দুয়িক অণুক্রিয়া) ক্রম। আমরা আপাতত কেবল এগুলোর precedence order (অগ্রগণ্যতার ক্রম) বিবেচনা করবো। অন্যান্য আরো operator ও তাদের ক্রম সম্পর্কে আমরা পরে জানবো।

১. ++ −− ২টি unary operator (একিক অণুক্রিয়া) post-increment (উত্তর-বৃদ্ধি) ও post-decrement (উত্তর-হ্রাস) x++, x−− এরা l-valueএর (বাম-মান) ওপরে প্রযুক্ত হয়ে r-value (ডান-মান) ফল দেয়। ফলে x++++ বা x−−−− করা যায় না।

২. ++ −− + − ৪টি unary operator (একিক অণুক্রিয়া) pre-increment (পূর্ব-বৃদ্ধি) ++x ও pre-decrement (পূর্ব-হ্রাস) −−x এরা l-valueএর (বাম-মান) ওপর প্রযুক্ত হয়ে lef-valueই ফল দেয়। ফলে ++++x বা −−−−x করা যায়, আর সবেচেয়ে ডানের ++ বা −− আগে প্রযুক্ত হয়। (pre) unary operator (একিক অণুক্রিয়া) +x ধনাত্মক (positive) −x আর ঋণাত্মক (negative) এরা r-valueএর (ডান-মান) ওপর প্রযুক্ত হয়ে r-valueই দেয়। ফলে + + x বা − − x করা সম্ভব, খেয়াল করো দুটো + বা দুটো − এর মধ্যে ফাঁকা দিতে হয়েছে না হলে ওগুলো বৃদ্ধি বা হ্রাস হিসাবে চিহ্নিত হয়ে যাবে।

৩. ∗ / % ৩টি binary operaror (দুয়িক অণুক্রিয়া) এরা দুটি r-valueএর (ডান-মান) operandএর ওপর প্রযুক্ত হয়ে r-valueই ফল দেয়। এই operatorগুলো পরপর অনেকগুলো থাকলে বাম থেকে ডানে একে একে হিসাব হতে থাকে। যেমন 10 / 2 ∗ 4 % 6 এ বাম থেকে ডানে প্রথমে ভাগফল, তারপর গুণফল, তারপর ভাগশেষ হিসাব হবে।

৪. + − ২টি binary operator (দুয়িক অণুক্রিয়া) এরা দুটি r-valueএর operandএর (উপাদান) ওপর প্রযুক্ত হয়ে r-valueই ফল দেয়। এই operatorগুলো পরপর অনেকগুলো থাকলে বাম থেকে ডানে একে একে হিসাব হতে থাকে। যেমন 10 − 2 + 5 এ বাম থেকে ডানে প্রথমে বিয়োগফল, তারপর যোগফল হিসাব হবে।

৫. = += −= ∗= /= %= এই সব binary operator (দুয়িক অণুক্রিয়া) assignmentগুলোর বামপাশে এমন কিছু থাকতে হবে যার l-value (বাম-মান) আছে, আর ডান পাশে এমন কিছু থাকতে হয় যার r-value (ডান-মান) আছে। এই operatorগুলো পরপর অনেগুলো থাকলে ডান থেকে বামে একে একে হিসাব হতে থাকে। যেমন x += y = z ∗= 3 তে প্রথমে ডানের ∗= এর কারণে z এর সাথে 3 গুণ হবে, তারপর মাঝের = এর কারণে z এর মান y assign হবে, শেষে y এর মান x এর সাথে যোগ হবে।

৬. , comma (বির্তি) একটি binary operator (দুয়িক অণুক্রিয়া) যেটির ফলাফল কেবল ডানপাশের operand (উপাদান)। বাম পাশের operandটি হিসাব হয়, কিন্তু তার ফলাফল হবে void (নর্থক)। এই operator একাধিক পরপর থাকলে, বাম থেকে ডানে একে একে হিসাব হতে থাকে। যেমন x + 2, y * 3, z / 4 প্রথমে যোগ হবে, তারপর গুণ আর শেষে ভাগ, ফলাফল হবে একদম ডানের ভাগফলটিই।

দুটো একই বা একই ক্রমের operator পরপর থাকলে কোন পাশেরটি আগে হবে এইটি নির্ধারণ করে দেয়াকে বলা হয় associativity (সহযোজ্যতা)। যেমন x−y−z থাকলে আমাদের প্রথমে বামের বিয়োগ করতে হবে, তারপর ডানের বিয়োগ, তাই বিয়োগ হল left associative (বাম সহযোজ্য) অর্থাৎ x − y − z আর (x −y) − z একই। খেয়াল করো বিয়োগ কিন্তু right associative (ডান সহযোজ্য) নয় কারণ x − y − z আর x − (y − z) এক নয়। যোগ আবার বাম ও ডান both associative (উভয় সহযোজ্য) বা সংক্ষেপে associative (সহযোজ্য) কারণ x + y + z, (x + y) + z ও x + (y + z) একই। সাধারণত both associativeদের ক্ষেত্রে সুবিধার্থে তাদের left-associative হিসাবে বিবেচনা করা হয়। উপরের তালিকায় আলোচিত operatorগুলোর ক্ষেত্রে একই রকম operator পরপর থাকলে কোন পাশেরটি আগে হবে, সেটাও কিন্তু আলোচনা করা হয়েছে। সেখান থেকে বুঝতো পারো কোন operator টি এখানে left associative (বাম সহযোজ্য), আর কোনটি right associative (ডান সহযোজ্য)?

সবেশেষে একটা গুরুত্বপূর্ণ বিষয় মনে রাখবে বন্ধনী () এর শক্তি কিন্তু সবচেয়ে বেশী। যে কোন স্থানে কোন রকমের দ্বিধাদ্বন্দ্ব থাকলে সেখানে বন্ধনী ব্যবহার করে দ্বিধা পরিস্কার করবে। Operatorগুলোর (অণুক্রিয়া) precedence order (অগ্রগণ্যতার ক্রম) ব্যবহার করে নানা রকম জটিল জটিল statement (বিবৃতি) ও expression (রাশি) তৈরী করা যায়, যেগুলো ক্রম বিবেচনায় নিয়ে বুঝতে গেলে মাথা গরম হয়ে যেতে পারে, ভুল হলে বের করা কঠিন হয়ে যাবে। কাজেই আমার পরামর্শ হচ্ছে তোমার statement বা expression অবশ্যই সহজে পাঠযোগ্য হতে হবে, আর এ কাজে যত দরকার বন্ধনী ব্যবহার করবে। যেমন ধরো x += y − z এর চেয়ে x += (y − z) বুঝা আমাদের জন্য বেশী সহজ, কারণ এতে একদম পরিস্কার বিয়োগ আগে হবে।

১৪.৯ Mathematical Problems (গাণিতিক সমস্যা)

Two dimensional coordinate systemএ (দ্বিমাত্রিক স্থানাঙ্ক ব্যবস্থায়) দুটি বিন্দুর স্থানাঙ্ক input (যোগান) নিয়ে তাদের মাঝে দূরত্ব output (ফলন) হিসাবে দেখাও। ধরো স্থানাঙ্কগুলো ভগ্নকে দেয়া আছে। তোমার নিশ্চয় জানা আছে যে দুটো বিন্দুর (x_1, y_1) ও (x_2, y_2) দূরত্ব হলো $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ অর্থাৎ ভুজদ্বয়ের দূরত্বের বর্গ ও কোটিদ্বয়ের দূরত্বের বর্গের যোগফলের বর্গমূল। বর্গমূল নির্ণয়ের জন্য cmath header file (শির নথি) থেকে sqrt function ব্যবহার করো। আর বর্গ নির্ণয়ের জন্য তোমাকে একই জিনিস দুইবার গুণ করতে হবে।

ফিরিস্তি ১৪.৫: দুটি বিন্দুর মধ্যের দূরত্ব (Distance Between Two Points)

```
// নীচের শির নথি main বিপাতকের বাইরে অন্তর্ভুক্ত করো

#include <cmath>  // বর্গমূল নির্ণয়ের জন্য  sqrt বিপাতক লাগবে

// নীচের অংশ main বিপাতকের ভিতরে return এর আগে লিখো

float x1, y1, x2, y2; // স্থানাঙ্ক দুটো (x1,y1), (x2,y2)
```

```
cout << "first point x y: ";        // যোগান যাচনা
cin >> x1 >> y1;                    // প্রথম বিন্দু যোগান
cout << "second point x y: ";       // যোগান যাচনা
cin >> x2 >> y2;                    // দ্বিতীয় বিন্দু যোগান

float xd = abs(x1 - x2);            // ভুজ দুটির দূরত্ব
float yd = abs(y1 - y2);            // কোটি দুটির দূরত্ব

float dd = sqrt(xd * xd + yd * yd); // দূরত্ব হিসাব করো
cout << "distance in between " << dd << endl;// ফলন
```

উপরের program খেয়াল করো। খুবই সাদামাটা। প্রথমে main functionএর বাইরে cmath header file অন্তর্ভুক্ত করতে হবে বলে দেখানো হয়েছে। তারপর main functionএর ভিতরে বিন্দু দুটোর ভুজ ও কোটি ধারণ করার জন্য চারটি float ধরনের ভগ্নক variable (চলক) নেয়া হয়েছে। এরপর input prompt (যোগান যাচনা) দিয়ে বিন্দুদুটোর স্থানাঙ্ক input (যোগান) নেয়া হয়েছে। তারপর ভুজ দ্বয়ের দূরত্ব abs(x1 − x2) বের করে xd নামের আরেকটি variableএ নেয়া হয়েছে, একই ভাবে কোটিদ্বয়ের দূরত্ব abs(y1 − y2) বের করে yd নামের আরেকটি variableএ নেয়া হয়েছে। মনে করে দেখো abs functionটি (বিপাতক) কোন সংখ্যার absolute value (পরম মান) অর্থাৎ চিহ্ন বাদ দিয়ে কেবল মানটুকু ফেরত দেয়। যাইহোক তারপর xd এর বর্গ ও yd এর বর্গের যোগফল নিয়ে তার বর্গমূল বের করা হয়েছে sqrt function ব্যবহার করে আর রাখা হয়েছে dd variableএ। সবশেষে দূরত্ব dd variable থেকে output (ফলন) দেয়া হয়েছে। এখানে একটা কথা বলে রাখি sqrt(xd ∗ xd + yd ∗ yd) এর বদলে cmath header file (শির নথি) থেকেই hypot নামের functionও (বিপাতক) আমরা ব্যবহার করতে পারতাম। সেক্ষেত্রে আমাদের লিখতে হতো hypot(xd, yd) আর সেটি ঠিক একই কাজ করতো। Function hypot আসলে সমকোণী ত্রিভুজের অতিভুজের দৈর্ঘ্য নির্ণয় করে, কিন্তু তার সূত্র আর দুটো বিন্দুর দূরত্ব নির্ণয়ের সূত্রের মধ্যে মিল রয়েছে।

১৪.১০ Header File cmath (শির নথি cmath)

Header file cmath এ mathematical processingএ ব্যবহৃতব্য নানান function (বিপাতক) আছে। আমরা এখানে ওই functionগুলোর সাথে সংক্ষিপ্ত আকারে পরিচিত হবো। এই functionগুলো কী তা বুঝতে তোমার উচ্চমাধ্যমিক গণিতের ধারণাবলী দরকার হবে। নীচের hyperbolic (পরাবৃত্তীয়) functionগুলো ছাড়া প্রায় সবগুলো functionই আমাদের প্রায়শই কাজে লাগে।

Mathematical Functions (গাণিতিক বিপাতক)

- abs(x): কোন সংখ্যা x এর পরম মান। abs(3) হবে 3 এবং abs(−3) হবে 3।

Trigonometric Functions (ত্রিকোণমিতিক বিপাতক)

- cos(x): Cosine (লগ্নানুপাত) যেখানে x হল রেডিয়ানে।

- sin(x): Sine (লম্বানুপাত) যেখানে x হল রেডিয়ানে।
- tan(x): Tangent (স্পর্শানুপাত) যেখানে x হল রেডিয়ানে।
- acos(x): Arc-cosine (বিলগ্নানুপাত) যেখানে ফেরত মান রেডিয়ানে।
- asin(x): Arc-sine (বিলম্বানুপাত) যেখানে ফেরত মান রেডিয়ানে।
- atan(x): Arc-tangent (বিস্পর্শানুপাত) যেখানে ফেরত মান রেডিয়ানে।
- atan2(x,y): Arc-tangent (বিস্পর্শানুপাত) যেখানে $\frac{x}{y}$ এর x হল লব (numerator) আর y হল হর (denominator) আর ফেরত মান রেডিয়ানে।

Hyperbolic Functions (পরাবৃত্তীয় বিপাতক)

- cosh(x): Hyperbolic cosine (পরাবৃত্তীয় লগ্নানুপাত) যেখানে x হল রেডিয়ানে।
- sinh(x): Hyperbolic sine (পরাবৃত্তীয় লম্বানুপাত) যেখানে x হল রেডিয়ানে।
- tanh(x): Hyperbolic tangent (পরাবৃত্তীয় স্পর্শানুপাত) যেখানে x হল রেডিয়ানে।
- acosh(x): Hyperbolic arc-cosine (পরাবৃত্তীয় বিলগ্নানুপাত), ফেরত রেডিয়ানে।
- asinh(x): Hyperbolic arc-sine (পরাবৃত্তীয় বিলম্বানুপাত), ফেরত রেডিয়ানে।
- atanh(x): Hyperbolic arc-tangent (পরাবৃত্তীয় বিস্পর্শানুপাত), ফেরত রেডিয়ানে।

Exponents and Logarithms (সুচক ও ঘাতাঙ্ক)

- exp(x): e^x বা exponential function (সুচকীয় বিপাতক)
- log(x): $\log_e x$ বা logarithmic function (ঘাতাঙ্ক বিপাতক)
- log10(x): $log_{10}x$ বা ১০-ভিত্তিক logarithm (ঘাতাঙ্ক)
- exp2(x): 2^x বা ২-ভিত্তিক exponential (সুচকীয়) function
- log2(x): log_2x বা ২-ভিত্তিক logarithm (ঘাতাঙ্ক)

Power and Index (শক্তি ও ঘাত)

- pow(x,y): x^y অর্থাৎ x এর y তম শক্তি যেমন pow(2,3) হল 2^3 বা 8
- sqrt(x): $\sqrt{x}$ অর্থাৎ x এর বর্গমূল যেমন sqrt(16.0) হল 4.0
- cbrt(x): $\sqrt[3]{x}$ অর্থাৎ x এর ঘনমূল যেমন cbrt(8.0) হল 2.0
- hypot(x,y): $\sqrt{x^2+y^2}$ অর্থাৎ x ও y কে সমকোনী ত্রিভুজের লম্ব (perpendicular) ও ভুমি (base) ধরলে অতিভুজের (hypotenuse) দৈর্ঘ্য

Rounding Functions (নৈকট্যায়নের বিপাতক)

- round(x): নৈকট্যায়ন বিপাতক x এর নিকটতম পূর্ণক।
- floor(x): মেঝে বিপাতক x এর সমান বা ঠিক ছোট পূর্ণকটি।
- ceil (x): ছাদ বিপাতক x এর সমান বা ঠিক বড় পূর্ণকটি।
- trunc(x): কর্তন বিপাতক x এর ভগ্নাংশটুকু কেটে ফেলবে।

উপরের functionগুলোর ফলাফল বুঝার জন্য নীচের সারণী লক্ষ্য করো।

মান x	নৈকট্যায়ন round(x)	মেঝে floor(x)	ছাদ ceil (x)	কর্তন trunc(x)
2.3	2.0	2.0	3.0	2.0
2.8	3.0	2.0	3.0	2.0
2.5	3.0	2.0	3.0	2.0
2.0	2.0	2.0	2.0	2.0
-2.3	-2.0	-3.0	-2.0	-2.0
-2.8	-3.0	-3.0	-2.0	-2.0
-2.5	-3.0	-3.0	-2.0	-2.0

১৪.১১ Exercise Problems (অনুশীলনী সমস্যা)

Conceptual Questions: নীচে কিছু conceptual প্রশ্ন রয়েছে। প্রশ্নগুলোর উত্তর নিজে নিজে বের করবে।

১. Function (বিপাতক) ও expression (রাশি) বলতে কী বুঝো? উদাহরণ দাও।

২. Unary (একিক) ও binary (দুয়িক) operation (অণুক্রিয়া) বলতে কী বুঝো? কয়েকটা করে unary (একিক) ও binary (দুয়িক) operationএর (অণুক্রিয়া) নাম বলো।

৩. Type casting (উপাত্ত প্রকারান্তর) কী? Binary operationএ (দুয়িক অণুক্রিয়ায়) কী ভাবে type casting (উপাত্ত প্রকারান্তর) হয়?

৪. Execution-time error (নির্বাহ-কালীন ত্রুটি) বলতে কী বুঝো? ভাগফল ও ভাগশেষ নির্ণয়ের সময় কোন execution-time error ঘটতে পারে?

৫. Programএ অদরকারী code (সংকেত) মুছে না দিয়ে কীভাবে আমরা comment (টীকা) ব্যবহার করে সেগুলোকে অকার্যকর করে রাখতে পারি, ব্যাখ্যা করো।

৬. ঋণাত্মক পূর্ণকের (integer) ভাগফল ও ভাগশেষ নির্ণয়ের নিয়ম বর্ণনা করো।

৭. Assignment operatorএর (আরোপণ অণুক্রিয়া) ফলাফল ঠিক কী? Compound assignment (যৌগিক আরোপণ) বলতে কী বুঝো? কয়েকটি উদাহরণ দাও।

৮. সাধারণ compound assignment (যৌগিক আরোপন) যেমন x += 1 ব্যবহার না করে কেন increment (বৃদ্ধি) x++ বা ++x কেন ব্যবহার করা হয়?

৯. Post-increment (উত্তর-বৃদ্ধি) ও pre-incrementএর (পূর্ব-বৃদ্ধি) মধ্যে পার্থক্যগুলো আলোচনা করো। তুমি কোনটি ব্যবহার করতে চাইবে এবং কেন?

১০. Comma (বির্তি) operatorএর কাজ কী? এর ফলাফলই বা কী?

১১. Precedence order (অগ্রগণ্যতার ক্রম) ও associativity (সহযোজ্যতা) কী?

১২. সিপিপিতে এ পর্যন্ত শেখা operatorগুলোর (অণুক্রিয়া) precedence order (অগ্রগণ্যতার ক্রম) ও associativity (সহযোজ্যতা) আলোচনা করো।

Programming Problems: নীচে আমরা কিছু programming problems দেখবো। এই সমস্যাগুলো আগে ধৈর্য্য ধরে নিজে নিজে সমাধান করতে চেষ্টা করবে। যখন একেবারেই পারছো না বলে মনে হয় তখনই কেবল সমাধান দেখে নিতে পারো। সমাধানগুলো programming problemগুলোর শেষে আছে।

১. একটি arithmetic seriesএর (সমান্তর ধারা) প্রথম পদ a সাধারণ অন্তর d হলে n-তম পদ কতো? n পদের সমষ্টিই বা কত? এর জন্য সিপিপিতে একটা program (ক্রমলেখ) (program) তৈরী করো যেটি a, d, ও n input (যোগান) নিবে, আর n-তম পদ ও n পদের সমষ্টি output (ফলন) দিবে। এর জন্য তুমি সূত্র ব্যবহার করবে n-তম পদ = $a+(n-1)*d$ আর n পদের সমষ্টি = $n*(2a+(n-1)*d)/2$। প্রদত্ত বিভিন্ন ধারার জন্যে এই সূত্র a আর d বসালে আমরা ওই ধারাগুলোর জন্য সরাসরি সূত্র পেতে পারি।

- $1+2+3+\ldots$ ধারাতে $a=1, d=1$। সুতরাং n-তম পদ = $a+n-1$, n পদের সমষ্টি = $n(n+1)/2$। যেমন $n=10$ হলে 10-তম পদ 10, সমষ্টি 55।
- $2+4+6+\ldots$ ধারাতে $a=2, d=2$। সুতরাং n-তম পদ = $2n$, n পদের সমষ্টি = $n(n+1)$। যেমন $n=10$ হলে 10-তম পদ 20, সমষ্টি 110।
- $1+3+5+\ldots$ ধারাতে $a=1, d=2$। সুতরাং n-তম পদ = $2n-1$, n পদের সমষ্টি = n^2। যেমন $n=10$ হলে 10-তম পদ 19, সমষ্টি 100।

২. নীচের মতো output (ফলন) দেয় এরকম একটি program (ক্রমলেখ) তৈরী করো। Result columnএ তুমি binary operatorগুলো (দুয়িক অণুক্রিয়া) ব্যবহার করবে।

```
x=10  y=5

expr  result
x=y+3  x=  8
x=y-2  x=  3
x=y*5  x=  25
x=x/y  x=  2
x=x%y  x=  0
```

৩. এমন একটি program (ক্রমলেখ) রচনা করো যেটি একটি তিন অঙ্কের সংখ্যাকে উল্টো করে যেমন 326 হয়ে যায় 623। এ কাজে তুমি ভাগফল, ভাগশেষ, গুণ, যোগ ও বিয়োগ ব্যবহার করবে। 326 থেকে অঙ্কগুলো আলাদা করে তারপর 623 তৈরী করবে।

৪. একটি ত্রিভুজের তিন বাহুর দৈর্ঘ্য a, b, c input (যোগান) নিয়ে ত্রিভুজটির ক্ষেত্রফল নির্ণয় করো। তুমি হয়তো জানো ত্রিভুজের ক্ষেত্রফল = $\sqrt{s(s-a)(s-b)(s-c)}$ যেখানে s হলো অর্ধ পরিসীমা অর্থাৎ $s = (a+b+c)/2$।

৫. এমন একটি program (ক্রমলেখ) রচনা করো যেটি সেকেন্ড input নিয়ে তাকে ঘন্টা-মিনিট-সেকেন্ডে রূপান্তর করে। এ কাজে তুমি ভাগফল ও ভাগশেষ ব্যবহার করবে।

৬. একটি ত্রিভুজের তিন বাহুর দৈর্ঘ্য a, b, c যোগান (input) নিয়ে এর কোণগুলো নির্ণয় করো। ধরো ত্রিভুজের কোন তিনটি A, B, C। এখান A, B, C যথাক্রমে a, b, c বাহুর বিপরীত কোণ। তুমি হয়তো জানো কোণ $C = \cos^{-1}((a^2+b^2-c^2)/(2ab))$, কোণ $B = \cos^{-1}((c^2+a^2-b^2)/(2ca))$ ও কোণ $A = \cos^{-1}((b^2+c^2-a^2)/(2bc))$। তোমার programএ ত্রিভুজের কোনগুলোকে তুমি ডিগ্রীতে রূপান্তর করে output দিবে।

৭. এমন একটি program (ক্রমলেখ) রচনা করো যেটি দুটো সময় ঘন্টা, মিনিট, সেকেন্ডে নিয়ে সময় দুটিকে যোগ করে। এ কাজে তুমি যোগ, ভাগফল ও ভাগশেষ ব্যবহার করবে।

৮. এমন একটি program রচনা করো যেটি দুটো সমীকরণ $ax+by=c$ ও $dx+ey=f$ এর a, b, c, d, e, f input (যোগান) নিয়ে x ও y এর মান output (ফলন) দেয়।

৯. একটি বাস u আদিবেগ ও a সমত্বরণ নিয়ে যাত্রা শুরু করলো। সময় t সেকেন্ড পরে বাসের গতিবেগ v নির্ণয় করো। t সময় পরে বাসটি অতিক্রান্ত দুরত্ব sও নির্ণয় করো। এ কাজে তুমি গতিবিদ্যার সূত্র $v = u + at$ ও $s = ut + \frac{1}{2}at^2$ ব্যবহার করবে।

১০. নীচের pseudocodeএর (ছদ্ম-সংকেত) জন্য একটি program (ক্রমলেখ) তৈরী করো।

ক) পড়ো (read) x ও y

খ) গণো (compute) $p = x * y$

গ) গণো (compute) $s = x + y$

ঘ) গণো (compute) $t = s^2 + p * (s - x) * (p + y)$

ঙ) লিখো (write) t

Programming Solutions: এবার আমরা programming problemগুলোর সমাধান দেখবো। সমাধানগুলো দেয়া হয়েছে তুমি যদি একান্তই নিজে নিজে করতে পারছোনা বলে মনে হয়, তখন কেবল একটু সাহায্য যাতে পেতে পারো তাই। কাজেই পারতপক্ষে সমাধান দেখবে না।

১. একটি arithmetic seriesএর (সমান্তর ধারা) প্রথম পদ a সাধারণ অন্তর d হলে n-তম পদ কতো? n পদের সমষ্টিই বা কত? এর জন্য সিপিপিতে একটা program (ক্রমলেখ) (program) তৈরী করো যেটি a, d, ও n input (যোগান) নিবে, আর n-তম পদ ও n পদের সমষ্টি output (ফলন) দিবে। এর জন্য তুমি সূত্র ব্যবহার করবে n-তম পদ = $a + (n-1) * d$ আর n পদের সমষ্টি = $n * (2a + (n-1) * d)/2$। প্রদত্ত বিভিন্ন ধারার জন্যে এই সূত্র a আর d বসালে আমরা ওই ধারাগুলোর জন্য সরাসরি সূত্র পেতে পারি।

- $1 + 2 + 3 + \ldots$ ধারাতে $a = 1, d = 1$। সুতরাং n-তম পদ = $a + n - 1$, n পদের সমষ্টি = $n(n+1)/2$। যেমন $n = 10$ হলে 10-তম পদ 10, সমষ্টি 55।

- $2+4+6+\ldots$ ধারাতে $a=2, d=2$। সুতরাং n-তম পদ = $2n$, n পদের সমষ্টি = $n(n+1)$। যেমন $n=10$ হলে 10-তম পদ 20, সমষ্টি 110।
- $1+3+5+\ldots$ ধারাতে $a=1, d=2$। সুতরাং n-তম পদ = $2n-1$, n পদের সমষ্টি = n^2। যেমন $n=10$ হলে 10-তম পদ 19, সমষ্টি 100।

আমরা এখানে কেবল সাধারণ সুত্রের জন্য program (ক্রমলেখ) তৈরী করবো। প্রদত্ত বিশেষ ধারার জন্য তুমি এই program (ক্রমলেখ) দরকার মতো বদলে নিতে পারবে।

ফিরিস্তি ১৪.৬: Arithmetic Series Problem (সমান্তর ধারার সমস্যা)

```
int a, d, n;
cout << "first term? "; cin >> a;
cout << "common diff? "; cin >> d;
cout << "which term?"; cin >> n;

int t = a + (n - 1) * d;  //  n-তম পদ
cout << n << "-th term = " << t << endl;

int s = n * (2*a + (n - 1)*d) / 2; // সমষ্টি
cout << n << " term sum = " << s << endl;
```

যোগান-ফলন (input-output)

```
first term? 1
common diff? 1
which term? 10
10-th term = 10
10 term sum = 55
```

২. নীচের মতো output (ফলন) দেয় এরকম একটি program (ক্রমলেখ) তৈরী করো।

```
x=10 y=5

expr result
x=y+3 x= 8
x=y-2 x= 3
x=y*5 x= 25
x=x/y x= 2
x=x%y x= 0
```

ফিরিস্তি ১৪.৭: Binary Operation Results (দুয়িক অণুক্রিয়ার ফলাফল)

```
int x = 10, y = 5;

cout << "x=" << x << " y=" << y << endl;
cout << endl; //  ফাঁকা সারি
```

```
cout << "expr "<< " result " << endl;
cout << "x=y+3" << " x= " << y+3 << endl;
cout << "x=y-2" << " x= " << y-2 << endl;
cout << "x=y*5" << " x= " << y*5 << endl;
cout << "x=x/y" << " x= " << x/y << endl;
cout << "x=x%y" << " x= " << x%y << endl;
```

৩. এমন একটি program (ক্রমলেখ) রচনা করো যেটি একটি তিন অঙ্কের সংখ্যাকে উল্টো করে যেমন 326 হয়ে যায় 623। এ কাজে তুমি ভাগফল, ভাগশেষ, গুণ, যোগ ও বিয়োগ ব্যবহার করবে। 326 থেকে অঙ্কগুলো আলাদা করে তারপর 623 তৈরী করবে।

```
int given = 326;

int right = given % 10; // ভাগশেষ 6
int left = given / 100; // ভাগফল 3
int middle = given / 10 % 10; //ফল 2

int reverse = left;      // উল্টা = 3
reverse += middle * 10; // উল্টা = 23
reverse += right * 100; // উল্টা = 623
```

৪. একটি ত্রিভুজের তিন বাহুর দৈর্ঘ্য a, b, c input (যোগান) নিয়ে ত্রিভুজটির ক্ষেত্রফল নির্ণয় করো। তুমি হয়তো জানো ত্রিভুজের ক্ষেত্রফল = $\sqrt{s(s-a)(s-b)(s-c)}$ যেখানে s হলো অর্ধ পরিসীমা অর্থাৎ $s = (a+b+c)/2$।

ফিরিস্তি ১৪.৮: Triangle's Area From Sides (ত্রিভুজের বাহু হতে ক্ষেত্রফল)

```
// main বিপাতকের বাইরে
#include <cmath>

// main বিপাতকের ভিতরে
float a, b, c;                 //  বাহুগুলো
cout << "sides a b c: ";  // যোগান যাচনা
cin >> a >> b >> c;         // যোগান নেওয়া

float s = (a + b + c) / 2;  //  অর্ধ পরিসীমা
float k = sqrt(s*(s-a)*(s-b)*(s-c));  // ক্ষেত্রফল

cout << "area = " << k << endl; // ফলন
```

যোগান-ফলন (input-output)

```
sides a b c: 100 60 90
khetrofol = 2666
```

৫. এমন একটি program (ক্রমলেখ) রচনা করো যেটি সেকেন্ড input নিয়ে তাকে ঘন্টা-মিনিট-সেকেন্ডে রূপান্তর করে। এ কাজে তুমি ভাগফল ও ভাগশেষ ব্যবহার করবে।

ফিরিস্তি ১৪.৯: Time in Seconds (সময়কে সেকেন্ডে প্রকাশ)

```
int totalsec = 38185;

int sec = totalsec % 60; // ফল  25
int totalmin = totalsec / 60; // ফল 636

int min = totalmin % 60;  //  ফল 36
int hour = totalmin / 60; // ফল 10
```

৬. একটি ত্রিভুজের তিন বাহুর দৈর্ঘ্য a, b, c input (যোগান) নিয়ে এর কোণগুলো নির্ণয় করো। ধরো ত্রিভুজের কোন তিনটি A, B, C। এখান A, B, C যথাক্রমে a, b, c বাহুর বিপরীত কোণ। তুমি হয়তো জানো কোণ $C = \cos^{-1}((a^2 + b^2 - c^2)/(2ab))$, কোণ $B = \cos^{-1}((c^2+a^2-b^2)/(2ca))$ ও কোণ $A = \cos^{-1}((b^2+c^2-a^2)/(2bc))$। তোমার ক্রমলেখতে ত্রিভুজের কোনগুলোকে তুমি ডিগ্রীতে রূপান্তর করে output দিবে।

আমরা cmath header file থেকে arccosineএর জন্য acos functionটিকে (বিপাতক) ব্যবহার করবো। কিন্তু এটি আমাদের রেডিয়ানে কোণ ফেরত দিবে। রেডিয়ান থেকে ডিগ্রীতে নিতে চাইলে আমাদের 180/pi দিয়ে গুণ করতে হবে। কথা হচ্ছে পাই কেমনে পাবো। আমরা pai একটা constant ঘোষণা করতে পারি যার মান দিয়ে দিব 3.1416 অথবা আরো নিখুত মান পেতে চাইলে acos(−1) থেকেও মান বের করে নিতে পারি।

ফিরিস্তি ১৪.১০: Triangle's Angles From Sides (ত্রিভুজের বাহু হতে কোণ)

```
// main বিপাতকের বাইরে
#include <cmath>

// main বিপাতকের ভিতরে
float a, b, c;                 //  বাহুগুলো
cout << "sides a b c: ";  // যোগান যাচনা
cin >> a >> b >> c;          // যোগান নেওয়া

//  কোণ নির্ণয় রেডিয়ানে
float C = acos((a*a + b*b - c*c)/(2*a*b));
float B = acos((c*c + a*a - b*b)/(2*c*a));
float A = acos((b*b + c*c - a*a)/(2*b*c));

//  ডিগ্রীতে রূপান্তর
float const pai = arccos(-1); // বিকল্প হলো 3.1416
C *= 180/pai; B *= 180/pai; A *= 180/pai;

cout << "angles A B C= "; // ফলন
cout << A << " " << B << " " << C << endl;
```

যোগান-ফলন (input-output)

```
sides a b c: 145 60 90
angles A B C= 149.703 12.049 18.2475
```

৭. এমন একটি program (ক্রমলেখ) রচনা করো যেটি দুটো সময় ঘন্টা, মিনিট, সেকেন্ডে নিয়ে সময় দুটিকে যোগ করে। এ কাজে তুমি যোগ, ভাগফল ও ভাগশেষ ব্যবহার করবে।

ফিরিস্তি ১৪.১১: Adding Two Times (দুটি সময়ের যোগ)

```
int hour1, min1, sec1; // ১ম সময় যোগান নিবে
int hour2, min2, sec2; // ২য় সময় যোগান নিবে

int sec = sec1 + sec2;     //  সেকেন্ড দুটো যোগ
int min = min1 + min2;     // মিনিট দুটো যোগ
int hour = hour1 + hour2; //  ঘন্টা দুটো যোগ

minit += sec / 60;  // মোট সেকেন্ড 60 এর বেশী হলে
sec = sec % 60;     // মিনিট হওয়ার পরে অবশিষ্ট সেকেন্ড
hour += min / 60; // মোট মিনিট 60 এর বেশী হলে
min = min % 60;     // ঘন্টা হওয়ার পরে অবশিষ্ট মিনিট
```

৮. এমন একটি program রচনা করো যেটি দুটো সমীকরণ $ax+by=c$ ও $dx+ey=f$ এর a, b, c, d, e, f input নিয়ে x ও y এর মান output দেয়। এরকম সহ সমীকরণ সমাধানের সূত্র হল $x=(ce-bf)/(ae-bd)$ আর $y=(af-cd)/(ae-bd)$।

ফিরিস্তি ১৪.১২: Simultaneous Equations (সহ সমীকরণ সমাধান)

```
float a, b, c, d, e, f;

cout << "first equation a b c:";
cin >> a >> b >> c;
cout << "second equation e f g:";
cin >> d >> e >> e;

float x = (c*e - b*f)/(a*e - b*d);
float y = (a*f - c*d)/(a*e - b*d);

cout << "x = " << x << " ";
cout << "y = " << y << endl;
```

যোগান-ফলন (input-output)

```
first equation a b c: 2 1 4
second equation e f g: 1 -1 -1
x = 1.33333 y = 1.33333
```

৯. একটি বাস u আদিবেগ ও a সমত্বরণ নিয়ে যাত্রা শুরু করলো। সময় t সেকেন্ড পরে বাসের গতিবেগ v নির্ণয় করো। t সময় পরে বাসটি অতিক্রান্ত দুরত্ব sও নির্ণয় করো। এ কাজে তুমি গতিবিদ্যার সূত্র $v = u + at$ ও $s = ut + \frac{1}{2}at^2$ ব্যবহার করবে।

ফিরিস্তি ১৪.১৩: Solving Motion Equations (গতির সমীকরণ সমাধান)

```
float u, a, t;

cout << "init-speed acceleration time: ";
cin >> u >> a >> t;

float v = u + a * t;
float s = u*t + a * t * t / 2;

cout << "speed: " << v << " ";
cout << "distance: " << s << endl;
```

যোগান-ফলন (input-output)

```
init-speed acceleration time: 2 1 4
speed: 6 distance: 16
```

১০. নীচের pseudocodeএর (ছদ্ম-সংকেত) জন্য একটি program (ক্রমলেখ) তৈরী করো।

ক) পড়ো (read) x ও y

খ) গণো (compute) $p = x * y$

গ) গণো (compute) $s = x + y$

ঘ) গণো (compute) $t = s^2 + p * (s - x) * (p + y)$

ঙ) লিখো (write) t

ফিরিস্তি ১৪.১৪: Program from Pseudocode (ছদ্মসংকেত থেকে ক্রমলেখ তৈরী)

```
int x, y;   // কেবল main বিপাতকের ভিতরের অংশটুকু

cin >> x >> y;  // ধাপ ক

int p = x * y;  // ধাপ খ
int s = x + y;  // ধাপ গ

int t = s*s + p * (s - x) * (p + y);  // ধাপ ঘ

cout << t << endl;  // ধাপ ঙ
```

১৪.১২ Computing Terminologies (গণনা পরিভাষা)

- Positive (ধনাত্মক)
- Negative (ঋণাত্মক)
- Operator (অণুক্রিয়া)
- Operand (উপাদান)
- Unary (একিক)
- Binary (দুয়িক)
- Type casting (উপাত্ত প্রকারান্তর)
- Run-time (চলা-কালীন)
- Execution-time (নির্বাহ-কালীন)
- Commenting (টীকা দেয়া)
- Uncommenting (টীকা তোলা)
- Composition (যোজন)
- Void (নর্থক)
- Separator (পৃথকী)
- Precedence (অগ্রগণ্যতা)
- Associativity (সহযোজ্যতা)

অধ্যায় ১৫

Conditional Programming (শর্তালি পরিগণনা)

আমাদের জীবনটা নাক বরাবর সোজা একটা পথ নয়, প্রতিটা মোড়ে মোড়ে এটা শাখায় শাখায় বিভক্ত। তোমাকে একটা শাখায় যেতে হবে, একসাথে একের বেশী শাখায় যেতে পারবে না। কোনটায় যাবে তার জন্য ভাবতে হবে, তোমার অবস্থা ও লক্ষ্য বিবেচনা করতে হবে। Conditional programming (শর্তালি পরিগণনায়) আমরা শাখায় শাখায় ভাবা শিখবো, আমাদের সামনের গমন পথ বাছাই করা শিখবো, আমরা আমাদের জীবনের সিদ্ধান্ত নেয়া শিখবো।

১৫.১ If Then Else (যদি তাহলে নাহলে)

ধরো গণিত পরীক্ষায় তুমি ৫০ বা বেশী পেলে পাশ করবে আর নাহলে করবে ফেল। আর যদি ৮০ বা বেশী পাও তাহলে তুমি star (তারকা) পাবে। এমন একটি program (ক্রমলেখ) তৈরী করো যেটি তোমার গণিতে পাওয়া নম্বর input (যোগান) নিয়ে তোমাকে পাশ না ফেল output (ফলন) দিবে। আর তুমি যদি star পেয়ে থাকো সেটাও জানাবে, star না পেলে কিছু জানাবে না।

ফিরিস্তি ১৫.১: Pass Fail Star Marks (পাশ-ফেল-তারকা নম্বর নির্ণয়)

```
int number;                         //  চলক ঘোষণা

cout << "number? ";                 //  যোগান যাচনা
cin >> number;                      //  যোগান নেয়া

if (number >= 50)                   //  যদি পাশের নম্বর
  cout << "pass" << endl;           //  পাশ ফলন
else                                //  না হলে
  cout << "fail" << endl;           //  ফেল ফলন

if (number >= 80)                   //  যদি তারকা নম্বর
  cout << "star" << endl;           //  তারকা ফলন
```

উপরের অংশটুকু কোন programএ (ক্রমলেখ) নিয়ে compile (সংকলন) করে চালিয়ে (run) দেখো। যদি 50 এর কম কোন নম্বর input (যোগান) দাও যেমন 45 তাহলে output (ফলন) দেখাবে fail । আর যদি 50 ও 79 এর মাঝের কোন নম্বর input (যোগান) দাও যেমন 65 তাহলে output (ফলন) দেখাবে pass। আর যদি 80 বা বেশী কোন মান input (যোগান) দাও যেমন 85 তাহলে দুই সারি output (ফলন) দেখাবে: প্রথম সারিতে pass আর পরের সারিতে star। নীচের input-output (যোগান-ফলন) এই programটি (ক্রমলেখ) তিন বার চালিয়ে বামে, মাঝে, ও ডানে এই তিনটি ব্যাপার দেখানো হয়েছে।

যোগান-ফলন (input-output)

```
number? 45          number? 65          number? 85
fail                pass                pass
                                        star
```

এবার programটি (ক্রমলেখ) বিশ্লেষণ করি। Variable declaration (চলক ঘোষণা), input-prompt (যোগান যাচনা), ও input (যোগান) নেয়া তো তুমি আগেই শিখেছো। এর পরে খেয়াল করো আমরা লিখেছি if (number >= 50) অর্থাৎ যদি নম্বর 50 বা তার বেশী হয় তাহলে কী করতে হবে সেটা কিন্তু তার পরপরই বলেছি cout << "pass" << endl; অর্থাৎ পাশ output (ফলন) দেখাতে হবে। তারপরের সারি খেয়াল করো else মানে হলো না হলে অর্থাৎ নম্বর যদি 50 বা তার বেশী না হয় মানে 50 এর কম হয়, আমাদের ফেল outputএ দেখাতে হবে যা বলা হয়েছে ঠিক পরের সারিতে cout << "fail" << endl;। যে কোন নম্বর হয় 50 এর কম হবে না হয় বেশী বা সমান হবে, এই দুটো ছাড়া আর ভিন্ন কিছু হতে পারে না, এমনকি ওই দুটো একসাথেও সত্যি হতে পারে না। কাজেই আমাদের programএ (ক্রমলেখ) হয় cout << "pass" << endl; না হয় cout << "fail" << endl; execute (নির্বাহ) হবে, দুটোই একসাথে হতে পারবে না। ঠিক যেন দুটো শাখা তৈরী হয়ে গেলো।

আমরা উপরের program হতে দেখতে পেলাম প্রাপ্ত নম্বরের ওপর ভিত্তি করে ফলাফল পাশ না ফেল দেখাতে হবে অর্থাৎ output (ফলন) দেখানোর ওই দুটো statementএর মধ্যে কোনটা নির্বাহিত হবে সেটা আমরা নম্বর 50 এর কম না বেশী বা সমান এই শর্তটি পরীক্ষা করে বাছাই করতে পারলাম। অনেক programming language (পরিগণনা ভাষা) (number > 50) এর পরে cout << "pass" << endl; এর আগে then লিখতে হয়, কিন্তু c++ এ এটা লিখতে হয় না। এখানে বরং শর্ত number >= 50 এটাকে দুটো () বন্ধনী দিয়ে বন্দী করতে হয়। বন্ধনী দেয়ার ব্যাপারটা মনে রাখবে, কারণ প্রথম প্রথম তুমি এটা নিয়ে প্রায়ই ভুল করে compilation error (সংকলন ত্রুটি) পাবে, আর তোমাকে তখন এটি ঠিক করতে হবে। বন্ধনী দুটো এখানে আশে পাশের শর্তকে পৃথক করে, যা সফল compileএর (সংকলন) জন্যে জরুরী।

উপরের programএ (ক্রমলেখ) খেয়াল করো পাশ ফেল দেখানোর if এর পরে আরো একটা if আছে যেটি দিয়ে আমরা প্রাপ্ত নম্বরটি star (তারকা) কিনা তা দেখাই। এই if এ শর্ত হচ্ছে (number >= 80) অর্থাৎ নম্বর যদি 80 বা এর বেশী হয় তাহলে output (ফলন) দেখাবে star। কিন্তু আর একটু সতর্ক ভাবে খেয়াল করো এই শর্ত মিথ্যা হলে বা পূরণ না হলে কী দেখাবে সেটা কিন্তু নাই। সোজা কথায় এই if এর সাথে কোন else ব্যবহার করা হয় নি। মানে নম্বর যদি 80 এর কম হয় তাহলে স্রেফ কিছুই দেখানোর দরকার নাই। তাহলে আমরা জানলাম if এর শর্ত পূরণ হলে আমাদের কী করতে হবে সেটা লিখতে হবে, কিন্তু শর্ত পূরণ না হলে আমরা দরকার মতো কী করতে হবে সেটা লিখবো, অথবা দরকার না হলে কিছুই লিখবো না।

এবার নীচে programটি খেয়াল করো। এখানে আমরা উপরের পাশ-ফেল দেখানো অংশটিই আবার দেখিয়েছি, তবে একটু ভিন্ন ভাবে। ভিন্নতাটা হলো উপরে যেমন else এর পরে সরাসরি cout << "fail" << endl; লিখেছিলাম, এখানে তা না করে else এর পরে if (number

< 50) লিখেছি। তোমাদের কাছে মনে হতে পারে, এটা তো সুন্দর, বুঝতে সুবিধা কারণ ঠিক যেন মানুষের ভাষায় আমরা যে ভাবে বলি যেমন যদি নম্বর 50 বা বেশী হয় output দেখাও পাশ নাহলে যদি নম্বর 50 এর কম হয় output দেখাও ফেল ঠিক তার মতো। কথা সত্য আমাদের বুঝা সুবিধা হয় এ ভাবে। কিন্তু আমরা এভাবে লিখবো না, কারণ else এর পরে ওই if (number < 50) লিখা আসলে অদরকারী আর সে কারণে তোমার program (ক্রমলেখ) খামোকা slow (শ্লথ) হয়ে যাবে। ওই if (number < 50) লেখাটা অদরকারী কারণটা আগেই খানিকটা জেনেছি তবুও আরেকবার বলি else এর শাখায় আসা মানে হলো number >= 50 এই শর্তটি মিথ্যা হয়েছে। আর এই শর্তটি মিথ্যা হওয়া মানে number < 50 শর্তটি অবশ্যই সত্য। কাজেই এটি আবার আর একটি if লাগিয়ে পরীক্ষা করার কোন প্রয়োজন নাই।

```
if (number >= 50)                    //  যদি পাশের নম্বর
  cout << "pass" << endl;            //  পাশ ফলন
else if (number < 50)                //  না হলে
  cout << "fail" << endl;            //  ফেল ফলন
```

তুমি যদি একান্তই মানুষের বুঝার সুবিধার্থে ওই if (number < 50) টা লিখতে চাও, সে-টা commentএর (টীকা) ভিতরে লিখতে পারো। নীচে যেমন লিখে দেখালাম। এতে তোমার programও (ক্রমলেখ) ধীর গতির হলো না, আবার তোমার পক্ষে program পড়তেও সহজ হয়ে গেলো। আমরা এ রকমই প্রায়ই করে থাকি। তবে অনেক ক্ষেত্রে else এর পরে ওইরকম একটা if দেওয়া অবশ্যম্ভাবীও হয়ে যায়, এটা আমরা পরের একটা পাঠেই বিস্তারিত দেখবো।

```
if (number >= 50)                    //  যদি পাশের নম্বর
  cout << "pass" << endl;            //  পাশ ফলন
else //if (number < 50)              //  না হলে
  cout << "fail" << endl;            //  ফেল ফলন
```

এই আলোচনার শেষ করি আরেকটা ব্যাপার দিয়ে, সেটা হলো indentation (ছাড়ন দে-য়া)। Indentation নিয়ে আগে একবার আমরা আলোচনা করেছিলাম। খেয়াল করো আমরা if (number >=50) এর পরে এই শর্ত সত্য হলে যে cout << "pass" << endl; টা execute (নির্বাহ) করতে হবে সেটা পরের সারিতে একটু ভিতরে থেকে লিখতে শুরু করেছি। এটা করলে computerএর (গণনি) জন্য কিন্তু কোন লাভ বা ক্ষতি নেই, কিন্তু আমরা সহজে চোখে দেখেই কেমন বুঝতে পারি যে ওই cout এর সারিটি আসলে তার আগের সারির if এর সাথের শর্ত সত্য হওয়ার ওপরে নির্ভরশীল। তারপর দেখো পরের সারিতে থাকা else আবার একটু ভিতর থেকে শুরু না হয়ে if বরাবরই শুরু হয়েছে। এটা দিয়ে আমরা বুঝাতে চাই এই else টা আসলে ওই if এর শর্তটা মিথ্যা হলে প্রযোজ্য হবে। লম্বা programএ (ক্রমলেখ) যখন অনেক if আর অনেক else থাকবে তখন কোন else কোন if এর সাথে তা match (মিলানো) আমাদের পক্ষে চোখে দেখে কঠিন হয়ে যেতে পারে। ওই মিলানোর সুবিধার্থে if আর তার সাথের else এক বরাবর লিখা হয়। সবশেষে খেয়াল করো else এর পরের সারির cout আবার একটু ভিতর থেকে লেখা, কারণ এটা execute (নির্বাহ) হবে কিনা তা নির্ভর করে else এর ওপরে। একটু ভিতর থেকে লেখা শুরু করে সেইটাই বুঝানো হয়।

১৫.২ Relational Operators (অন্বয়ী অণুক্রিয়া)

Relational (অন্বয়ী) operators কী? সিপিপিতে ছয়টি relational operators >= > = = != < <= রয়েছে দুটো রাশির তুলনা করার জন্য। এই operatorগুলো আলোচনা করো।

```
cout << "x y x>=y x>y x==y x!=y x<y x<=y" << endl;

cout << 3 << " " << 4 << "    ";
cout << (3>=4) << "    " << (3>4) << "    ";
cout << (3==4) << "     " << (3!=4) << "     ";
cout << (3<4) << "    " << (3<=4) << endl;

cout << 4 << " " << 4 << "    ";
cout << (4>=4) << "    " << (4>4) << "    ";
cout << (4==4) << "     " << (4!=4) << "     ";
cout << (4<4) << "    " << (4<=4) << endl;

cout << 4 << " " << 3 << "    ";
cout << (4>=3) << "    " << (4>3) << "    ";
cout << (4==3) << "     " << (4!=3) << "     ";
cout << (4<3) << "    " << (4<=3) << endl;
```

উপরের programএ (ক্রমলেখ) প্রথমে আমরা দুটো অসমান সংখ্যার তুলনা করেছি যেখানে আগেরটি পরেটি থেকে ছোট। তারপরে আমরা দুটো সমান সংখ্যার তুলনা করেছি। সবশেষে আবারো দুটো অসমান সংখ্যার তুলনা করেছি কিন্তু এখানে আগেরটি বড়, পরেরটি ছোট। উক্ত programএর প্রেক্ষিতে output (ফলন) কী হবে তা নীচে দেখানো হয়েছে।

ফলন (output)

```
x y x>=y x>y x==y x!=y x<y x<=y
3 4   0   0   0     1     1   1
4 4   1   0   1     0     0   1
4 3   1   1   0     1     0   0
```

এখানে ছয়টি relational operators (অন্বয়ী অণুক্রিয়া) ব্যবহার করা হয়েছে। এগুলো হল >= বড় বা বেশী (greater or equal), > বড় (greater), == সমান (equal), != অসমান (unequal), < ছোট (smaller), <= ছোট বা কম (smaller or equal)। একটা বিষয় খেয়াল করো এখানে == সমান চিহ্ন কিন্তু দুটো = চিহ্ন দিয়ে, একটা দিয়ে নয়। Assignment (আরোপণ) হলো একটা = দিয়ে। Program (ক্রমলেখ) লিখতে গেলে আমাদের প্রায়শই এই ভুলটি হয়ে যায়, আর program ঠিকঠাক কাজ করে না। তোমরা এদিকে বিশেষ নজর রাখবে সব সময়।

যাইহোক উপরের outputএ (ফলন) দেখো, যখনই কোন তুলনার ফলাফল সত্য হয়েছে, outputএ (ফলন) সেটি এসেছে 1 হিসাবে আর যেখানে তুলনার ফলাফল মিথ্যা তখনই এসেছে 0। আসলে এই relational operators (অন্বয়ী অণুক্রিয়া) Boolean (বুলক) নামের এক প্রকারের মান ফলাফল হিসাবে দেয়। Boolean যেটাকে সিপিপিতে bool হিসাবে লেখা হয় সেটা হলো এক রকমের data type (উপাত্ত প্রকরণ)। Boolean মান কেবল সত্য আর মিথ্যা হতে পারে। সিপিপিতে মান দুটো হল মিথ্যা 0 ও সত্য 1। Programএর (ক্রমলেখ) ভিতরে অবশ্য মিথ্যা আর সত্যকে 0 আর 1 দিয়ে না বুঝিয়ে আমরা চাইলে স্পষ্টাকারে false ও true দিয়ে বুঝাতে পারি, কিন্তু outputএ (ফলন) দেখালে ওটা 0 আর 1 হিসাবে দেখানো হয়ে যাবে।

উপরের programএ (ক্রমলেখ) যদিও কেবল পূর্ণক (integer) ব্যবহার করা হয়েছে, তবে relational operators (অন্বয়ী অণুক্রিয়া) আসলে ভগ্নকের (fractioner) সাথেও একই ভাবে

কাজ করে। চাইলে একটা ভগ্নক ও একটা পূর্ণকও ওই operatorগুলোতে এক সাথে ব্যবহার করা যায়, আর সেক্ষেত্রে পূর্ণকটি (int) প্রথমে ভগ্নকে (float) type cast (প্রকারান্তরিত) হয়ে যাবে, তারপর তুলনাটি হবে দুটো ভগ্নক (float) এর মধ্যে। ফলাফল অবশ্যই হবে একটি Boolean (bool) অর্থাৎ 0 বা 1। Relational operators (অন্বয়ী অণুক্রিয়া) Boolean (bool) এর ওপরও কাজ করে। সেক্ষেত্রে false আর true কে স্রেফ 0 আর 1 ধরে পূর্ণক হিসাবে বিবেচনা করলে তুলনার যে ফলাফল আসার কথা তাই আসবে। উপরের programএ তুমি 3 ও 4 এর বদলে নানা রকম Boolean (bool) বা পূর্ণক (int) বা ভগ্নক (float) মান দুটো operandই (উপাদান) একরকম বা দুটো দুইরকম করে বসিয়ে ফলাফলগুলো পর্যবেক্ষণ করতে পারো।

১৫.৩ If-Else Ladder (যদি-নাহলে মই)

If-Else Ladder (যদি-নাহলে মই) কী? If-Else Ladder ব্যবহার করে কোন প্রদত্ত বছর leap year (অধিবর্ষ) কিনা তা নির্ণয়ের জন্য একটি program (ক্রমলেখ) তৈরী করো।

প্রথমে আমরা দেখি একটা বছর কখন leap year (অধিবর্ষ) হয়। একটি প্রদত্ত বছর যদি ৪০০ দ্বারা বিভাজ্য হয় তাহলে এটি leap year, যেমন ১৬০০ ও ২০০০। তা নাহলে অর্থাৎ বছরটি যদি ৪০০ দিয়ে বিভাজ্য না হয় কিন্তু এটি যদি ১০০ দিয়ে বিভাজ্য হয় তাহলে এটি leap year নয়, যেমন ১৮০০ ও ১৯০০। তাও নাহলে অর্থাৎ বছরটি ১০০ দ্বারাও বিভাজ্য নয় কিন্তু যদি ৪ দ্বারা বিভাজ্য তাহলে এটি leap year, যেমন ২০১২ বা ২০১৬। তাও নাহলে অর্থাৎ বছরটি যদি ৪ দ্বারা বিভাজ্য না হয় তাহলে এটি leap year নয় অর্থাৎ সাধারণ বর্ষ যেমন ২০১৪ বা ২০১৫। এই কথাগুলোকে সংক্ষেপে লিখলে দাঁড়ায় "যদি ৪০০ দ্বারা বিভাজ্য হয় তাহলে leap year, নাহলে যদি ১০০ দ্বারা বিভাজ্য হয় তাহলে leap year নয়, নাহলে যদি ৪ দ্বারা বিভাজ্য হয় তাহলে leap year, নাহলে leap year নয়।" আমরা এটিকেই programএ লিখে ফেলবো।

ফিরিস্তি ১৫.২: Leap Year Determination (অধিবর্ষ নির্ণয়)

```
int year;
cout << "year is ? ";
cin >> year;

if (year % 400 == 0)  // ৪০০ দিয়ে বিভাজ্য
  cout << "leap year yes" << endl;
else if (year % 100 == 0) // ১০০ দিয়ে বিভাজ্য
  cout << "leap year no" << endl;
else if (year % 4 == 0) // ৪ দিয়ে বিভাজ্য
  cout << "leap year yes" << endl;
else    //  if (year % 4 != 0) ৪ দিয়ে বিভাজ্য নয়
  cout << "leap year no" << endl;

cout << "how lovely!" << endl;
```

এবার আমাদের programএর (ক্রমলেখ) দিকে তাকাই। উপরে সংক্ষেপে ঠিক যে ভাবে leap year নির্ণয় করার নিয়ম বর্ণনা করেছি, আমাদের programএ আমরা যেন তাই লিখেছি। মিলিয়ে নাও। Arithmetical operatorগুলোর (পাটিগণিতীয় অণুক্রিয়া) পাঠ থেকে মনে করো দেখো % operator আমাদের ভাগশেষ ফলাফল দেয়। তো ভাগশেষ যদি শুন্য হয় তাহলে আমরা

বিভাজ্যতা বুঝতে পারবো, আর ভাগশেষ শুন্য না হলে অবিভাজ্যতা। আমরা প্রথমে ৪০০ দিয়ে বিভাজ্যতা পরীক্ষা করেছি, না হলে তারপর ১০০ দিয়ে বিভাজ্যতা, তাও নাহলে তারপর ৪ দিয়ে বিভাজ্যতা পরীক্ষা করেছি। কেমন হুবহু একই রকম করে programটি লেখা গেছে!

খেয়াল করো বিভাজ্য হওয়া year %400 == 0 আর অবিভাজ্য হওয়া year %400 != 0 এই দুটোতো বিপরীত শর্ত। Programএ (ক্রমলেখ) প্রথম শর্ত ব্যবহার করলে ওই শর্ত সত্য (অথবা মিথ্যা) হলে যা করতে হবে, একই কাজ দ্বিতীয় শর্ত ব্যবহার করলে সেই শর্ত মিথ্যা (অথবা সত্য) হলে করতে হবে। প্রশ্ন হলো পরস্পর বিপরীত এই দুটোর মধ্যে কোন শর্তটা ব্যবহার করা চিন্তার জন্য সহজ। তাছাড়া 400 দিয়ে বিভাজ্যতাই বা আগে কেন করবো, 4 বা 100 দিয়ে বিভাজ্য-তাও তো আগে করতে পারি? এসবের উত্তর হল ব্যতিক্রম ও বেশী ব্যতিক্রমগুলো তাহলেতে রাখো, আর বাঁদবাকী কম ব্যতিক্রমগুলো সব রাখো নাহলেতে, তাতে চিন্তা করা সহজ হয়ে যায়, ক্রমলেখ (program) তৈরীও সহজ হয়। যেমন ৪০০ দিয়ে বিভাজ্য হলে অধিবর্ষ, এটা অনেক বেশী ব্য-তিক্রম, তুলনামুলক অল্প সংখ্যক বছর ৪০০ দিয়ে বিভাজ্য হবে। ১০০ দিয়ে বিভাজ্য হওয়া আর একটু কম ব্যতিক্রম মানে তুলনামুলক ভাবে অনেক বছরই ১০০ দিয়ে বিভাজ্য। ৪ দিয়ে বিভাজ্য হওয়া আরো কম ব্যতিক্রম মানে তুলনামুলক ভাবে অনেক বেশী সংখ্যক বছর ৪ দিয়ে বিভাজ্য। আর ৪ দিয়ে বিভাজ্য না হওয়া মোটামুটি সাধারণ ঘটনা ধরা যায়, বাদবাঁকী সব বছরই ৪ দিয়ে অবিভাজ্য। খেয়াল করো program (ক্রমলেখ) সেভাবেই ব্যতিক্রম মাথায় রেখেই লেখা হয়েছে। সব চেয়ে বেশী ব্যতিক্রমী ব্যাপার সবচেয়ে আগে, সবচেয়ে কম ব্যতিক্রম সবচেয়ে পরে।

আমাদের programএ indentation (ছাড়ন) দেয়ার ব্যাপারটা একটু খেয়াল করো। যদিও আমরা জানি indentation দেয়া না দেওয়া অথবা ফাঁকা দিয়ে দিয়ে লেখা বা না লেখাতে আমা-দের programএর (ক্রমলেখ) ফলাফলে কোন পরিবর্তন হয় না। আমরা কেবল মানুষের বোঝার সুবিধার্থে ওগুলো করি। তারপরও খেয়াল করো আমাদের বুঝার সুবিধার্থে আমরা প্রথমের if, তা-রপরের else if গুলো, সবশেষের else আর তাদের শর্ত সত্য হলে যা করতে হবে সব মিলিয়ে কী সুন্দর একটা pattern (ধাঁচ) তৈরী করেছি। এই patternটি একটি মইয়ের মতো কারণ আমা-দের প্রথম if দিয়ে শুরু করে শর্ত পরীক্ষা করতে করতে নীচের দিকে নামতে হবে। আর যে কোন একটি শর্ত পূরণ হলেই তার জন্য যে কাজটি করতে হবে পাশের দিকে গিয়ে সেটি করলেই পুরো patternটির কাজই আসলে শেষ হয়ে যাবে। মানে একটা শর্ত সত্য হলে নীচের দিকের আরো কোন শর্ত আর পরীক্ষা করা হবে না, পুরো patternএর কাজ শেষ হয়ে যাবে। আর ঠিক এর পরে যে statement (বিবৃতি) execute (নির্বাহ) হবে সেটি হলো এই পুরো patternএর বাইরে থাকা কোন statement। যেমন উপরের programএ লক্ষ্য করো cout << "how lovely! " << endl; হলো পুরো patternএর বাইরে, সুতরাং if-lese ladder থেকে বের হয়েই ওইটি execute হতে শুরু করবে।

১৫.৪ Nested If-Else (অন্তান্তি যদি-নাহলে)

Nested if-else (অন্তান্তি যদি-নাহলে) কী? Nested if-else ব্যবহার করে কোন প্রদত্ত বছর leap year (অধিবর্ষ) কিনা তা নির্ণয়ের জন্য একটি program (ক্রমলেখ) তৈরী করো।

একটা বছর কখন leap year হয়, সেটা আগেই জেনেছি, তবুও আরেকবার: বছরটি যদি ৪০০ দ্বারা বিভাজ্য হয় তাহলে এটি leap year, যেমন ১৬০০ ও ২০০০। তা নাহলে অর্থাৎ বছরটি যদি ৪০০ দিয়ে বিভাজ্য না হয় কিন্তু এটি যদি ১০০ দিয়ে বিভাজ্য হয় তাহলে এটি leap year নয়, যেমন ১৮০০ ও ১৯০০। তাও নাহলে অর্থাৎ বছরটি ১০০ দ্বারাও বিভাজ্য নয় কিন্তু যদি ৪ দ্বারা বিভাজ্য তাহলে এটি leap year, যেমন ২০১২ বা ২০১৬। তাও নাহলে অর্থাৎ বছরটি যদি ৪ দ্বারা বিভাজ্য না হয় তাহলে এটি leap year নয় অর্থাৎ সাধারণ বর্ষ যেমন ২০১৪ বা ২০১৫। আগের পাঠে দেখানো আমাদের নীচের programটি সে ভাবেই if-else ladder দিয়ে লেখা।

```
if (year % 400 == 0)  // ৪০০ দিয়ে বিভাজ্য
  cout << "leap year yes" << endl;
else if (year % 100 == 0) // ১০০ দিয়ে বিভাজ্য
  cout << "leap year no" << endl;
else if (year % 4 == 0) // ৪ দিয়ে বিভাজ্য
  cout << "leap year yes" << endl;
else    //  if (year % 4 != 0) ৪ দিয়ে বিভাজ্য নয়
  cout << "leap year no" << endl;
```

উপরের programএ (ক্রমলেখ) দেখো দ্বিতীয় if statementএ (বিবৃতি) ১০০ দিয়ে বিভাজ্যতা পরীক্ষা করা হয়েছে কিন্তু ৪০০ দিয়ে অবিভাজ্য হওয়ার পরে। তো আমরা যদি year % 400 == 0 লিখে বিভাজ্যতা পরীক্ষা না করে তার উল্টোটা year % 400 != 0 লিখে অবিভাজ্যতা পরীক্ষা করতাম তাহলে programটি কেমন হতো? তাহলে সালটি যে leap year সেটা দেখানোর cout চলে যেতো else এর সাথে। নীচের programএর সাথে মিলিয়ে নাও।

```
if (year % 400 != 0)  // ৪০০ দিয়ে অবিভাজ্য
  if (year % 100 == 0)  // ১০০ দিয়ে বিভাজ্য
    cout << "leap year no" << endl;
  else if (year % 4 == 0) // ৪ দিয়ে বিভাজ্য
    cout << "leap year yes" << endl;
  else    //  if (year % 4 != 0) ৪ দিয়ে বিভাজ্য নয়
    cout << "leap year no" << endl;
else      // if (year % 400 == 0) ৪০০ দিয়ে বিভাজ্য
  cout << "leap year yes" << endl;
```

তুমি এবার জিজ্ঞেস করতে পারো, আচ্ছা আমি কি একই ভাবে ১০০ বা ৪ দিয়ে বিভাজ্য হওয়ার if গুলোকেও ১০০ বা ৪ দিয়ে অবিভাজ্যতার if দিয়ে লিখতে পারতাম? হ্যাঁ অবশ্যই। নীচের program (ক্রমলেখ) খেয়াল করো। আমরা প্রতিটি if এর শর্তই বদলে এখন অবিভাজ্যতার শর্ত দিয়ে দিয়েছি। If-else ladderএ আমরা if এর সাথে থাকা শর্ত মিথ্যা হলে তার else এর পরপরই একটা if দেখতে পেতাম। এখানে দেখো উল্টোটা, if এর শর্ত সত্য হলে বরং তার পরপরই আরেকটা if দেখা যাচ্ছে। এটাকে আমরা বলবো nested if-else (অন্তান্তি যদি-নাহলে) অর্থাৎ একটা if-elseএর ভিতরে আরেকটা if-else, তার ভিতরে আরেকটা!

```
if (year % 400 != 0)  // ৪০০ দিয়ে অবিভাজ্য
  if (year % 100 != 0)  // ১০০ দিয়ে অবিভাজ্য
    if (year % 4 != 0)  // ৪ দিয়ে অবিভাজ্য
      cout << "leap year no" << endl;
    else  //  if (year % 4 == 0) ৪ দিয়ে বিভাজ্য
      cout << "leap year yes" << endl;
  else    //  if (year % 100 == 0) ১০০ দিয়ে বিভাজ্য
    cout << "leap year no" << endl;
else      //  if (year % 400 == 0) ৪০০ দিয়ে বিভাজ্য
  cout << "leap year yes" << endl;
```

Nested if-else (অন্তান্তি যদি না হলে) লেখায় indentation (ছাড়ন) দেয়ার ব্যাপারটি খেয়াল করো। সবচেয়ে ভিতরের else সবচেয়ে ভিতরের if এর সাথে। মাঝের else মাঝের if এর সাথে আর সবচেয়ে বাইরের else বাইরের if এর সাথে। Commentএর (টীকা) অংশগুলো দেখে মিলিয়ে নাও। Program (ক্রমলেখ) লিখতে indentation (ছাড়ন) দেয়া যে কতটা গুরুত্বপূর্ণ সেটা এখান থেকে তোমার বেশ বুঝতে পারার কথা। Indentation (ছাড়ন) না থাকলে program (ক্রমলেখ) বুঝা আমাদের জন্য দুরহ হবে।

উপরের আলোচনায় একটা জিনিস আমরা দেখেছি: if-else ladder (যদি-নাহলে মই) আর nested if-else (অন্তান্তি যদি-নাহলে) খানিকটা পরস্পরের বিপরীত। তুমি কিন্তু চাইলে এ দুটোর মিশ্রণ ঘটাতে পারো মানে পুরোটাই মই না করে বা পুরোটাই nested না করে দুইরকমটাই ব্যবহার করলে! যেমন ধরো আমরা যদি প্রথমে ১০০ দিয়ে বিভাজ্যতা পরীক্ষা করি। তাহলে শর্ত সত্য হলেই আমরা অধিবর্ষ বলতে পারি না। আমাদের দেখতে হবে ৪০০ দিয়ে বিভাজ্য কিনা। আর ১০০ দিয়ে বিভাজ্য না হলে আমাদের দেখতে হবে ৪ দিয়ে বিভাজ্য কিনা। তো সেই অনুসারে নীচের program (ক্রমলেখ) খেয়াল করো এখানে nested if-else (অন্তান্তি যদি-নাহলে) আছে আবার if-else ladderও (যদি-নাহলে মই) আছে। Indentation (ছাড়ন) দেখে চিনতে পারছো? তুমি কিন্তু আরো নানান ভাবে leap year (অধিবর্ষ) নির্ণয় নিয়ে আর if-else নিয়ে খেলতে পারো। কোন শর্ত আগে, কোনটা পরে, কোনটা মাঝে, কোনটাকে nested করবে, কোনটাকে ladderএ দিবে, চেষ্টা করে দেখবে, মজাও পাবে, বিষয়গুলো শিখবেও!

```
if (year % 100 == 0)  // ১০০ দিয়ে বিভাজ্য
  if (year % 400 == 0)  // ৪০০ দিয়ে বিভাজ্য
      cout << "leap year yes" << endl;
  else                  //  ৪০০ দিয়ে অবিভাজ্য
      cout << "leap year no" << endl;
else if (year % 4 == 0) // ৪ দিয়ে বিভাজ্য
    cout << "leap year yes" << endl;
else                    //   ৪ দিয়ে অবিভাজ্য
    cout << "leap year no" << endl;
```

১৫.৫ Dangling Else (ঝুলন্ত নাহলে)

ডাক বিভাগ সারাদেশকে অনেক অঞ্চলে ভাগ করে প্রতিটি অঞ্চলের একটা করে ক্রমিক নম্বর দিয়ে দেয়। ঢাকার অঞ্চলগুলোর ক্রমিক নম্বর ১০০ পর্যন্ত, তার মধ্যে ১৩ দিয়ে বিভাজ্য নম্বরগুলো হলো সংরক্ষিত অঞ্চল যেমন ১৩, ২৬, ৩৯, ৫২, ৬৫, ৭৮, ৯১। ঢাকার ভিতর থেকে ডাকে চিঠি পাঠানোর খরচ সারাদেশের যে কোন জায়গায় হলে ৪ টাকা। কিন্তু গন্তব্য ঠিকানা ঢাকার ভিতরেই হলে খরচ ২টাকা, আর ঢাকার ভিতরেই কিন্তু সংরক্ষিত অঞ্চলে হলে খরচ ৩ টাকা। তুমি বেশীর ভাগ সময় ঢাকার ভিতরেই কোথাও না কোথাও চিঠি পাঠাও, তবে মাঝে মাঝে অন্যত্রও পাঠাও। তো তোমাকে একটি program (ক্রমলেখ) লিখতে হবে যেটি তোমার চিঠির গন্তব্য কত নম্বর অঞ্চলে input (যোগান) নিয়ে তোমাকে চিঠি পাঠানোর খরচ outputএ (ফলন) দেখাবে। তোমার programএ (ক্রমলেখ) তুমি অবশ্যই if-else (যদি নাহলে) ব্যবহার করবে কিন্তু তাতে যেন কোন ভাবেই dangling else (ঝুলন্ত নাহলে) দিয়ে ভুল না করে বসো, সেটা খেয়াল রাখবে।

এই program (ক্রমলেখ) লেখা তো খুব সহজ। If-else ladder (যদি-নাহলে মই) ব্যবহার করে তুমি সহজেই লিখে ফেলতে পারো। প্রথমে পরীক্ষা করবে অঞ্চল ১০০ এর চেয়ে বড় কিনা। ১০০ এর বড় হলে খরচ ৪ টাকা, কারণ অঞ্চলটি ঢাকার বাইরে। আর নাহলে মানে অঞ্চলটি ঢাকার

ভিতরে হলে এবার পরীক্ষা করে দেখবে ১৩ দিয়ে বিভাজ্য কিনা। ১৩ দিয়ে বিভাজ্য হওয়া মানে সংরক্ষিত অঞ্চল সুতরাং খরচ ৩ টাকা, আর ১৩ দিয়ে বিভাজ্য না হলে মানে অসংরক্ষিত এলাকা হলে খরচ ২ টাকা। নীচের programএর (ক্রমলেখ) সাথে মিলিয়ে দেখো।

```
int zone; // অঞ্চল

cout << "which zone? ";
cin >> zone;

if (zone > 100)             // ঢাকার বাইরে
  cost = 4;
else if (zone % 13 == 0)    // সংরক্ষিত অঞ্চল
  zone = 3;
else                          // অসংরক্ষিত অঞ্চল
  zone = 2;
```

এই programটি আরো নানান ভাবেই লেখা সম্ভব তুমি সেগুলো নিজে নিজে চেষ্টা করবে। তবে আমরা তো কেবল এটি সমাধানই শিখছি না, আমরা শিখবো dangling else (ঝুলন্ত নাহলে) Patternটি (ধাঁচ) কেমন সেটি। তো আমাদের সমস্যার বিবরণে খেয়াল করো একটা কথা আছে তুমি বেশীর ভাগ চিঠিই পাঠাও ঢাকায়। আর সেখানে অসংরক্ষিত এলাকার সংখ্যায় বেশী। এ থেকে আমরা ধরে নিতে পারি যে খরচ বেশীর ভাগ সময়ই ২ টাকা। কাজেই আমরা cost variableটির মান শুরুতেই ২টাকা initial assignment (আদি আরোপণ) করে ফেলতে পারি। তারপর শর্ত পরীক্ষা করে যদি দেখি ঢাকার ভিতরে আর সংরক্ষিত তাহলে খরচ করে দিবো ৩ টাকা আর ঢাকার বাইরে হলে করে দেবো ৪ টাকা। নীচের programটি দেখো। আমরা সে রকমটি করার চেষ্টা করেছি।

```
int cost = 2; //  ঢাকার ভিতরে অসংরক্ষিত

if (zone <= 100)  //  ঢাকার ভিতরে
  if (zone % 13 == 0) //  সংরক্ষিত
    cost = 3;
else              // দেখতে মনে হয় ঢাকার বাইরে
  cost = 4;
```

উপরের অংশটুকু ব্যবহার করে কোন program (ক্রমলেখ) তৈরী করে চালালে সেটি সঠিক cost জানাবে না। ঢাকার বাইরের অঞ্চলগুলোর জন্য যেখানে খরচ ৪টাকা হওয়ার কথা, তা না হয়ে বরং ২টাকাই থাকবে। আর ঢাকার ভিতরের অসংরক্ষিত এলাকার জন্য যেখানে খরচ হওয়ার কথা ২টাকা তা না হয়ে খরচ ৪টাকা হবে। Program (ক্রমলেখ) চোখে দেখে তো মনে হচ্ছে সব ঠিক আছে, তবে কেন এই বিপত্তি! আসলে বিপত্তি বাঁধিয়েছে else অংশটি। আমরা যেভাবে indentation (ছাড়ন) দিয়ে লিখেছি তাতে মনে হচ্ছে else অংশটুকু প্রথম if সাথের অর্থাৎ zone <= 100 মিথ্যা হওয়ার সাথে জড়িত। কিন্তু আসলে তা নয়। প্রতিটি else তার পূর্বের নিকটতম সঙ্গীহীন if এর সাথে জড়িত। তার মানে এইখানে else টি পরের if এর সাথে জড়িত। অর্থাৎ zone যদি 13 দিয়ে বিভাজ্য না হয় তার সাথে জড়িত।

Nested if-else (অন্তান্তি যদি-নাহলে) আলোচনায় আমরা দেখেছিলাম সবচেয়ে ভিতরের else ঠিক সবচেয়ে ভিতরের if এর সাথে, মাঝের else ঠিক মাঝের if এর সাথে, আর বা-

ইরের else ঠিক বাইরের if এর সাথে। আসলে কোন else কোন if এর সাথে যাবে এখানে indentationএর (ছাড়ন) কোন প্রভাবই নেই। যে else এর জন্য if মিলানো দরকার সেখান থেকে উপরের দিকে যেতে থাকলে প্রথম যে if পাওয়া যাবে যার সাথে কোন ইত্যমধ্যে else দেওয়া হয় নাই, সেই if-ই হলো আমাদের ওই else এর সাথের if। Indentation (ছাড়ন) কেবল আমাদের চোখের দেখার জন্য, computerএর (গণনি) কাছে এর কোন অর্থ নেই। তাহলে সঠিকভাবে indentation (ছাড়ন) দিয়ে লিখলে আমাদের উপরের program (ক্রমলেখ) আসলে নীচের মতো হবে। সুতরাং বুঝতেই পারছো উল্টাপাল্টা indentaiton (ছাড়ন) দেখে তুমি ভাববে তোমার program এরকম কাজ করবে, কিন্তু আসলে সেটা কাজ করবে ভিন্ন রকম। আর ভুলটা কোথায় তা বের করতে তুমি গলদঘর্ম হয়ে যাবে!

```
int cost = 2; //  ঢাকার ভিতরে অসংরক্ষিত

if (zone <= 100)  //  ঢাকার ভিতরে
  if (zone % 13 == 0) // সংরক্ষিত
    cost = 3;
  else                // অসংরক্ষিত
    cost = 4;
```

এরকমের সমস্যা যেখানে else কার সাথে তা বুঝতে আমাদের ঝামেলা লাগে, সেই সমস্যাকে বলা হয় dangling else (ঝুলন্ত নাহলে)। উপরের সঠিক indentation (ছাড়ন) দিয়ে আমরা বুঝতে পারলাম সমস্যা কোথায় কিন্তু সমাধান কিন্তু আমরা এখনো জানিনা, else কিন্তু আসলেই আমরা বাইরের if এর zone <= 100 মিথ্যা হলে কী হবে তার জন্য লিখতে চাই। উপায় কী? উপায় খুবই সহজ। ভিতরের if এর জন্য একটা else লাগিয়ে দাও, আর সেই else এর জন্য তো আমাদের কিছু করার নাই। কারণ ওই else এর জন্য খরচ ২টাকা সেটা তো আমরা আগেই initial assignmentএর (আদি আরোপণ) সময় দিয়ে এসেছি। কিছু করার নাই বুঝাতে আমরা সাধারণত empty statement (শুন্য বিবৃতি) ব্যবহার করি। আর কোন কিছু ছাড়া কেবল semicolor (দির্তি) ; দিয়ে আমরা empty statement (শুন্য বিবৃতি) বুঝাই। এবার তাহলে পরিস্কার হয়ে গেলো কোন else কোন if এর জন্যে।

```
int cost = 2; //  ঢাকার ভিতরে অসংরক্ষিত

if (zone <= 100)  // ঢাকার ভিতরে
  if (zone % 13 == 0) //  সংরক্ষিত
    cost = 3;
  else              //   অসংরক্ষিত
    ;               //  শুন্য বিবৃতি
else                //  ঢাকার বাইরে
  cost = 4;
```

১৫.৬ Compound Statement (যৌগিক বিবৃতি)

Compound statement (যৌগিক বিবৃতি) বলতে কী বুঝো? Compound statement ও if-else (যদি-নাহলে) ব্যবহার করে একটি program (ক্রমলেখ) লিখো যেটি প্রথমে দুটি সংখ্যা

input নিবে। তারপর প্রথম সংখ্যাটি 0 হলে পরের সংখ্যাটিকে ব্যাসার্ধ ধরে ক্ষেত্রফল ও পরিধি output দিবে। আর প্রথম সংখ্যাটি 1 হলে দ্বিতীয় সংখ্যাটিকে বর্গের এক বাহুর দৈর্ঘ্য ধরে বর্গটির ক্ষেত্রফল ও পরিসীমা output দিবে। প্রথম সংখ্যাটি 0 বা 1 ছাড়া অন্যকিছু হলে দেখাবে "unsupported shape" অর্থাৎ এর জন্য আমাদের program কাজ করবে না।

```
float number1, number2;          //  চলক ঘোষণা
cout << "two numbers are? ";     //  যোগান যাচনা
cin >> number1 >> number2;       //  যোগান নেওয়া

if (number1 == 0)    //  যদি বৃত্ত হয়
{
  cout << "area is: ";
  cout << 3.1416 * number2 * number2;
  cout << "perimeter is: ";
  cout << 2 * 3.1416 * number2 << endl;
}
else if number == 1)  // যদি বর্গ হয়
{
  cout << "area is: ";
  cout << number2 * number2;
  cout << "periment is: ";
  cout << 4 * number2 << endl;
}
else
  cout << "unsupported shape" << endl;

cout << "wow so easy!" << endl;
```

এই programটি লেখা খুবই সহজ। কেবল একটাই ঝামেলা আছে সেটা হল if-elseএ (যদি-নাহলে) শর্ত সত্য হোক বা মিথ্যা হোক আমাদের একটা statementএর (বিবৃতি) বদলে একগুচ্ছ statement (বিবৃতি) execute (নির্বাহ) করতে হবে। এর আগের সব উদাহরণে আমরা দেখেছি if-else (যদি-নাহলে) শর্ত সত্য বা মিথ্যা হলে কেবল একটা মাত্র statement (বিবৃতি) নির্বাহ করতে। তো ঝামেলাটার সমাধানও আসলে সহজ। একগুচ্ছ statementকে { } বাঁকা বন্ধনীর (curly brackets) ভিতরে ঢুকিয়ে দিলেই হলো। এর আগে আমরা জেনেছিলাম দুটো বাঁকা বন্ধনীর (curly brackets) { } দিয়ে আমরা একটা block (মহল্লা) তৈরী করি। তো বাঁকা বন্ধনীর ভিতরে থাকা একগুচ্ছ statementকে আমরা বলি compund statement (যৌগিক বিবৃতি)।

Compound statement হলেই যে তার ভিতরে একাধিক statement থাকতে হবে এমন কথা নেই। Block তৈরী করে কেবল একটা statementও তার ভিতরে লিখতে পারো।

```
if (number % 2 == 0)
  { cout << number << " is even" << endl; }
else
  { cout << number << " is odd" << endl; }
```

Block (মহল্লা) তৈরীর ফলে অনেকসময় dangling elseএর (ঝুলন্ত নাহলে) ঝামেলা সহজে এড়ানো সম্ভব হয়। আগের পাঠের এর চিঠি পাঠানোর খরচ নির্ণয়ের সমস্যাটি বিবেচনা করো। সেখানে else টি কোন if এর তা নিয়ে ঝামেলা তৈরী হয়েছিল আর আমরা empty statement (শুন্য বিবৃতি) দিয়ে সেটা সমাধান করেছিলাম। Empty statement হল স্রেফ ; semicolon (দির্তি) তার আগে কিছু নেই। নীচের programএ (ক্রমলেখ) আমরা ওই ভিতরের if টিকে একটি blockএর ভিতরে ঢুকিয়ে দিলাম। ফলে blockএর (মহল্লা) বাইরে থাকা else টি কোনভাবেই blockএর ভিতরের if এর সাথে মিলানো যাবে না, কাজেই সেটা আর dangling থাকবে না।

```
int cost = 2; //  ঢাকার ভিতরে অসংরক্ষিত
if (zone <= 100)  //  ঢাকার ভিতরে
{
  if (zone % 13 == 0) // সংরক্ষিত
    cost = 3;
}
else           // ঢাকার বাইরে
  cost = 4;
```

Block (মহল্লা) তৈরী করে চাইলে তার ভিতরে কিন্তু আমরা কোন statement একদমই না দিতে পারি অর্থাৎ কেবলই দুটি বাঁকা বন্ধনী (curly brackets) পরপর { }। সেক্ষেত্রে এটাও একরকমের empty statement (শুন্য বিবৃতি) তৈরী হবে। কাজেই empty statement তৈরীর দুটো উপায় আমরা শিখলাম একটা হলো কেবলই ; semicolon (দির্তি) দেয়া আরেকটি হলো { } দুটো বাঁকা বন্ধনীর ভিতরে কিছু না লেখা। প্রথমটির ব্যবহার আগে দেখেছি আর নীচে দ্বিতীয়টি ব্যবহার করে dangling else (ঝুলন্ত নাহলে) আরেকটি সমাধান দেয়া হলো।

```
int cost = 2; //  ঢাকার ভিতরে অসংরক্ষিত

if (zone <= 100)  // ঢাকার ভিতরে
  if (zone % 13 == 0) //  সংরক্ষিত
    cost = 3;
  else              //   অসংরক্ষিত
    {}              //  শুন্য বিবৃতি
else                //  ঢাকার বাইরে
  cost = 4;
```

তাহলে যেখানেই তুমি একটা statement (বিবৃতি) দিতে পারো, সেখানেই তুমি আসলে চাইলে একটা statementএর বদলে একটা compund statementও (যৌগিক বিবৃতি) দিতে পারো, আবার একটা empty statementও (শুন্য বিবৃতি) দিতে পারো। এখন থেকে আমরা যখন statement (বিবৃতি) বলবো তখন তুমি সেটা মানে কেবল একটা statement বুঝবে না, বরং দরকার মতো সেটা যে compound statementও (যৌগিক বিবৃতি) হতে পারে বা empty statementও (শুন্য বিবৃতি) হতে পারে, তা বুঝে নিবে কেমন!

১৫.৭ Error Detection (ত্রুটি শনাক্তকরণ)

একটা দ্বিঘাত সমীকরণ $ax^2 + bx + c = 0$ এর সহগগুলো (coefficient) a, b ও ধ্রুবক (constant) c এর মান input (যোগান) নিয়ে xএর মান দুটি নির্ণয় করার জন্য একটি program (ক্রমলেখ) রচনা করো। এই programএ তোমাকে সব রকমের execution-time (নির্বাহকালীন) error detect (ত্রুটি শনাক্ত) করে তা ব্যবহারকারীকে জানাতে হবে।

```
#include <cmath>  // বর্গমূলের জন্য sqrt বিপাতক লাগবে

//  ওপরের অংশ main বিপাতকের আগে, নীচের অংশ ভিতরে

float a, b, c;  //  সহগ ও ধ্রুবক গুলোর জন্য চলক

cout << "equatioin ax^2 + bx + c = 0" << endl;
cout << "values of a b c ? "; //  যোগান যাচনা
cin >> a >> b >> c;              //  যোগান নেয়া

float d = b*b - 4*a*c;           //  নিশ্চায়ক

float x1 = (-b + sqrt(d)) / (2*a);  //  প্রথম সমাধান
float x2 = (-b - sqrt(d)) / (2*a);  //  দ্বিতীয় সমাধান

cout << "first solution x1 = " << x1 << endl;
cout << "second solution x2 = " << x2 << endl;
```

দ্বিঘাত সমীকরণ $ax^2+bx+c=0$ এর সহগ ও ধ্রুবকের মান না জেনেও আমরা সমাধানের সূত্র বের করতে পারি $x = (-b \pm \sqrt{b^2-4ac})/(2a)$। এই সূত্রের বর্গমূল বের করার জন্য আমাদের cmath header file (শির নথি) থেকে sqrt function ব্যবহার করতে হবে। বাদ বাঁকী অংশটুকু সহজ, উপরের programএ (ক্রমলেখ) দেখানো হলো। প্রথমে সমীকরণটা দেখানো হয়েছে। খেয়াল করো x^2 দিয়ে আমরা কিন্তু x এর বর্গ বুঝিয়েছি। সহগ ও ধ্রবকগুলোর মান input (যোগান) নেয়ার পরে আমরা b*b − 4*a*c নির্ণয় করে variable d তে নিয়েছি কারণ এটি দুইটি সমাধানের জন্য দুইবার ব্যবহার করতে হবে। যাইহোক ওপরের অংশটুকু ব্যবহার করে লেখা program (ক্রমলেখ) কাজ করবে যদি সমীকরণটা সহজ সোজা হয়, তাতে কোন ঝামেলা না থাকে! কী রকমের ঝামেলা থাকতে পারে, কিছু অনুমান করতে পারো?

আসলে program (ক্রমলেখ) তৈরীর সময় আমাদের ধরে নিতে হয় যে ব্যবহারকারী সঠিক input (যোগান) যেমন দিতে পারে তেমনি যা ইচ্ছা তা বেঠিক inputও দিতে পারে। এইটা সে ভুল করে করতে পারে, না জেনে করতে পারে, ইচ্ছা করেও করতে পারে। তোমার কাজ নষ্ট করে দেয়ার আরো নানাবিধ উদ্দেশ্যও থাকতে পারে। তবে আমরা আপাতত ধরে নিই ব্যবহারকারী ঝামেলা যা করার তা কেবল ওই সহগ ও ধ্রুবকের মান input (যোগান) দেওয়ার মাধ্যমেই করবে। আর ওই ঝামেলাগুলো করলে যা হবে তা হলো উপরের program আমাদের নিয়ন্ত্রণের বাইরে execution-time (নির্বাহকালীন) error দেখিয়ে বন্ধ হয়ে (abort) যেতে পারে। এরকম একটা error হলো divide by zero (শুন্য দিয়ে ভাগ), আর একটা error হতে পারে ঋণাত্মক সংখ্যার বর্গমূল বের করা! এই দুটো ত্রুটিই দ্বিঘাত সমীকরণ সমাধানের ক্ষেত্রে ঘটতে পারে।

এই সব ক্ষেত্রে আমাদের আগে থেকে বুঝতে পারতে হবে যে ওই রকম ত্রুটিপূর্ণ ঘটনা ঘটবে কী না, যদি ওইরকম ত্রুটি সত্যিই ঘটে সেটা আমাদের ব্যবহারকারীকে জানাতে হবে। আমাদের তরফে জানানোটা স্বাভাবিক। কিন্তু আমাদের অজান্তে যদি ত্রুটি ঘটে program (ক্রমলেখ) বন্ধ হয়ে যায় তাহলে সেটা কোনভাবেই গ্রহণযোগ্য নয়। সেটা একটা দুর্বল programmingএর (পরিগণনা) উদাহরণ। আর আমরা ত্রুটিটা ঘটবার আগেই ধরতে পারলে সেটা ব্যবহারকারীকে জানিয়ে চাইলে আমাদের programকে (ক্রমলেখ) তারপরেও নির্বাহ করা চালিয়ে যেতে দিতে পারবো। তো আমরা নীচে দ্বিঘাত সমীকরণ সমাধানের পুরো programটি (ক্রমলেখ) লিখবো আর তাতে সব রকম ত্রুটি ধরে সেটা ব্যবহার কারীকে জানানোর চেষ্টা করবো। আর যখন ত্রুটি হচ্ছে তখন আমরা return EXIT_SUCCESS; না করে return EXIT_FAILURE; করবো।

ফিরিস্তি ১৫.৩: Solving Quadratic Equations (দ্বিঘাত সমীকরণ সমাধান)

```
#include <iostream>    //  cout ব্যবহার করার জন্য
#include <cstdlib>     //  EXIT_SUCCESS/FAILURE এর জন্য
#include <cmath>       //  sqrt বিপাতক ব্যবহার করার জন্য

using namespace std;   //  প্রমিত নামাধার

int main()
{
  float a, b, c;       //  সহগ রাখার জন্য চলক।

  cout << "equation ax^2 + bx + c = 0" << endl;
  cout << "values of a b c? ";  //  যোগান যাচনা
  cin >> a >> b >> c;           //  যোগান নেওয়া

  // a বা b যদি শুন্য হয় তখন কী হবে? c শুন্য হলে সমস্যা নেই!
  if (a == 0)   // a শুন্য হলে সমীকরণ দ্বিঘাত নয়, একঘাত!
  {
    if (b == 0) // b শুন্য হলে কোন বৈধ সমীকরণই নয়!
    {
      cout << "not a legal equation!" << endl;
      return EXIT_FAILURE;  // ক্রমলেখ বিফল
    }

    //  b শুন্য নয়, কাজেই সমীকরণ কেবলই একঘাত
    cout << "not a quadratic equation!" << endl;
    cout << "assuming linear equation." << endl;
    cout << "solution is x = " << -c/b << endl;
    return EXIT_SUCCESS; // ক্রমলেখ তবুও সফল ধরা যায়
  }

  // a শুন্য নয়, কাজেই বৈধ দ্বিঘাত সমীকরণ
```

```
    float d = b*b - 4*a*c;  //  নিশ্চায়ক হিসাব করো

    if (d < 0)  // ঋণাত্মক নিশ্চায়কের বর্গমূল করা যায় না!
    {
      cout << "determinant negative!" << endl;
      cout << "no real solution!" << endl;
      return EXIT_FAILURE;  // ক্রমলেখ বিফল
    }

    if (d == 0) //  নিশ্চায়ক শুন্য হলে দুটো সমাধান সমান
    {
      cout << "two solutions are the same!" << endl;
      cout << "coincidental x = " << -b/(2*a) << endl;
      return EXIT_SUCCESS;  // ক্রমলেখ সফল
    }

    // দুটো সমাধান আছে, আর তারা অসমান

    float x1 = (-b + sqrt(d)) / (2*a);  // প্রথম সমাধান
    float x2 = (-b - sqrt(d)) / (2*a);  // দ্বিতীয় সমাধান

    cout << "first solution x1 = " << x1 << endl;
    cout << "second solution x2 = " << x2 << endl;

    return EXIT_SUCCESS;  // ক্রমলেখ সমাধান
}
```

Input (যোগান) নেবার পর প্রথমে যেটি আমাদের বিবেচনা করতে হবে তা হলো সমীকরণটি আসলে দ্বিঘাত সমীকরণ কিনা? যদি a শুন্য হয়, তাহলে সমীকরণে কোন x^2 থাকে না, এটি হয়ে যায় $bx + c = 0$ যেটি একটি একঘাত সমীকরণ। এমন অবস্থায় আমরা আরো পরীক্ষা করে দেখব bও শুন্য কিনা। যদি b শুন্য হয় তাহলে থাকে কেবল $c = 0$, যেখানে কোন variable (চলক) নেই। কাজেই আমাদের error message (ত্রুটি বার্তা) দেখিয়ে return EXIT_FAILURE; বলে ফেরত যেতে হবে। নীচের input-output (যোগান-ফলন) খেয়াল করো, a ও b শুন্য হওয়ায় error message (ত্রুটি বার্তা) দিয়েছে।

```
equation ax^2 + bx + c = 0
values of a b c? 0 0 3
not a legal equation!
```

যদি a শুন্য হয় কিন্তু b শুন্য না হয়, তাহলে আমরা if (a == 0) blockএর (মহল্লা) ভিতরেই আছি, কিন্তু b শুন্য না হওয়ায় সমীকরণটি আসলেই একঘাত হয়েছে $bx + c = 0$। আমরা কিন্তু দ্বিঘাত সমীকরণ সমাধান করতে চাই, কিন্তু আমাদের দেয়া হয়েছে একটা একঘাত সমীকরণ। তুমি চাইলে এখানে error message (ত্রুটি বার্তা) দেখিয়ে বিফল হয়ে ফেরত যেতে পারো। আবার উদারতা দেখিয়ে নীচের মতো ওই একঘাত সমীকরণের সমাধানই output (ফলন) দিতে পারো। তবে সাথে warning messageও (সতর্ক বার্তা) দিয়ে দিলে যে এটা দ্বিঘাত সমীকরণ নয়!

```
equation ax^2 + bx + c = 0
values of a b c? 0 2 1
not a quadratic equation!
assuming linear equation.
solution is x = -0.5
```

যদি a শুন্য না হয় তাহলে এটা একটা বৈধ দ্বিঘাত সমীকরণ। সুতরাং প্রথমে আমরা নিশ্চায়ক (discriminant) নির্ণয় করে একটা varaibleএ নেবো। উপরের programএ খেয়াল করো float d = b*b - 4*a*c; লিখে তাই করা হয়েছে। এখন নিশ্চায়ক যদি ঋণাত্মক (negative) হয় তাহলে তো বর্গমূল নির্ণয় করা সম্ভব না, কিন্তু দ্বিঘাত সমীকরণের সমাধানের সূত্রে নিশ্চায়কের বর্গমূল আমাদের দরকার। কাজেই নিশ্চায়কের মান ঋণাত্মক হলে আমাদের পক্ষে সমাধান করা সম্ভব নয়। একটি error message (ত্রুটি বার্তা) দেখিয়ে return EXIT_FAILURE; ফেরত যাওয়া উচিৎ। নীচের input-output (যোগান-ফলন) খেয়াল করো, ঠিক তাই ঘটেছে।

```
equation ax^2 + bx + c = 0
values of a b c? 2 -5 2
determinant negative!
no real solution!
```

Discriminant (নিশ্চায়ক) যদি ঋণাত্মক (negative) না হয়, তাহলে এবার দেখতে হবে এটি শুন্য কিনা। কারণ শুন্য হলে সেক্ষেত্রে আমাদের সমাধান দুটিই হবে, কিন্তু সমাধান দুটি আবার আলাদা আলাদা না হয়ে একই হবে। এইরকম অবস্থাকে বলা হয় সমাপতিত (coincidental) সমাধান। নীচের input-outputএ (যোগান-ফলন) এটি দেখানো হলো।

```
equation ax^2 + bx + c = 0
values of a b c? 1 -2 1
two solutions are the same!
coincidental x = 1
```

সবশেষের যে অবস্থা সেটি হলো নিশ্চায়ক ঋণাত্মকও নয়, শুন্যও নয়, তাহলে সেটি ধনাত্মক (positive)। আর এটিই হলো সেই অবস্থা আমরা যেটি ধরে নিয়ে একদম শুরুতে একটা ছোট program (ক্রমলেখ) দেখিয়েছিলাম। কাজেই আমরা সেই কাজটুকু করে দুটো সমাধান আমাদের জানা সূত্রানুযায়ী নির্ণয় করে ফলন দেখিয়ে return EXIT_SUCCESS; করে ক্রমলেখ শেষ করবো। নীচের input-outputএ (যোগান-ফলন) এই অবস্থা দেখানো হলো।

```
equation ax^2 + bx + c = 0
values of a b c? 2 -5 2
first solution x1 = 2
second solution x2 = 0.5
```

উপরের বিস্তারিত programএ (ক্রমলেখ) খেয়াল করো প্রতিটি if এর শর্ত সত্য হলে যে blockটি (মহল্লা) নির্বাহিত হবে সেই block শেষ হয়েছে একটি return দিয়ে। তার মানে ওই শর্তগুলো সত্য হলে programএর (ক্রমলেখ) নীচের কোন অংশ আর নির্বাহিত হবে না। আর একারণে সংশ্লিষ্ট if এর শর্ত মিথ্যা হলে যা হবে সেটি আর আমরা একটি else লিখে তারপর আরেকটি blockএ (মহল্লা) ঢুকিয়ে দেই নি। কারণ if এর শর্ত সত্য হলে যে block (মহল্লা) তার বাইরে পুরোটাইতো এখন else এর জন্য, কাজেই আলাদা করে block করার দরকার নেই।

১৫.৮ Boolean Connectives (বুলক সংযোজক)

একটি program (ক্রমলেখ) রচনা করো যেটি একটি পূর্ণক (integer) input (যোগান) নিবে। তারপর সংখ্যাটি যদি ৭ ও ১৩ উভয় দ্বারা বিভাজ্য হয় তাহলে output (ফলন) দিবে "mixed luck", যদি ৭ দ্বারাও বিভাজ্য না হয় আবার ১৩ দ্বারাও বিভাজ্য না হয় তাহলে output দিবে "no luck", যদি কেবল ৭ দ্বারা বিভাজ্য হয় কিন্তু ১৩ দ্বারা বিভাজ্য নয় তাহলে ফলন দিবে "good luck", আর যদি কেবল ১৩ দ্বারা বিভাজ্য হয় কিন্তু ৭ দ্বারা নয় তাহলে ফলন দিবে "bad luck"। যদি সংখ্যাটি ৭ বা ১৩ যে কোন একটি বা উভয়টি দ্বারা বিভাজ্য হয় তাহলে output দিবে "luck luck"। একটি সংখ্যা একই সাথে উপরের এক বা একাধিক ভাগে পড়তেই পারে।

ফিরিস্তি ১৫.৪: Lucky & Unlucky Numbers (সৌভাগ্য ও দুর্ভাগ্যের সংখ্যা)

```
int number;
cout << "number is? ";
cin >> number;

if (number % 7 == 0 && number % 13 == 0)
  cout << "mixed luck" << endl;

if (number % 7 != 0 && number % 13 != 0)
  cout << "no luck" << endl;

if (number % 7 == 0 && number % 13 != 0)
  cout << "good luck" << endl;

if (number % 13 == 0 && number % 7 != 0)
  cout << "bad luck" << endl;

if (number % 7 == 0 || number % 13 == 0)
  cout << "luck luck" << endl;
```

উপরের programএ (ক্রমলেখ) && হলো "এবং" আর || হলো "অথবা"। তুমি চাইলে সিপিপিতে && এর বদলে and আর || এর বদলে or লিখতে পারো। আর বাংলায় কখনো কখনো আমরা "এবং" এর বদলে "ও" বা "আর" লিখবো, আর "অথবা" এর বদলে লিখবো "বা"। যাই হোক মনে রাখো && এর ফলাফল সত্য হয় যখন এর দুপাশের operandই (উপাদান) সত্য হয়, আর যেকোন একটা মিথ্যা হলেই ফলাফল মিথ্যা। অন্যদিকে || এর ফলাফল মিথ্যা হয় যখন এর দুপাশের operandই (উপাদান) মিথ্যা, আর যে কোন একটি সত্য হলেই ফলাফল সত্য। তো উপরের program বুঝার চেষ্টা করো। খুব কঠিন কিছু নয়। সমস্যাটি ঠিক যেমন করে বাংলায় বর্ণনা করা হয়েছে, programএও যেন ঠিক সে রকম করেই লেখা হয়েছে।

এবার ওই programকে আমরা কিছু উন্নয়নের চেষ্টা করি। একটা বিষয় খেয়াল করো বিভাজ্য হওয়া বা বিভাজ্য না হওয়া আমরা বারবার হিসাব করেছি। এইটা তো হওয়া উচিৎ নয়। তাছাড়া ভাগশেষ বের করা অন্যান্য অনেক operator (অণুক্রিয়া) তুলনায় মোটামুটি সময় সাপেক্ষ কাজ। আমাদের তাই একবার ভাগশেষ হিসাব করে সেটাই বারবার ব্যবহার করা উচিৎ। তো সেই অনুযায়ী আমরা programএ কিছু পরিবর্তন করতে পারি। মুলত ভাগশেষের জন্য আমাদের দুটো পূর্ণক (integer) varaible নিতে হবে `int remain7 = number % 7;` আর `int remain13`

= number % 13; আর তারপর প্রতিটি number % 7 এর বদলে remain7 এবং প্রতিটি number % 13 এর বদলে remain13 লিখতে হবে। খুবই সহজ পরিবর্তন।

কিন্তু আমরা আসলে এই পরিবর্তনের কথা বলছি না। আমরা ভাগশেষগুলো পূর্ণক (integer) চলকে না রেখে বরং সংখ্যাটি ৭ বা ১৩ দ্বারা বিভাজ্য কিনা তার সত্যতা বুলক (Boolean) চলকে রাখতে চাই। এক্ষেত্রে আমরা bool divisible7 = number % 7 == 0; লিখবো। তাতে সংখ্যাটি ৭ দ্বারা বিভাজ্য হলে divisible7 চলকের মান হবে true বা 1 আর ৭ দ্বারা বিভাজ্য না হলে ওই variableএর মান হবে false বা 0। একই ভাবে bool divisible13 = number % 13 == 0; লিখলে divisible13 এর মান হবে true বা 1 যদি সংখ্যাটি 13 দ্বারা বিভাজ্য হয় আর মান হবে false বা 0 যদি সেটি 13 দ্বারা বিভাজ্য না হয়। নীচে এই program দেখানো হলো।

```
int number;
cout << "number is? ";
cin >> number;

bool divisible7 = number % 7 == 0;
bool divisible13 = number % 13 == 0;

if (divisible7 && divisible13)
  cout << "mixed luck" << endl;

if (!divisible7 && !divisible13)
  cout << "no luck" << endl;

if (divisible7 && !divisible13)
  cout << "good luck" << endl;

if (divisible13 && !divisible7)
  cout << "bad luck" << endl;

if (divisible7 || divisible13)
  cout << "luck luck" << endl;
```

উপরের programএ (ক্রমলেখ) খেয়াল করো divisible7 (বা একই ভাবে divisible13) এর মান সত্য নাকি মিথ্যা আমরা কিন্তু divisible7 == true অথবা divisible7 == 1 লিখে করি নাই, যদিও তা করতে পারতাম। আমরা বরং স্রেফ varaibleটা ব্যবহার করেছি কারণ variableটার মানই তো সরাসরি সত্য বা মিথ্যা। আবার আলাদা করে == operator (অণুক্রিয়া) দিয়ে সত্য বা মিথ্যা পরীক্ষা করার দরকার নেই। তবে খেয়াল করো যখন বিভাজ্য নয় পরীক্ষা করতে হবে তখন আমরা !divisible7 (বা একই ভাবে !divisible13) লিখে অর্থাৎ variableএর নামের সামনে ! লাগিয়ে দিয়েছি। এখানে ! হলো নয় বা না operator। তুমি চাইলে ! এর বদলে সিপিপিতে not লিখতে পারতে। নয় operator সত্যকে operand (উপাদান) হিসাবে নিয়ে মিথ্যা ফলাফল দেয় আর মিথ্যাকে operand হিসাবে নিয়ে সত্য ফলাফল দেয়। আর সে কারণে divisible7 == false না লিখে আমরা !divisible7 লিখলেই আমাদের কাজ হয়ে যায়।

১৫.৯ Boolean, Integer, Float (বুলক, পূর্ণক, ভগ্নক)

একটি সংখ্যা জোড় না বিজোড় তা নির্ণয়ের program রচনা করো। তোমার programএ তুমি কোন Boolean variable (বুলক চলক) বা relational operator (অন্বয়ী অণুক্রিয়া) ব্যবহার করতে পারবে না। তোমাকে পূর্ণক মানকেই Boolean হিসাবে ব্যবহার করতে হবে।

```
int number = 41;  // তুমি চাইলে input নিতে পারো।

if (number % 2 != 0)
  cout << "odd" << endl;
else
  cout << "even" << endl;
```

এই programটি (ক্রমলেখ) তুমি চাইলে উপরের মতো করে লিখতে পারো। কোন সংখ্যা ২ দিয়ে ভাগ দিলে যদি ভাগশেষ শুন্য না হয় তাহলে সংখ্যাটি বিজোড়, আর ভাগশেষ শুন্য হলে সংখ্যাটি জোড়। কাজেই programটি সহজেই লিখে ফেলা যায়। কিন্তু এতে অসমান নির্ণয়ের জন্য একটি operator != ব্যবহার করতে হচ্ছে, যেটি চাইলে আমরা ব্যবহার না করেও কাজ চালাতে পারি। এর কারণ হলো যে কোন সময় শুন্যকে আমরা মিথ্যা ধরে নিতে পারি আর যে কোন অশুন্য মানকে, সেটা ধনাত্মক হোক বা ঋণাত্মক হোক, আমরা সেটাকে সত্য ধরে নিতে পারি। তাতে আমাদের মানটি আলাদা করি শুন্য কিনা তা আর পরীক্ষা করার দরকার পড়ে না। কাজেই নীচের programএর (ক্রমলেখ) মতো করে != বাদ দিয়ে স্রেফ if (number % 2) লেখা মানেই হলো if (number % 2 != 0) লেখা।

```
int number = 41;  // তুমি চাইলে যোগান নিতে পারো।

if (number % 2)    // ভাগশেষ অশুন্য হলে
  cout << "odd" << endl;
else                // ভাগশেষ শুন্য হলে
  cout << "even" << endl;
```

উপরের উদাহরণে আমরা কেবল পূর্ণক ব্যবহার করেছি। কিন্তু ভগ্নক সংখ্যার ক্ষেত্রেও একই কথা প্রযোজ্য। যেমন নীচের ক্রমলেখতে ভগ্নক সংখ্যাটি শুন্য কিনা তা নির্ণয় করা হয়েছে। খেয়াল করে দেখো আমরা সংখ্যাটিকেই সরাসরি শর্ত হিসাবে ব্যবহার করেছি। সংখ্যাটি শুন্য না হলেই এটি সত্য হবে zero ney output আসবে, আর সংখ্যাটি শুন্য হলে output আসবে zero yes।

```
float number = -3.5;  // তুমি চাইলে যোগান নিতে পারো।

if (number)    // সংখ্যাটি অশুন্য হলে
  cout << "zero ney" << endl;
else                // সংখ্যাটি শুন্য হলে
  cout << "zero yes" << endl;
```

তাহলে এখানকার আলোচনায় আমরা দেখলাম operand (উপাদান) হিসাবে শুন্য হলো মিথ্যা (false) আর অন্য যেকোন ধনাত্মক (positive) বা ঋণাত্মক (negative) পূর্ণক (integer) বা ভগ্নক (float) হলো সত্য (true)। আর relational operators (অন্বয়ী অণুক্রিয়া) আলোচনার সময় জেনেছি ফলাফল (result) হিসাবে সবসময় false হলো 0 এবং true হলো 1। খেয়াল

করো operand (উপাদান) হিসাবে true 0 ছাড়া যে কোন কিছু হলেও ফলাফল (result) হিসাবে true কেবল 1, false অবশ্য উভয় ক্ষেত্রেই কেবল 0।

১৫.১০ Boolean Algebra (বুলক বীজগণিত)

দক্ষতার সাথে if else (যদি নাহলে) ব্যবহার করতে চাইলে আমাদের Boolean algebra (বুলক বীজগণিত) জানা দরকার খানিকটা। অনেক সময় এবং &&, অথবা ||, নয় ! operators (অণুক্রিয়া) বেশ কয়েকবার করে নিয়ে তুমি হয়তো জটিল একটা expression (রাশি) তৈরী করবে যেটা সরাসরি evaluate (মূল্যায়ন) করতে গেলে প্রত্যকটি operator (অণুক্রিয়া) ধাপে ধাপে evaluate করতে হবে। কিন্তু Boolean algebra ব্যবহার করে সেটা হয়তো সরলীকরণ করে ছোট করে অনেক কম operator (অণুক্রিয়া) দিয়েই প্রকাশ করা সম্ভব। Operatorএর সংখ্যা কম হওয়া মানে সেটা দক্ষ হবে, program execution (নির্বাহ) করতে সময় কম লাগবে।

Boolean algebraতে (বুলক বীজগণিত) সত্যকে represent (প্রমূর্তায়ন) করা হয় true বা 1 দিয়ে আর মিথ্যাকে করা হয় false বা 0 দিয়ে। সিপিপিতে operatorসমুহ (অণুক্রিয়া) ফলাফলের ক্ষেত্রে একদম এইরূপ representationই (প্রমূর্তায়ন) মেনে চলে, তবে operandএর (উপাদান) ক্ষেত্রে কিছুটা উদার হয়ে 0 ছাড়া যেকোন মানকেই true হিসাবে ধরে নেয়, false ধরে নেয় যথারীতি কেবল 0 কে। Operand ও ফলাফলের ক্ষেত্রে true এর এই ভিন্নতা মনে রাখবে। অনেক সময় এটি সুবিধাজনক, আবার অনেক সময় এটি অনেক ত্রুটির (error) জন্মদেয়।

Boolean algebraএর (বুলক বীজগণিত) প্রথম যে operator (অণুক্রিয়া) তাহলো নয়, না যেটা ! বা not লিখে প্রকাশ করা হয়। নয় operatorএর operand (উপাদান) ও ফলাফল (result) নীচে খেয়াল করো !true হলো false আর !false হলো true। আমরা এখানে $\equiv$ বা সমতুল (equivalence) প্রতীক ব্যবহার করে বুঝাবো যে ওই প্রতীকের বাম ও ডানপাশ সমতুল।

- !true $\equiv$ false
- !false $\equiv$ true

ধরো দুটো ! পরপর আছে যেমন !!true বা !!false বা !!x, তাহলে ফলাফল কী হবে। এইসব ক্ষেত্রে আমাদের ডানের ! আগে হিসাব করতে হবে, তার ওপর বামের ! ধরে শেষ ফলাফল হিসাব করতে হবে। একারণে ! হলো right associative (ডান সহযোজ্য)। তো এখানে ডানের ! operator true বা false কে উল্টে দিবে আর বামের ! সেটাকে আবার সিধা করবে। সুতরাং !!true হবে true, !!false হবে false, আর !!x হবে x। Boolean algebraতে এই বিধিকে বলা হয় double negation (দুনো ঋণায়ন)। তুমি কি তিন বা বেশী সংখ্যক ! পরপর থাকলে কী হবে বের করতে পারবে? অবশ্যই পারবে, প্রতি দুইটি ! পরস্পরকে বাতিল করে দিবে।

- !!x $\equiv$!(!x) — right associative (ডান সহযোজ্য)
- !!x $\equiv$ x — double negation (দুনো ঋণায়ন)
- !!true $\equiv$ true — double negation (দুনো ঋণায়ন)
- !!false $\equiv$ false — double negation (দুনো ঋণায়ন)

Boolean algebraএর (বুলক বীজগণিত) দ্বিতীয় operator (অণুক্রিয়া) এবং, ও যেটা && বা and লিখে প্রকাশ করা হয়। লক্ষ্য করো এবং operatorএর ফলাফল (result) সত্য যখন উভয় operandই (উপাদান) সত্য, আর যেকোন একটি operand মিথ্যা হলেই ফলাফল মিথ্যা।

- true && true ≡ true
- false && true ≡ false
- true && false ≡ false
- false && false ≡ false

একটি operand সত্য বা মিথ্যা ধরে নিলে এবং && operatorএর জন্যে আমরা বেশ কিছু সরলীকরণ করে ফেলতে পারি যেগুলোকে আমরা true simplification (সত্যের সরল) ও false simplification (মিথ্যার সরল) বলবো। কোন একটি operand আমরা যদি আগেই বুঝে ফেলি সেটি সত্য না মিথ্যা তাহলে আমরা এই সরলীকরণগুলো কাজে লাগাতে পারবো।

- x && true ≡ x সত্যের সরল
- x && false ≡ false মিথ্যার সরল
- true && x ≡ x সত্যের সরল
- false && x ≡ false মিথ্যার সরল

Boolean algebraএর (বুলক বীজগণিত) তৃতীয় operator (অণুক্রিয়া) অথবা, বা যেটা || বা or লিখে প্রকাশ করা হয়। লক্ষ্য করো অথবা operatorএর ফলাফল (result) মিথ্যা যখন উভয় operandই (উপাদান) মিথ্যা, আর যেকোন একটি operand সত্য হলেই ফলাফল সত্য।

- true || true ≡ true
- false || true ≡ true
- true || false ≡ true
- false || false ≡ false

একটি operand সত্য বা মিথ্যা ধরে নিলে অথবা || operatorএর জন্যে আমরা বেশ কিছু সরলীকরণ করে ফেলতে পারি যেগুলোকে আমরা true simplification (সত্যের সরল) ও false simplification (মিথ্যার সরল) বলবো। কোন একটি operand আমরা যদি আগেই বুঝে ফেলি সেটি সত্য না মিথ্যা তাহলে আমরা এই সরলীকরণগুলো কাজে লাগাতে পারবো।

- x || true ≡ true সত্যের সরল
- x || false ≡ x মিথ্যার সরল
- true || x ≡ true সত্যের সরল
- false || x ≡ x মিথ্যার সরল

Boolean algebraতে operatorগুলোর (অণুক্রিয়া) precedence order (অগ্রগণ্যতার ক্রম) হলো প্রথমে নয় !, তারপর এবং &&, আর শেষে অথবা ||, এই ক্রমের অন্যথা করতে চাইলে প্রয়োজনে বন্ধনী ব্যবহার করতে হবে। তাছাড়া দ্ব্যর্থবোধকতা এড়াতে বন্ধনী ব্যবহার করা উচিৎ। নীচের উদাহরণ দুটি খেয়াল করো। প্রথমটিতে আগে ! তারপর &&, শেষে || করতে হবে, বন্ধনী ব্যবহার করে সেটাই বুঝানো হয়েছে। দ্বিতীয় উদাহরণটিতে ! আগে করলেও && আগে || করায় সেটা সঠিক হয় নি। এখানে ≢ দিয়ে বুঝানো হয়েছে দুইপাশ পরস্পরের সমতুল নয়।

- x && !y || z ≡ (x && (!y)) || z আগে !, মাঝে &&, পরে ||
- x && !y || z ≢ x && ((!y) || z) আগে !, মাঝে || নয়, পরেও && নয়

Mathematical operators (গাণিতিক অণুক্রিয়া) ও assignmentএর (আরোপণ) সাথে যদি logical operatorগুলো (যৌক্তিক অণুক্রিয়া) মিলিয়ে দেখা হয় তাহলে সব মিলিয়ে precedence order (অগ্রগণ্যতার ক্রম) নিম্নরূপ:

১. ++ −− unary postfix (একিক উত্তর) left associative (বাম থেকে ডানে)

২.	++ −− + − !	unary prefix (একিক পূর্ব) right associative (ডান থেকে বামে)
৩.	∗ / %	binary (দুয়িক) left associative (বাম থেকে ডানে)
৪.	+ −	binary (দুয়িক) left associative (বাম থেকে ডানে)
৫.	&&	binary (দুয়িক) left associative (বাম থেকে ডানে)
৬.	||	binary (দুয়িক) left associative (বাম থেকে ডানে)
৭.	= += −= ∗= /= %=	binary (দুয়িক) right associative (ডান থেকে বামে)
৮.	,	binary (দুয়িক) left associative (বাম থেকে ডানে)

১৫.১১ Boolean Equivalence (বুলক সমতুল)

এবার আমরা বেশ কিছু equivalence law (সমতুল বিধি) দেখবো। এই lawগুলোর বামপাশ আর ডানপাশ সবসময় equivalent। আমরা তাই এগুলো ব্যবহার করে বিভিন্ন সময়ে আমাদের logical expression (যৌক্তিক রাশি) সরল করার চেষ্টা করবো।

নীচের দুটো বিধি হলো এবং, অথবার বিনিময় বিধি (commutative law)। বিনিময় বিধিতে operatorএর (অণুক্রিয়া) operandগুলো পার্শ পরিবর্তন করলেও ফলাফল একই থাকে।

- x && y ≡ y && x বিনিময়
- x || y ≡ y || x বিনিময়

নীচের দুটো বিধি হলো সহযোজন বিধি (associative law)। এই বিধিতে একই operator (অণুক্রিয়া) পরপর থাকলে আমরা যে কোনটি আগে মূল্যায়ন (evaluate) করে তার ফলাফলের সাথে অন্য operatorএর মূল্যায়ন করতে পারি, আর তাতে ফলাফল একই হবে।

- x && y && z ≡ (x && y) && z ≡ x && (y && z) সহযোজ্য
- x || y || z ≡ (x || y) || z ≡ x || (y || z) সহযোজ্য

নীচের দুটো বিধি হলো বন্টন বিধি (distributive law)। এই বিধিতে দুটি ভিন্ন operator (অণুক্রিয়া) পরপর থাকলে আমরা একটিকে আরেকটির ওপর বন্টন করে দিতে পারি। পাটিগণিতে বন্টন বিধির উদাহরণ হলো $x * (y + z) = x * y + x * z$।

- x && y || z ≢ x && (y || z) ≡ (x && y) || (x && z) বন্টন
- x || y && z ≢ (x || y) && z ≡ (x && z) || (y && z) বন্টন
- x || y && z ≡ x || (y && z) ≡ (x || y) && (x || z) বন্টন
- x && y || z ≡ (x && y) || z) ≡ (x || z) && (y || z) বন্টন

নীচের বিধিগুলো হলো শোষন বিধি (absorption law)। প্রথম চারটি বিধিতে খেয়াল করো x যদি true হয় তাহলে x || y বা y || x এর মানও true আর ফলে && এর ফলাফলও true। আবার x যদি false হয় তাহলে && এর ফলাফল অবশ্যই false। তাহলে বামদিকের রাশিগুলোর মান সবসময় x এর মান যা তাই। একই ভাবে শেষের চারটি বিধিতে খেয়াল করো x যদি false হয় তাহলে x && y বা y && x এর মানও false। আবার x যদি true হয় তাহলে || এর ফলাফল অবশ্যই true। তাহলে বামদিকের রাশিগুলোর মান সব সময় x এর মান যা তাই। কাজেই এই বিধিগুলো তোমাকরে বুলক রাশিকে কত সহজ ও ছোট করে ফেলে!

- x && (x || y) ≡ x শোষন
- x || (x && y) ≡ x শোষন
- x && (y || x) ≡ x শোষন
- x || (y && x) ≡ x শোষন
- (x || y) && x ≡ x শোষন
- (x && y) || x) ≡ x শোষন
- (y || x) && x ≡ x শোষন
- (y && x) || x) ≡ x শোষন

নীচের বিধি দুটোতে operatorগুলোর operandদুটো একই। এবং && ও অথবা || উভয়ের ফলাফল এক্ষেত্রে সবসময় operandটির মান যা তাই হবে। একটি operandএর নিজের সাথে নিজের ওপর কোন operator (অণুক্রিয়া) প্রযুক্ত হলে ফলাফল যদি operandটিই হয় তাহলে operatorটির এই ধর্মকে বলা হয় idempotence (অস্বক্রিয়তা)। সব operatorই কিন্তু idempotent নয়, যেমন পাটিগণিতে সর্বাবস্থায় $x + x = x$ সত্য নয়, কাজেই যোগ + idempotent নয়। Boolean algebraতে এবং && ও অথবা || উভয়েই idempotent।

- x && x ≡ x অস্বক্রিয়তা
- x || x ≡ x অস্বক্রিয়তা

নীচের বিধি দুটোতে operatorগুলোর (অণুক্রিয়া) operandদুটো পরস্পরের বিপরীত। এবং && এর ফলাফল এক্ষেত্রে সবসময় false হবে, কারণ দুটো operandএর মধ্যে যে কোন একটি তো মিথ্যা হবেই, আর যে কোন একটি মিথ্যা হলেই এবং এর ফলাফল মিথ্যা। তাই এই বিধিকে বলা হয় contradiction (অসঙ্গতি)। আর অথবা || এর ফলাফল এক্ষেত্রে সবসময় true হবে, কারণ দুটো operandএর মধ্যে যে কোন একটি তো সত্য হবেই, আর যে কোন একটি সত্য হলেই অথবা এর ফলাফল সত্য। তাই এই বিধিকে বলা হয় excluded middle (নঞ মধ্যম)।

- x && !x ≡ false অসঙ্গতি
- x || !x ≡ true নঞ মধ্যম

নীচের বিধি দুটোর নাম De Morgan's Law (ডি মরগানের বিধি)। এই বিধিদুটো খুবই গুরুত্বপূর্ণ এবং প্রায়শই Boolean expressionএর সরলীকরণে ব্যবহৃত হয়। এই বিধি অণুযায়ী এবং && এর ফলাফলের ওপর নয় ! করলে যে ফলাফল পাওয়া যায় তা আগে operandগুলোর ওপরে নয় ! করে সেই ফলাফলের ওপর অথবা || চালিয়ে পাওয়া ফলাফলের সমতুল। একই ভাবে অথবা || এর ফলাফলের ওপর নয় ! করলে যে ফলাফল পাওয়া যায় তা আগে operandগুলোর ওপরে নয় ! করে সেই ফলাফলের ওপর এবং && চালিয়ে পাওয়া ফলাফলের সমতুল।

- !(x && y) ≡ !x || !y ডি মরগান
- !(x || y) ≡ !x && !y ডি মরগান

১৫.১২ Truth Table (সত্যক সারণী)

Equivalence lawগুলো যে সঠিক, অথবা যে কোন দুটো Boolean expression সমতুল কিনা, এইটা তুমি কীভাবে প্রমাণ করবে। প্রমাণ করাটা আসলে খুবই সহজ। Operandগুলোর মানের

যত রকম combinaiton (সমাবেশ) সম্ভব, প্রতিটির জন্য তোমাকে equivalence lawএর বাম ও ডান পাশ সমান কিনা পরীক্ষা করে দেখতে হবে। আমরা সাধারণত truth table (সত্যক সারণী) ব্যবহার করে সেটা করে থাকি। চলো উদাহরণ হিসাবে আমরা De Morgan's Law দু-টোর প্রথমটি প্রমাণ করি। একই পদ্ধতি অনুসরণ করে তুমি ডি মরগানের অন্য বিধিটি প্রমাণ করতে পারবে। আর চাইলে উপরের অন্যান্য যে কোন সমতুলের বিধিগুলোও নিজে নিজে প্রমাণ করবে।

ডি মরগানের প্রথম সূত্রটিতে variable (চলক) আছে দুইটি x ও y, আর variable দুটির মান সম্ভব কেবল true ও false। এখন দুটি variableএর জন্যে দুটি মান নিয়ে আমরা চারটি combination (সমাবেশ) পেতে পারি। এর প্রতিটির জন্য আমরা বিধিটির বাম পাশ ও ডান পাশ মূল্যায়ন (evaluate) করে দেখবো। এখানে বলে রাখি কোন সমতুল বিধিতে (equivalence law) যদি ৩টি variable থাকে তাহলে সমাবেশ হবে ৮টি, ৪টি থাকলে হবে ১৬টি, অর্থাৎ nটি variable থাকলে combination হবে 2^nটি। আর এর প্রতিটি combinationএর জন্য truth tableতে (সত্যক সারণী) একটি করে row (আড়ি) থাকবে। Truth tableতে columnগুলো (খাড়ি) হবে বিভিন্ন subexpressionএর (উপরাশি) মান যে গুলোর মান আমাদের মূল্যায়ন করতে হবে যদি আমরা মুল expressionএর মান পেতে চাই। যেমন !(x && y) মূল্যায়ন করতে গেলে আমাদের x && y আগে মূল্যায়ন করতে হবে, তেমনি ভাবে !x || !y মূল্যায়ন করতে গেলে !x ও !y মূল্যায়ন করতে হবে।

Truth Table (সত্যক সারণী)

x	y	x && y	!(x && y)	!x	!y	!x || !y
true	true	true	false	false	false	false
true	false	false	true	false	true	true
false	true	false	true	true	false	true
false	false	false	true	true	true	true

উপরের truth table (সত্যক সারণীতে) প্রতিটি row (আড়ি) খেয়াল করো:

১. প্রথম rowতে (আড়ি) x ও y উভয়ের মানই true। সুতরাং x && y ও true, ফলে !(x && y) হবে false। তারপর !x আর !y উভয়ই হলো false, ফলে !x || !y হলো false। কাজেই !(x && y) আর !x || !y উভয়ের মান সমান।

২. দ্বিতীয় rowতে (আড়ি) x,y যথাক্রমে true, false, ফলে x && y হলো false আর !(x && y) হলো true। তারপর !x ও !y হবে যথাক্রমে false ও true, ফলে !x || !y হলো true। সুতরাং !(x && y) আর !x || !y এর মান সমান।

৩. তৃতীয় rowতে (আড়ি) x,y যথাক্রমে false, true, ফলে x && y হলো false আর !(x && y) হলো true। তারপর !x ও !y হবে যথাক্রমে true ও false, ফলে !x || !y হলো true। সুতরাং !(x && y) আর !x || !y এর মান সমান।

৪. চতুর্থ rowতে (আড়ি) x ও y উভয়ের মানই false। সুতরাং x && y ও false, ফলে !(x && y) হবে true। তারপর !x আর !y উভয়ই হলো true, ফলে !x || !y হলো true। কাজেই !(x && y) আর !x || !y উভয়ের মান সমান।

সুতরাং operandগুলোর (উপাদান) মান যাই হোক না কেন সর্বাবস্থায় !(x && y) আর !x || !y এর মান সমান, অর্থাৎ তারা একে অপরের সমতুল প্রমাণ হয়ে গেলো।

১৫.১৩ Boolean Simplification (বুলক সরলীকরণ)

Conditional programmingএ (শর্তালি পরিগণনা) Boolean algebra (বুলক বীজগণিত) ঠিক কী কাজে লাগে? Boolean algebra ব্যবহার করে কিছু সরলীকরণের উদাহরণ দেখাও। আর এই সরলীকরণের কারণে programএ (ক্রমলেখ) কী প্রভাব পড়ে সেটাও দেখাও।

ধরো তোমাকে একটি program (ক্রমলেখ) লিখতে হবে যেটি তুমি কোন শ্রেণীতে পড়ো আর তোমার বয়স কত এই দুটি input (যোগান) নিয়ে জানাবে তুমি মোরগ লড়াই খেলতে পারবে কি না। তুমি যদি পঞ্চম শ্রেণীতে পড়ো তাহলে তুমি মোরগ লড়াই খেলতে পারবে। আর তুমি যদি পঞ্চম শ্রেণীতে নাও হও কিন্তু তোমার বয়স যদি ১০ বছর হয় তাহলেও তুমি মোরগ লড়াই খেলতে পারবে। এই programটি আমরা if-else দিয়ে খুব সহজে লিখে ফেলতে পারি।

নীচের program খেয়াল করো। আমরা দুটো variable ব্যবহার করছি level ও age, যে দুটো প্রথমে ঘোষণা (declare) করে তারপর input prompt (যোগান যাচনা) দেখিয়ে input (যোগান) নিতে হবে। ধরে নিই তুমি ওগুলো নিজে নিজে করতে পারবে। আমরা কেবল প্রাসঙ্গিক অংশটুকু দেখি। প্রথমে if (level==5) দিয়ে পরীক্ষা করা হলো পঞ্চম শ্রেণী কিনা, হলে output (ফলন) হবে "can play"। আর পঞ্চম শ্রেণী যদি না হয় কিন্তু বয়স যদি ১০ বছর হয় সেটা পরীক্ষা করার জন্য আমাদের লাগবে if (level != 5 && age == 10) যেটি আমরা আগের if এর else এর সাথে লাগিয়ে দিবো। আর সবশেষে কোন if এর শর্তই সত্য না হলে আমরা output (ফলন) দেখাবো "can't play"। একটা গুরুত্বপূর্ণ বিষয় খেয়াল করো, বাংলা ভাষায় যেটা "কিন্তু" সেটা সিপিপিতে গিয়ে হয়ে যাচ্ছে "এবং" &&।

```
if (level == 5)
  cout << "can play" << endl;
else if (level != 5 && age == 10)
  cout << "can play" << endl;
else  // উপরের কোনটিই না হলে
  cout << "can't play" << endl;
```

উপরের ক্রমলেখতে দুটো if এর শর্ত সত্য হলেই আমাদের একই ফলন দেখাতে হয়। আমরা তাই চেষ্টা করতে চাই একটা if দিয়ে বিষয়টা সামলাতে। সেটা করা খুবই সহজ যদি তুমি সমস্যাটা উল্টো দিক থেকে ভাবো। তুমি মোরগ লড়াই খেলতে পারবে যদি তুমি ৫ম শ্রেণী পড়ো অথবা তুমি ৫ম শ্রেণীতে না কিন্তু তোমার বয়স ১০ বছর হলে। তো এই থেকে তুমি খুব সহজে খেলতে পারার শর্ত লিখে ফেলতে পারো level == 5 || level != 5 && age == 10, তাই না!

```
if (level == 5 || level != 5 && age == 10)
  cout << "can play" << endl;
else  // উপরের শর্ত সত্য না হলে
  cout << "can't play" << endl;
```

এখন কথা হচ্ছে এই যে খানিকটা জটিল একটা শর্ত আমরা লিখে ফেললাম, এটাকে কি কোন ভাবে সরলীকরণ করা যায়? সরলীকরণ করার জন্য চলো ধরে নিই p ≡ level == 5 আর q ≡ age == 5। তাহলে level != 5 কে লেখা যায় !p। ফলে আমাদের শর্তটি দাঁড়ালো p || !p && q, আমরা এটিকে Boolean algebra (বুলক বীজগণিত) দিয়ে সরল করবো।

```
  p || !p && q                    প্রদত্ত শর্ত যা সরল করতে হবে
≡ (p || !p) && (p || q)           বন্টন বিধি (distribution)
```

≡ true && (p || q) নঞ মধ্যম (excluded middle)

≡ p || q সত্যের সরল (true simplification)

সুতরাং উপরের সরলের ফলে প্রাপ্ত expression (রাশি) অনুযায়ী আমাদের program দাঁড়াবে নিম্নরূপ, যেখানে আমাদের একটি অতিরিক্ত শর্ত আর মূল্যায়ন করতে হচ্ছে না। আমরা p এর বদলে level == 5 আর q এর বদলে age == 10 লিখবো।

```
if (level == 5 || age == 10)
  cout << "can play" << endl;
else  //  উপরের শর্ত সত্য না হলে
  cout << "can't play" << endl;
```

একই রকম আরেকটি উদাহরণ দেখো। ধরো কোন একটা programএ (ক্রমলেখ) শর্ত দাঁড়াচ্ছে !(p && (!p || q)) || q। এখন কথা হচ্ছে এটিকে সরল করলে কী দাঁড়াবে।

!(p && (!p || q)) || q প্রদত্ত শর্ত যা সরল করতে হবে

≡ !((p && !p) || (p && q)) || q বন্টন বিধি (distribution)

≡ !(false || (p && q)) || q অসঙ্গতি (contradiction)

≡ !(p && q) || q মিথ্যার সরল (false simplification)

≡ (!p || !q) || q ডি মরগান (De Morgan)

≡ !p || (!q || q) সহযোজন (associative)

≡ !p || true নঞ মধ্যম (excluded middle)

≡ true সত্যের সরল (true simplification)

উপরের সরলীকরণের ফলে আমরা if (!(p && (!p || q)) || q) না লিখে কেবল if (true) লিখতে পারবো। কিন্তু একটা বিষয় দেখেছো, সরলীকরণের ফলাফল একদম একটা ধ্রুবক মান true হয়ে গেছে। এর অর্থ প্রদত্ত শর্তের মান কখনো variable p বা q এর ওপর নির্ভর করেনা। সুতরাং আমাদের আদৌ কোন if লাগানোর দরকার নাই। কারণ শর্ত সত্য হলে যেটি করতে হতো শর্ত সবসময় সত্য হওয়ায় তুমি সেটি এখন শর্ত পরীক্ষণ ছাড়াই করবে।

```
// if (true)  // শর্ত লেখার দরকার নাই, টীকায় আটকে দিয়েছি
  cout << "kee moja" << endl; // কেবল এটি লিখলেই হবে
```

তুমি এবার জিজ্ঞেস করতে পারো সরলীকরণের ফলে যদি false আসে তাহলে কী হবে? সত্যিই তো কী হবে? সেক্ষেত্রে আমাদের লিখতে হবে if (false) তাই না! কিন্তু সেটা মানে তো শর্ত সব সময় মিথ্যা, শর্তটির সত্য হওয়ার কোন সম্ভাবনা নেই। আর সেক্ষেত্রে শর্ত সত্য হলে যা করার কথা ছিলো সেটা কখনোই করতে হবে না। ফলে তুমি এই if (false) আর তারপর শর্ত সত্য হলে যা করতে তার সব program (ক্রমলেখ) থেকে মুছে দিতে দিতে পারো।

```
// if (false) // শর্ত লেখার দরকার নাই, টীকায় আটকে দিয়েছি
//  cout << "kee moja" << endl; // শর্ত সব সময় মিথ্যা
```

১৫.১৪ Ladder, Nesting, Connective(মই, অন্তান্তি, সংযোজক)

If-else ladder (যদি-নাহলে মই) ও nested if-else (অন্তান্তি যদি-নাহলে) ব্যবহার না করে কী ভাবে connectives (সংযোজক) এবং &&, অথবা ||, নয় ! ব্যবহার করে একই উদ্দেশ্য বাস্তবায়ন করা যায় তা আলোচনা করো। অথবা উল্টোটা অর্থাৎ connective ব্যবহার না করে কী ভাবে if-else ladder বা nested if ব্যবহার করে কাজ চালানো যায় তা দেখাও।

নীচের উদাহরণগুলো খেয়াল করো। এগুলোতে দুটো করে স্তম্ভ আছে। বামপাশের স্তম্ভে if-else ladder (যদি-নাহলে মই) অথবা nested if-else (অন্তান্তি যদি-নাহলে) দিয়ে program লেখা হয়েছে, আর ডান পাশের স্তম্ভে তার equivalent (সমতুল) program (ক্রমলেখ) লেখা হয়েছে connectives (সংযোজক) এবং && অথবা || না ! দিয়ে। আমরা আসলে সুবিধামতো কখনো বামপাশের মতো করে লিখি আবার কখনো ডানপাশের মতো করেও লিখি।

```
if (cond1)
  cout << "something";
else if (cond2)
  cout << "something";
else
  cout << "other thing";
cout << "be done" << endl;
```

```
if (cond1 || cond2)
  cout << "something";
else
  cout << "other thing";
cout << "koro" << endl;
```

উপরের if-else ladderএর (যদি-নাহলে মই) উদাহরণে খেয়াল করো cond1 সত্য হলেও "something" outputএ (ফলন) যাবে আবার cond2 সত্য হলেও outputএ (ফলন) যাবে "something"। আর এ দুটোই মিথ্যা হলে outputএ যাবে "other thing"। বাম ও ডান উভয় পাশের programএই (ক্রমলেখ) এই একই ব্যাপার ঘটবে। একটা বিষয় উল্লেখ করা দরকার: cond1 সত্য হলে বামপাশে দেখো cond2 পরীক্ষণই দরকার পরে না। ডানপাশেও আসলে একই ঘটনা ঘটবে। অথবা || এর ফলাফল যেহেতু যে কোন একটি operand সত্য হলেই সত্য হয়, সেহেতু cond1 সত্য হলেই cond2 এর মুল্যায়ন ছাড়াই || এর ফলাফল সত্য হয়ে যাবে। এই যে ব্যাপারটি এটাকে বলা partial evaluation (আংশিক মুল্যায়ন), এতে অদরকারী কাজ কিছুটা কমে, program (ক্রমলেখ) কিঞ্চিত দ্রুতগতির হয়।

```
if (cond1)
  if (cond2)
    cout << "something";
  else
    cout << "other thing";
else
  cout << "be done" << endl;
```

```
if (cond1 && cond2)
  cout << "something";
else
  cout << "other thing";
cout <<"be done"<< endl;
```

উপরের nested if-elseএর (অন্তান্তি যদি-নাহলে) উদাহরণে খেয়াল করো cond1 সত্য হলে তারপর cond2ও সত্য হলে "something" outputএ (ফলন) যাবে। আর শর্তদুটোর যেকোন একটা মিথ্যা হলেও outputএ (ফলন) যাবে "other thing"। বাম ও ডান উভয় পাশে programএই (ক্রমলেখ) এই একই ব্যাপার ঘটবে। এখানেও সেই একটা বিষয় উল্লেখ করা দরকার: cond1 মিথ্যা হলে বামপাশে দেখো cond2 পরীক্ষণই দরকার পরে না। ডানপাশেও আসলে একই ঘটনা ঘটবে। অথবা && এর ফলাফল যেহেতু যে কোন একটি operand মিথ্যা

হলেই মিথ্যা হয়, সেহেতু cond1 মিথ্যা হলেই cond2 এর মুল্যায়ন ছাড়াই && এর ফলাফল মিথ্যা হয়ে যাবে। এই যে ব্যাপারটি এটাকে বলা partial evaluation (আংশিক মুল্যায়ন), এতে অদরকারী কাজ কিছুটা কমে, গতি কিছুটা বাড়ে।

```
if (cond)                          if (!cond)
  cout << "something";               cout << "other thing
 ";
else                                else
  cout << "other thing";              cout << "something";
cout << "be done" << endl;          cout << "be done" <<
 endl;
```

উপরের উদাহরণে বামপাশে cond ব্যবহার করা হয়েছে আর ডানপাশে !cond। ফলে শর্ত সত্য হলে যা করতে হবে আর মিথ্যা হলে যা করতে হবে এই দুটো স্থান বদলাবদলি করেছে।

```
if (cond1)                         if (cond1 || !cond2)
  cout << "something";               cout << "something";
else if (cond2)                    else
  cout << "other thing";              cout << "other thing";
else                               cout <<"be done"<< endl;
  cout << "something";
cout << "be done" << endl;
```

উপরের উদাহরণে লক্ষ্যে করে দেখো "something" outputএ (ফলন) যাবে যদি cond1 সত্য হয় অথবা যদি cond2 মিথ্যা হয়, অন্য কথায় !cond2 সত্য হয়। আর cond1 মিথ্যা হলে তারপর cond2ও মিথ্যা হলে outputএ (ফলন) যাবে "other thing"। ঠিক এই ব্যাপারটিই উভয়পাশের programএ প্রতিফলিত হয়েছে।

```
if (cond1)                         if (!cond1 || cond2)
  if (cond2)                         cout << "something";
    cout << "something";           else
  else                                 cout << "other thing";
    cout << "other thing";          cout << "be done" <<
 endl;
else
  cout << "something";
cout << "be done" << endl;
```

উপরের উদাহরণটি একটু জটিল। বামপাশে খেয়াল করো "something" outputএ যাবে যদি cond1 মিথ্যা হয় অথবা তা না হলে যদি cond2 সত্য হয়। কথায় বললে ঠিক তাই-ই ডানপাশেও লিখা হয়েছে। আর একটু বেশী গভীরে বুঝতে চাইলে ধরো বামপাশে "something" outputএ যাবে যদি cond1 && cond2 || !cond1 সত্য হয়। Boolean algebra দিয়ে সরলীকরণ করলে এটি আসবে !cond1 || cond2, তুমি নিজে চেষ্টা করে দেখো।

```
bool cond = true;                  if (cond1 && cond2)
if (!cond1)                          cout << "something";
  cond = false;                    else
```

```
if (!cond2)                    cout << "other thing";
  cond = false;              cout << "be done" << endl;
if (cond)
  cout << "something";
else
  cout << "other thing";
cout << "be done" << endl;
```

উপরের এই উদাহরণটি খেয়াল করো। প্রথম দুটি if একেবারে আলাদা আলাদা, কেউ কারো nestedও (অন্তান্তি) নয়, আবার ladderও (মই) নয়। ডানপাশে যেমন খুব সুন্দর করে সংক্ষেপে আমরা cond1 && cond2 লিখেছি। অনেকসময়ই এটা করা সম্ভব হয় না। কারণ শর্তদুটো আলাদা করে প্রথমে মূল্যায়ন করাটা হয়তো বেশ এক একটা কাজ। তো এইরকম ক্ষেত্রে আমরা বামপাশে যেটি করেছি আলাদা একটা varaible নিয়েছি cond যেখানে মুলত আমরা && এর ফলাফল চাই। আমরা জানি && ফলাফল যে কোন একটি operand (উপাদান) মিথ্যা হলেই মিথ্যা হয়। তাই আমরা শুরুতে cond এর মান নিয়েছি true, এরপর cond1 মিথ্যা হলে অর্থাৎ !cond1 সত্য হলে আমরা cond কে মিথ্যা করে দিয়েছি। একই ভাবে cond2 মিথ্যা হলে অর্থাৎ !cond2 সত্য হলেও আমরা cond কে মিথ্যা করে দিয়েছি। তাহলে দুটো শর্তের যে কোনটি মিথ্যা হলেই cond মিথ্যা হয়ে যাবে। ঠিক && এর ফলাফলের মতো। শেষের if elseএ এবার cond ব্যবহার করে output দেবার পালা। তবে একটা বিষয় খেয়াল করো ডানপাশে যেমন cond1 মিথ্যা হলে partial evaluationএর (আংশিক মূল্যায়ন) cond2 আর পরীক্ষণই করা হবে না, বামপাশে কিন্তু তা হচ্ছে না। তুমি যদি এই উন্নয়ন টুকু করতে চাও তাহলে তোমাকে if (!cond2) বদলে লিখতে হবে else if (!cond2)।

```
bool cond = false;             if (cond1 || cond2)
if (cond1)                       cout << "something";
  cond = true;                 else
if (cond2)                       cout << "other thing";
  cond = true;                 cout << "be done" << endl;
if (cond)
  cout << "something";
else
  cout << "other thing";
cout << "be done" << endl;
```

এই উদাহরণটিও ঠিক আগের উদাহরণটি মতো, তবে এখানে || এর জন্য করা হয়েছে। অথবার || ক্ষেত্রে যেকোন একটি operand (উপাদান) সত্য হলেই ফলাফল সত্য হয়, আমরা তাই cond এর আদি মান ধরেছি false। আর তারপর শর্তদুটোর যে কোনটি সত্য হলেই cond কে সত্য করা হয়েছে। তুমি যদি partial evaluation (আংশিক মূল্যায়ন) এখানেও কাজে লাগাতে চাও তাহলে বামপাশে if (cond2) বদলে else if (cond2) লিখবে।

১৫.১৫ If-Else Optimisation (যদি-নাহলে অনুকুলায়ন)

ধরো তোমার ইশকুলে গণিত পরীক্ষায় ৫০ বা বেশী পেলে পাশ, না হলে ফেল। আর ৮০ বা বেশী পেলে তারকা নিয়ে পাশ। তোমার শ্রেণীতে ১০০ জন শিক্ষার্থী আছে, যাদের মধ্যে মোটামুটি ১০ জন

ফেল করবে, ২০ জন তারকা সহ পাশ করবে আর বাঁকী ৭০ জন স্রেফ পাশ করবে। তুমি এমন একটি program (ক্রমলেখ) রচনা করো যেটি একজন শিক্ষার্থীর ছাত্রের নম্বর input (যোগান) নিয়ে ফেল, পাশ, বা তারকা সহ পাশ output (ফলন) দিবে। তোমার ক্রমলেখটি ১০০ জন শিক্ষার্থীর জন্য ১০০ বার আলাদা আলাদা করে চালানো (run) হবে। তবে এই ১০০ বার চালানোতে মোট সময় যাতে কম লাগে programটা সেটা মাথায় রেখে রচনা করতে হবে।

```
if (number >= 50)                  // যদি পাশের নম্বর
  cout << "pass" << endl;          // পাশ ফলন
else                               // না হলে
  cout << "fail" << endl;          // ফেল ফলন

if (number >= 80)                  // যদি তারকা নম্বর
  cout << "star" << endl;          // তারকা ফলন
```

ধরো উপরের মতো করে তুমি program তৈরী করেছো। যে শিক্ষার্থী ফেল করলো বা পাশ করলো বা তারকা সহ পাশ করলো, তার জন্য তো যা output তা দেখাতেই হবে, সেখানে সময় কম লাগা বেশী লাগার ব্যাপার নাই। সময় কম বা বেশী লাগার প্রশ্ন হলো তুমি কতবার শর্ত পরীক্ষা করে কাজটা করতে পারছো সেটাতে। যেমন ধরো একজন ফেল করা শিক্ষার্থীর জন্য উপরের programএ (number >=50) শর্ত পরীক্ষা হবে আবার program যে ভাবে লেখা হয়েছে তাতে number >= 80 শর্তটিও পরীক্ষা হবে। শর্ত পরীক্ষার ফলাফল সত্য হোক আর মিথ্যা হোক পরীক্ষা তো করতেই হবে। ফলে মোট দুটি শর্ত পরীক্ষা হলো। যে শিক্ষার্থীটি কেবল পাশ করবে খেয়াল করে দেখো তার জন্যেও দুটিই শর্তই পরীক্ষা করতে হবে। একই হবে তারকাসহ পাশের ছাত্রের জন্যেও দুটি শর্তই পরীক্ষা করতে হবে। সুতরাং উপরের program দিয়ে এই সমস্যার সমাধান করলে ১০০ জন শিক্ষার্থীর জন্য মোট শর্ত পরীক্ষা হলো ১০০ * ২ = ২০০ বার।

```
if (number >= 50)                  // যদি পাশের নম্বর
{
  cout << "pass" << endl;          // পাশ ফলন
  if (number >= 80)                // যদি তারকা নম্বর
    cout << "star" << endl;        // তারকা ফলন
}
else                               // না হলে
  cout << "fail" << endl;          // ফেল ফলন
```

এবার একটু ভেবে দেখো পাশ বা ফেল নির্ণয় করার জন্য তো আমাদের একটা শর্ত লাগবেই, কিন্তু যখন আমরা জেনে গেলাম একজন শিক্ষার্থী ফেল করেছে, তখন তার জন্যেও কেন আমরা number >= 80 শর্ত পরীক্ষা করবো? সেটা তো অদরকারী কাজ হবে। সুতরাং তারকা দেখানো অংশটুকু যদি আমরা পাশের জন্য যে অংশ সেখানে একটা block (মহল্লা) তৈরী করে সেই blockএর ভিতরে নিয়ে যাই, তাহলে number >= 80 শর্তটি কেবল পাশ করা শিক্ষার্থীদের জন্য পরীক্ষা হবে। উপরের program দেখো। তো এই ক্ষেত্রে পাশ বা ফেল শিক্ষার্থীর জন্য কেবল ১ টা শর্ত পরীক্ষা হলো আর তারকা পাওয়া ছাত্রের জন্য ২টা সুতরাং মোট শর্ত পরীক্ষণ হলো ২০*২ + (৭০+১০)*১ = ১২০ বার মাত্র। নিশ্চিতভাবেই এই program আগেরটির চেয়ে তাড়াতাড়ি ১০০ জন শিক্ষার্থীর ফলাফল দেখানোর কাজ শেষ করবে! কেমন মজার বিষয় না!

```
if (number >= 80)                  // যদি তারকা নম্বর
{
```

```
    cout << "pass" << endl;       //  পাশ ফলন
    cout << "star" << endl;       //  তারকা ফলন
  }
  else if (number >= 50)          //  যদি পাশের নম্বর
    cout << "pass" << endl;       //  পাশ ফলন
  else                            //  না হলে
    cout << "fail" << endl;       //  ফেল ফলন
```

তুমি হয়তো ভাবছো দেখি আরেক ভাবে করা যায় কিনা যাতে আরো কম সময় লাগে। যেমন ধরো তুমি প্রথমে ৮০ বা বেশী কিনা পরীক্ষা করবে, তারপর ৫০ এর বেশী কিনা পরীক্ষা করবে, অর্থাৎ উপরের programএর (ক্রমলেখ) মতো করে। এখানে খেয়াল করো তারকা পাওয়া শিক্ষার্থীদের জন্য শর্ত পরীক্ষা করা লাগবে ১বার সেটি number >= 80 আর স্রেফ পাশ বা ফেল করা শিক্ষার্থীদের জন্য ২টি শর্তই পরীক্ষা করা লাগবে। ফলে মোট শর্ত পরীক্ষণ হবে ২০*১ + (৭০+১০)*২ = ১৮০ বার। সুতরাং উপরের এই তৃতীয় program আমাদের লেখা প্রথম programএর চেয়ে একটু দ্রুতগতির হলেও দ্বিতীয়টির চেয়ে যথেষ্ট ধীরগতির হবে। তুমি আরো নানান ভাবে চেষ্টা করে দেখতে পারো, তবে আমাদের দ্বিতীয় programটিই সবচেয়ে দ্রুতগতির হবে, কারণ এতে সবচেয়ে কম সংখ্যক বার শর্ত পরীক্ষা করতে হয়েছে।

আচ্ছা তুমি কী ধরতে পেরেছো কেন দ্বিতীয় programটিতে সবচেয়ে কম সংখ্যক বার শর্ত পরীক্ষা করতে হবে? উত্তরটা কিন্তু খুবই সহজ। আমাদের দেখতে হবে সবচেয়ে বেশী সংখ্যক শিক্ষার্থী কোন ভাগে পড়ে। এক্ষেত্রে স্রেফ পাশ করে সর্বোচ্চ ৭০ জন। আমরা চাইবো এই ৭০ জনের জন্য output (ফলন) যাতে কম সংখ্যক, এক্ষেত্রে মাত্র একটা শর্ত পরীক্ষা করেই দিতে পারি। উল্টা দিকে যে ভাগে শিক্ষার্থীর সংখ্যা যত কম তার জন্য তত বেশী শর্ত পরীক্ষা করা যেতে পারে। আমাদের তৃতীয় programএ আমরা আসলে এই নিয়ম ভঙ্গ করেছি। কারণ এটাতে তারকা পাওয়া ২০ জনের ফলন আমরা দেখাই মাত্র ১বার শর্ত পরীক্ষা করে, আর পাশ করা ৭০জনের output দেখাই ২বার শর্ত পরীক্ষা করে। আর সে কারণে এটি দ্বিতীয় output থেকে ধীরগতির হবে। তো এখন থেকে if-else নিয়ে কাজ করার সময় শর্তদিয়ে সৃষ্টি হওয়া ডাল-পালাগুলোর কোনটাতে কতগুলো case (ব্যাপার) আসতে পারে সেটা মাথায় রেখে দক্ষ program তৈরী করবে, কেমন!

১৫.১৬ Ternary Operator (তিনিক অণুক্রিয়া)

সিপিপিতে conditional programmingএ (শর্তালী পরিগণনা) ternary operator (তিনিক অণুক্রিয়া) কী? উদাহরণসহ ternary operatorটির ব্যবহার দেখাও।

সিপিপি ভাষায় ? : এই প্রতীক দুটিকে একসাথে ব্যবহার করে ternary operatorটি (তিনিক অণুক্রিয়া) পাওয়া যায়। Ternary operatorটি if-then-else (যদি-তাহলে-নাহলে) কাজ করে, তবে দুটোর মধ্যে তফাৎ হলো ternary operator একটি expressionএর (রাশি) অংশ হিসাবে থাকে, ফলে এর একটা ফলাফল তৈরী হবে। আর if-else একটা conditional statement (শর্তযুক্ত বিবৃতি) তৈরী করে যার কোন ফলাফল নেই।

```
int first, second;     // চলকদুটির মান যোগান নিতে পারো

int large = first > second ? first : second;
```

Ternary operator ব্যবহার করে আমরা উপরে দুটো সংখ্যার বড়টি বের করার program দেখিয়েছি। এখানে প্রথমে প্রশ্ন ? চিহ্নের আগে যে শর্ত পরীক্ষা আছে সেটি মূল্যায়ন হবে। শর্ত যদি

সত্য হয় তাহলে question প্রশ্ন ? আর colon দোঁটা : চিহ্নের মাঝে যে মানটি আছে সেটি হবে operatorটির ফলাফল আর শর্ত যদি মিথ্যা হয় তাহলে operatorটির ফলাফল হবে colon (দোঁটা) : চিহ্নের পরে থাকা অংশটুকু। তাহলে উপরের programএ first > second শর্তটি সত্য হলে ফলাফল হবে first অর্থাৎ বড়টি আর শর্তটি মিথ্যা হলে ফলাফল হবে second কারন এটিই তখন বড় অন্যটির চেয়ে। সুতরাং আমরা ফলাফল হিসাবে first ও second varaible-দুটির মধ্যে সবসময় বড়টিই পাচ্ছি। তুমি নিশ্চয় এখন দুটো সংখ্যার মধ্যে ছোটটি বের করার program এভাবে লিখতে পারবে!

```
int first, second;      // চলকদুটির মান যোগান নিতে পারো
int large;              // বড় মানটি রাখার জন্য চলক ঘোষণা

first > second ? large = first : large = second;
```

তুমি কিন্তু চাইলে দুটো সংখ্যার বড়টি বের করার জন্য উপরের মতো করেও লিখতে পারতে। এইক্ষেত্রে variable (চলক) largeএ মান assign (আরোপণ) আমরা ternary operatorএর ভিতরেই করেছি খেয়াল করো। Assign (আরোপণ) operatorএর ফলাফল তো assign করা মানটিই হয়, সুতরাং এক্ষেত্রেও ternary operatorএর ফলাফল হিসাবে আমরা বড়টিই পাবো, যদিও large variable মান assign আগেই হয়ে গিয়েছে। তুমি জিজ্ঞেস করতে পারো এই ক্ষেত্রে ternary operatorটির যেটা ফলাফল আসবে সেটা আসলে কী কাজে লাগবে। এইখানে আসলে আমরা ফলাফলটি কাজে লাগাচ্ছি না। কিন্তু তুমি চাইলে int result = first > second ? large = first : large = second; লিখতেই পারো। সেক্ষেত্রে বড় মানটি large চলকের মধ্যে যেমন থাকবে তেমনি result varaibleএর মধ্যেও থাকবে। Ternary operatorএর ব্যবহার এভাবে বেশ সংক্ষিপ্ত।

```
int first, second;      // চলকদুটির মান যোগান নিতে পারো
int large;              // বড় মানটি রাখার জন্য চলক ঘোষণা

if (first > second)     // প্রথমটি বড় হলে
  large = first;
else
  large = second;       // আর তা না হলে
```

Ternary operatorএর কাজ তো উপরের মতো করে if-else দিয়েও করা যেতে পারে। তাহলে কখন তুমি ternary operator ব্যবহার করবে কখন if-else ব্যবহার করবে? অত্যন্ত সংক্ষিপ্ত ধরনের বলে ternary operator (তিনিক অণুক্রিয়া) আসলে টুকটাক ছোটখাট কিছুর জন্য বেশী ব্যবহার করা হয়। আর if-else হলো একদম সব জায়গায় ব্যবহার করার জন্য, বিশেষ করে শর্ত সত্য বা মিথ্যা হলে যদি একটা block (মহল্লা) execute (নির্বাহ) করতে হয়।

```
int first, second, third; // মান যোগান নিতে হবে

int large = first > second ? first : second;
large = large > third ? large : third;
```

তুমি কি ternary operator (তিনিক অণুক্রিয়া) ব্যবহার করে তিনটি সংখ্যার মধ্যে সবচেয়ে বড়টি বের করতে পারবে। নিশ্চয় পারবে, এ আর এমন কঠিন কী? উপরের programএর মতো করে প্রথমে দুটোর মধ্যে বড়টি বের করবে। তারপর large এর সাথে thirdটি তুলনা করে যদি

largeটিই বড় হয় তাহলে ফলাফল large আর যদি thirdটি বড় হয় তাহলে ফলাফল third টি। কিন্তু আমরা আসলে এই রকম আলাদা দুটো ternary operator চাচ্ছি না। আমরা বরং একটা ternary operatorকে আরেকটি ternary operatorএর মধ্যে ঢুকিয়ে দিবো, আর যাকে বলব nested ternary operator (অন্তান্তি তিনিক অণুক্রিয়া)। নীচের program খেয়াল করো, আমরা একটু indentation (ছাড়ন) দিয়ে লিখেছি। প্রথমে first ও second তুলনা করা হয়েছে। শর্ত সত্য হওয়া মানে first বড় যেটিকে third এর সাথে তুলনা করা হয়েছে। আর শর্ত মিথ্যা হওয়া মানে second বড়, কাজেই এটিকে third এর সাথে তুলনা করা হয়েছে। Ternary operator ব্যবহার করেই আরো নানান ভাবে এটি করা সম্ভব, তুমি নিজে নিজে চেষ্টা করে দেখো।

```
int first, second, third; // মান যোগান নিতে হবে

int large = first > second ?
      (first > third ? first : third) :
      (second > third ? second : third) ;
```

১৫.১৭ Switch Cases (পল্টি ব্যাপার)

এমন একটি program (ক্রমলেখ) রচনা করো যেটি একটি পূর্ণক (integer) আর একটি ভগ্নক (fractioner) input (যোগান) নিবে। পূর্ণকটি ১ হলে programটি পরের সংখ্যাটিকে কোণের পরিমান রেডিয়ানে ধরে নিয়ে তার sine (লম্বানুপাত) output দিবে। আর পূর্ণকটি ২ হলে cosine (লগ্নানুপাত), ৩ হলে tangent (স্পর্শানুপাত) output দিবে। তবে এই তিনটির কোনটিই না হলে বলবে "unsupoorted choice"। এই programটিতে তুমি switch case (পল্টি ব্যাপার) ব্যবহার করবে আর if-elseএর ব্যবহারের সাথে কী তফাৎ হয় সেটাও আলোচনা করবে।

ফিরিস্তি ১৫.৫: Trigonometry with Menu (প্রাপণ্য সহ ত্রিকোণমিতি)

```
int ratio;     // কোন অনুপাত sine, cosine, tangent
float angle;    // কোণের পরিমান রেডিয়ানে

// প্রথমে প্রাপণ্য (menu) দেখানো হবে
cout << "ratio 1: sine" << endl;
cout << "ratio 2: cosine" << endl;
cout << "ratio 3: tangent" << endl
cout << endl;

// তারপর অনুপাত ও কোণ যোগান নেয়া হবে
cout << "ratio: " << endl;     //  যোগান যাচনা
cin >> ratio;                  //  যোগান নেওয়া
cout << "angle: " << endl;     //  যোগান যাচনা
cin >> angle;                  //  যোগান নেওয়া

// পল্টি ব্যাপার ব্যবহার করে ফলন দেখানো হবে
switch(ratio) // এখানে চলক না হয়ে কোন রাশিও হতে পারে
{
```

```
    case 1:      //  লম্বানুপাত (sine) cmath শিরনথি লাগবে
      cout << "sine: " << sin(angle) << endl;
      break;
    case 2:      //  লগ্নানুপাত (cosine) #include <cmath>
      cout << "cosine: " << cos(angle) << endl;
      break;
    case 3:      // স্পর্শানুপাত (tangent) cmath শিরনথি লাগবে
      cout << "tangent: " << tan(angle) << endl;
      break;
    default:     // অগত্যা ত্রুটি বার্তা (error)
      cout << "unsupported ratio" << endl;
      break;
  }

  cout << "how lovely!" << endl;  // পল্টির বাইরে অন্য কিছু
```

উপরের program (ক্রমলেখ) খেয়াল করো। যেমন বলা হয়েছে তেমন করে দুটি variable নেয়া হয়েছে: কোন অনুপাত তা রাখার জন্য variable ratio আর কত রেডিয়ান কোন তা রাখার জন্য variable angle। এরপর একটা menu (প্রাপণ্য) দেখানো হয়েছে, কোন সংখ্যা দিয়ে কোন অনুপাত বুঝানো হচ্ছে সেটা ব্যবহারকারীকে জানানোর জন্য: 1 দিলে sine (লম্বানুপাত), 2 দিলে cosine (লগ্নানুপাত), 3 দিলে tangent (স্পর্শানুপাত)। এরপরে অনুপাত ও কোণ input (যোগান) নেয়ার জন্য প্রথমে input prompt (যোগান যাচনা) করে তারপর input নেওয়া হয়েছে। তারপর মুল অংশ যেখানে switch case (পল্টি ব্যাপার) ব্যবহার করে যে অনুপাত চাওয়া হয়েছে সেটি দেখানো হবে। Switch-caseএর পরে আছে অন্য কিছু programএর বাঁকী অংশ।

আমরা কেবল switch-case (পল্টি ব্যাপার) অংশে নজর দেই। যেহেতু ratio variableটির (চলক) মান ওপর নির্ভর করবে আমরা কোন অনুপাত outputএ (ফলন) দেখাবো, আমরা তাই লিখেছি switch(ratio) আর তারপর আমাদের একটি block (মহল্লা) তৈরী করতে হবে { } বাঁকা বন্ধনী (curly brackets) যুগল দিয়ে। এবার অনুপাতের মান কত হলে কী করতে হবে তার সবকিছু আমরা রাখবো blockএর ভিতরে। খেয়াল করো ratio এর মান 1, 2, 3 হওয়ার জন্য আমাদের তিনটি case (ব্যাপার) আছে যেমন case 1: case 2: case 3:; খেয়াল করো প্রথমে case তারপরে ratio variableটির কোন মান সেটি তারপর একটা : colon (দোঁটা)। প্রতিটি caseএর (ব্যাপার) পরে দেখো আমরা cout দিয়ে ত্রিকোনমিতির অনুপাত sine, cosine, tangent ব্যবহার করে output দেখিয়েছি। তারপর লিখেছি break; অর্থাৎ এইখানে switch-caseএর break (ক্ষান্তি) ঘটবে। এই breakএর (ক্ষান্তি) কাজ আমরা একটু পরেই আলোচনা করছি। তার আগে দেখো case 3: এর breakএর (ক্ষান্তি) পরে রয়েছে default: যেটি হলো deafult case (অগত্যা ব্যাপার) অর্থাৎ ওপরের কোন case এর সাথেই ratio এর মান না মিললে default caseটি ঘটবে বলে ধরে নেয়া হবে। তাহলে ratio এর মান যদি 1, 2, 3 ভিন্ন অন্য কিছু হয় তাহলে default: default caseটি ঘটবে। যথারীতি সেখানে আমরা error message (ত্রুটিবার্তা) দেখিয়েছি। এখানে কিন্তু break; আছে শেষে।

Program execution (ক্রমলেখ নির্বাহ) করার সময় ধরে নিতে পারো অদৃশ্য বোতামের মতো একটা ব্যাপার আছে যেটাকে বলা হয় control (নিয়ন্ত্রণ)। এই control (নিয়ন্ত্রণ) বোতামটি program executionএর শুরুতে main functionএর একদম প্রথম সারিতে থাকে। বোতামটি যেই সারিতে থাকে সেই সারি execute (নির্বাহ) হয়। আর তারপর control বোতামটি পরের সারিতে লাফ দেয়, তখন সেই সারিটি execute হয়। এভাবে control বোতামের লাফালা-

ফি ও সেই সাথে সংশ্লিষ্ট সারির নির্বাহ একে একে চলতে থাকে। If-else (যদি-নাহলে) আলোচনা করার সময় আমরা বলেছিলাম শর্ত সত্য হলে কিছু কাজ হয় আবার শর্ত মিথ্যা হলে অন্য কিছু কাজ হয়। ঠিক যেন দুটো শাখা (branch) তৈরী হয়। শর্তের ওপর নির্ভর করে control বোতামটি আসলে হয় এই শাখায় নাহয় ওই শাখায় গিয়ে লাফ দিয়ে বসে। Control যে শাখায় বসে সেই শাখা নির্বাহিত হয়, অন্য শাখা নির্বাহিত হয় না। Control (নিয়ন্ত্রণ) বোতাম এরপর if-elseএর (যদি নাহলে) পরের অংশে চলে যায়।

Switch-caseএর (পল্টি-ব্যাপার) ক্ষেত্রে বলতো control switch(ratio) এর পরে লাফ দিয়ে কোন সারিতে গিয়ে বসবে? যদি ratio এর মান হয় 1 তাহলে গিয়ে বসবে case 1: এর সারিতে, 2 হলে গিয়ে বসবে case 2: এর সারিতে, আর 3 হলে বসবে case 3: এর সারিতে, আর তিনটের কোনটাই না হলে গিয়ে বসবে default: এর সারিতে। Control switch(ratio) হতে লাফ দিয়ে গিয়ে সংশ্লিষ্ট caseএ (ব্যাপার) বসার পরে সারির পর সারি একে একে যেতে থাকবে যতক্ষণনা একটি break; (ক্ষান্তি) পাচ্ছে। অর্থাৎ break (ক্ষান্তি) পাওয়ার আগে পর্যন্ত প্রত্যকটি সারিই একের পর এক execute (নির্বাহ) হতে থাকবে। আর break; পাওয়ার পরেই control আর একটি লাফ দিয়ে switch-caseএর (পল্টি-ব্যাপার) blockএর বাইরে চলে যাবে। Break না দিলে কী ঘটবে আমরা সেটা পরবর্তীতে আলোচনা করবো। তবে বলে রাখি প্রতিটি caseএর (ব্যাপার) শেষে আসলে break (ক্ষান্তি) দেয়াটা আসলেই খুব গুরুত্বপূর্ণ, আর আমরা আবার না দেয়ার ভুলটা প্রায়ই করি।

১৫.১৮ Nested Switch Cases (অন্তান্তি পল্টি ব্যাপার)

Nested switch case (অন্তান্তি পল্টি ব্যাপার) ব্যবহার করে এমন একটি program (ক্রমলেখ) রচনা করো, যেটি প্রথমে menu (প্রাপণ্য) দেখিয়ে জানতে চাবে আমরা বর্গের হিসাব করতে চাই, নাকি বৃত্তের হিসাব করতে চাই। সেটি input (যোগান) নেবার পরে আমাদের পছন্দ বর্গ হলে programটি input নিবে দৈর্ঘ্য আর কী দেখতে চাই ক্ষেত্রফল নাকি পরিসীমা তা, আর সেই অনুযায়ী output (ফলন) দেখাবে। আর আমাদের পছন্দ বৃত্ত হলে ক্রমলেখটি ব্যাসার্ধ input নিবে আর নিবে ক্ষেত্রফল নাকি পরিধি দেখতে চাই তা, আর সে অনুযায়ী output দিবে।

নীচের program (ক্রমলেখ) খেয়াল করো। প্রথমে আকৃতির menu (প্রাপণ্য) দেখানো হয়েছে। তারপর shape variable declare (চলক ঘোষণা) করে input prompt (যোগান যাচনা) করে input (যোগান) নেয়া হয়েছে। এরপর shape variableএর মানের ওপর switch (পল্টি) যাতে তিনটি case (ব্যাপার) আছে। Variable shape এর মান 1 হলে case 1: বর্গ, 2 হলে case 2: বৃত্ত, আর অন্য কিছু হলে অগত্যা ব্যাপারে default: ত্রুটিবার্তা দেখানো হয়েছে।

ফিরিস্তি ১৫.৬: Menu with Nested Switch (অন্তান্তি পল্টি দিয়ে প্রাপণ্য)

```
// আকৃতির প্রাপণ্য (menu)
cout << "shape 1 square" << endl;
cout << "shape 2 circle" << endl;

int shape;          // চলক ঘোষণা
cout << "shape: ";  // যোগান যাচনা
cin >> shape;       // যোগান নেওয়া

// বাইরের পল্টি যার ভিতরে আবার পল্টি থাকবে
switch(shape)       // আকৃতির পল্টি
```

```
{
  case 1:      // বাইরের পল্টি বর্গ হলে
    //  কী পছন্দ তা দেখানো হবে
    cout << "choice square" << endl;

    // বর্গের দৈর্ঘ্য যোগান নিতে হবে
    int length;          // চলক ঘোষণা
    cout << "length: "; // যোগান যাচনা
    cin >> length;         // যোগান নেওয়া

    // কী চাই তার প্রাপণ্য (menu)
    cout << "1 output area" << endl;
    cout << "2 output perimeter" << endl;

    int soutput;           // চলক ঘোষণা
    cout << "output: ";    // যোগান যাচনা
    cin >> soutput;        // যোগান নেওয়া

    // ভিতরের পল্টি যেটি আরেকটি পল্টির ভিতরে
    switch(soutput)        // পল্টি কী চাই
    {
      case 1:      // ভিতরের পল্টি ক্ষেত্রফল হলে
        cout << "area: ";
        cout << length * length;
        cout << endl;
        break;

      case 2:      // ভিতরের পল্টি পরিসীমা হলে
        cout << "perimeter: ";
        cout << 4*length;
        cout << endl;
        break;

      default:     // ভিতরের পল্টি অন্য কিছু হলে ত্রুটিবার্তা
        cout << "unsupported choice" << endl;
        break;
    }
    //  এটি ভিতরের পল্টি থেকে বাইরে
    cout << "square output ends" << endl;
    break;

  case 2:      // ভিতরের পল্টি বৃত্ত হলে
    //  কী পছন্দ তা দেখানো হবে
```

```
    cout << "choice circle" << endl;

    // বৃত্তের ব্যাসার্ধ যোগান নিতে হবে
    int radius;         // চলক ঘোষণা
    cout << "radius: ";
    cin >> radius;

    // কী চাই প্রাপণ্য
    cout << "1 output area" << endl;
    cout << "2 output perimeter" << endl;

    int coutput;            // চলক ঘোষণা
    cout << "output: ";     // যোগান যাচনা
    cin >> coutput;         // যোগান নেওয়া

    // ভিতরের পল্টি যেটি আরেকটি পল্টির ভিতরে
    switch(coutput)         // কী চাই পল্টি
    {
      case 1:       // ভিতরের পল্টি ক্ষেত্রফল হলে
        cout << "area: ";
        cout << 3.1416 * radius * radius;
        cout << endl;
        break;

      case 2:       // ভিতরের পল্টি পরিধি হলে
        cout << "perimeter: ";
        cout << 2 * 3.1416 * radius;
        cout << endl;
        break;

      default:      // ভিতরের পল্টি অন্য কিছু হলে ত্রুটিবার্তা
        cout << "unsupported choice" << endl;
        break;
    }
    //  এটি ভিতরের পল্টি থেকে বাইরে
    cout << "circle output ends" << endl;
    break;

  default:        // বাইরের পল্টি অন্য কিছু হলে ত্রুটিবার্তা
    cout << "unsupported choice" << endl;
    break;
}
```

```
//    বাইরের পল্টিরও বাইরে
cout << "how lovely!" << endl;
```

যখন shape এর মান 1 অর্থাৎ বর্গ বেছে নেয়া হয়েছে তখন প্রথমে outputএ (ফলন) দেখানো হয়েছে যে বর্গ পছন্দ করা হয়েছে। তারপর variable length declare (ঘোষনা) করে input prompt (যোগান যাচনা) করে input (যোগান) নেওয়া হয়েছে। তারপর squareএর কী জানতে চাই তার জন্য আরেকটি menu (প্রাপণ্য) দেখানো হয়েছে, যেখানে ক্ষেত্রফল নাকি পরিসীমা চাই সেটা দেখানো হয়েছে। ব্যবহারকারীর পছন্দ যোগান নেয়ার জন্য এখানেও soutput নামে একটি variable declare করে input prompt করে মান input নেয়া হয়েছে। তারপর variable keechai এর মানের ওপর নির্ভর করে আরেকটি switch case (পল্টি ব্যাপার) ব্যবহার করে ক্ষেত্রফল বা পরিসীমা outputএ (ফলন) দেখানো হয়েছে। এই switch caseটি (পল্টি ব্যাপার) আগের switch caseএর ভিতরে, আর তাই এই ভিতরেরটিকে বলা হবে nested switch case (অন্তান্তি পল্টি ব্যাপার)।

যখন shape এর মান 2 অর্থাৎ বৃত্ত বেছে নেয়া হয়েছে তখন প্রথমে outputএ (ফলন) দেখানো হয়েছে যে বৃত্ত পছন্দ করা হয়েছে। তারপর variable radius declare (ঘোষনা) করে input prompt (যোগান যাচনা) করে input (যোগান) নেওয়া হয়েছে। তারপর বৃত্তের কী জানতে চাই তার জন্য আরেকটি menu (প্রাপণ্য) দেখানো হয়েছে, যেখানে ক্ষেত্রফল নাকি পরিধি চাই সেটা দেখানো হয়েছে। ব্যবহারকারীর পছন্দ input নেয়ার জন্য এখানেও coutput নামে একটি variable declare করে input prompt করে মান input নেয়া হয়েছে। বর্গের ক্ষেত্রে ব্যবহৃত variable soutput থেকে ভিন্ন একটি নাম নেয়ার জন্যই মুলত নাম দেওয়া হয়েছে coutput। এই দুটো variableই বাইরের switch-caseএর যে block (মহল্লা) তার ভিতরে। একই blockএ দুটো variableএর (চলক) নাম একই হতে পারে না। আর সে কারণে নামের এই ভিন্নতা, যদিও তাদের উদ্দেশ্য এখানে একই রকম। যাইহোক, variable coutput এর মানের ওপর নির্ভর করে এরপর আরেকটি switch case ব্যবহার করে ক্ষেত্রফল বা পরিধি outputএ (ফলন) দেখানো হয়েছে। এই switch caseটি (পল্টি ব্যাপার) বর্গের switch-caseএর মতোই বাইরের switch caseটির ভিতরে, তাই এটিও একটি nested (অন্তান্তি) switch case।

এই পর্যায়ে জিজ্ঞেস করতে পারো, break; পাওয়া মাত্র control (নিয়ন্ত্রণ) সেই switch case (পল্টি ব্যাপার) থেকে বের হয়ে আসে বলে আমরা জানি, তো ভিতরের switch case থেকে break; পেলে কোথায় যাবে? উত্তর হচ্ছে ভিতরের switch case থেকে বের হয়ে যেখানে আসবে সেটা কিন্তু বাইরের switchএর block। ভিতরের switch থেকে বের হয়ে কোথায় আসবে সেটা বুঝার জন্য বর্গের switch caseএর বাইরে cout << " square output ends" << endl; আর বৃত্তের switch caseএর বাইরে cout << "circle output ends" << endl; লেখা হয়েছে। আর বাইরের switch caseএর বাইরে লেখা হয়েছে cout << "how lovely !" << endl;। মনে রাখবে break পেলে ভিতরের এক স্তরের switch থেকে বের হয়ে নিয়ন্ত্রণ ঠিক বাইরের স্তরটিতে যাবে।

১৫.১৯ Switch Cases Breaks (পল্টি ব্যাপার ক্ষান্তি)

Switch caseএ (পল্টি ব্যাপার) break (ক্ষান্তি) না দিলে কী ঘটে, আর break না দেওয়া কোথায় কাজে লাগতে পারে? যথাযথ উদাহরণ সহ program (ক্রমলেখ) লিখে দেখাও।

ধরো তোমার একজন অতিথি আসবে। সে যদি সকাল ১০ বা ১১টায় আসে তাকে তোমার সকালে নাস্তা, দুপুরের খাবার, আর বিকালের নাস্তা খাওয়াতে হবে। আর সে যদি ১২টায় বা ১৩টায় আসে তবে তাকে কেবল দুপুরের খাবার ও বিকালের নাস্তা খাওয়াতে হবে, আর তিনি যদি ১৪টা বা

১৫টায় আসে তাহলে তাকে কেবল বিকালের নাস্তা খাওয়াতে হবে। এই সময়গুলো ভিন্ন অন্য কোন সময়ে যদি সে আসে তাহলে তাকে কিছুই খাওয়ানোর দরকার নাই।

```
switch(vartime)
{
  case 10:
  case 11:
    cout << "morning breakfast" << endl;
  case 12:
  case 13:
    cout << "midday lunch" << endl;
  case 14:
  case 15:
    cout << "afternoon snacks" << endl;
}
```

উপরের programএ আমরা break (ক্ষান্তি) ছাড়া switch case (পল্টি ব্যাপার) লিখে program (ক্রমলেখ) তৈরী করেছি। এখানে variable vartime এ আমরা অতিথির আসার সময় রাখবো, সেটা input (যোগান) নেয়া হয়ে থাকতে পারে, বা কোন ভাবো assigned (আরোপিত) হয়ে থাকতে পারে। সাধারণত switchএ (পল্টি) যে ব্যাপারটার সাথে মিলে যায় সেখান থেকে statementগুলো (বিবৃতি) নির্বাহিত হতে শুরু করে আর break (ক্ষান্তি) পাওয়া পর্যন্ত চলে। আর একবার কোন ব্যাপারের সাথে মিলে গেলে পরে আর কোন ব্যাপারের সাথে মিলানোর চেষ্টা করাও হয় না, বরং break না পাওয়া পর্যন্ত ক্রমাগত statementগুলো নির্বাহিত হতে থাকে।

খেয়াল করো উপরের programএ (ক্রমলেখ) সময় যদি ১০টা হয়, ঠিক সেখানে কিছু না থাকলেও পরপর যে statementগুলো আছে সেগুলো একে একে নির্বাহিত হবে, ফলে morning breakfast, midday lunch, afternoon snacks সবগুলো একে একে outputএ আসতে থাকবে। সময় যদি ১১টা হয় তাহলেও একই ঘটনা ঘটবে। সময় যদি ১২ টা হয়, তাহলে morning breakfast outputএ আসবে না, কিন্তু midday lunch ও afternoon snacks একে একে আসতে থাকবে। পরের সময়গুলোর জন্যেও একই রকমের কথাবার্তা প্রযোজ্য।

আর একটা বিষয় খেয়াল করো, উপরের switchএ (পল্টি) আমরা default case default : দেই নাই। ফলে সময় যদি তালিকায় না থাকে তাহলে সেটি কোন caseএর (ব্যাপার) সাথেই মিলবে না, আর এতে outputএ (ফলন) কিছুই আসবে না। আসলে switchএ (পল্টি) default (অগত্যা) ব্যাপার দিতেই হবে এমন কোন কথা নেই, দরকার না লাগলে দিবে না।

```
switch(number)
{
  case 4:
  case 0:
  case 2:
    cout << "even" << endl;
    break;
  case 1:
  case 5:
  case 3:
    cout << "odd" << endl;
```

```
    break;
}
```

এবার কিছু প্রশ্ন: switchএ কী caseগুলো মানের ক্রমানুসারেই থাকতে হবে? মানগুলো কী ধারাবাহিকভাবে পরপর সংখ্যা হতে হবে? উভয় প্রশ্নের উত্তর হচ্ছে "না"। কাজেই ঠিক উপরের উদাহরণের মতো তুমি দরকার মতো caseগুলো পরপর না হলেও বা উল্টোপাল্টা ক্রমে হলেও লিখতে পারবে। আবার দেখো কিছু caseএ (ব্যাপার) break (ক্ষান্তি) নাই, আবার কিছু ব্যাপারে আছে। মোট কথা যেখানে break দেয়া দরকার সেখানে break; না দরকার হলে নাই।

আরো কিছু প্রশ্ন: switchএ (পল্টি) caseগুলো (ব্যাপার) কী পূর্ণক (integer) ছাড়া ভগ্নক (fractioner) হতে পারবে? আর switch() এ variable (চলক) ছাড়া অন্য কিছু ব্যবহার করা যাবে? তুমি কোন ভগ্নক (fractioner) case হিসাবে ব্যবহার করে দেখতে পারো, তাতে compileএ (সংকলন) error message (ত্রুটি বার্তা) দেখাবে, তার মানে হলো পারবে না। আর switch(number) এখানে switch এ যে কেবল variable হতে হবে তা নয়, যে কোন রাশি যেটি পূর্ণক ফলাফল দেয় সেটিই তুমি ব্যবহার করতে পারো, যেমন নীচের উদাহরণ দেখো, আমরা ২ দিয়ে ভাগশেষের ওপর switch ব্যবহার করছি। ভাগশেষ ০ হলো জোড়, আর ১ হলে বিজোড়।

```
switch(number % 2)
{
  case 0: cout << "even" << endl; break;
  case 1: cout << "odd" << endl; break;
}
```

Switchএ অবশ্য তুমি একই case দুইবার ব্যবহার করতে পারবে না, যেমন case 1: লিখে একই switchএর ভিতরে পরে আবার case 1: লিখতে পারবে না। তবে switchএর ভিতরে nested (অন্তান্তি) switch থাকলে সেখানে case 1: থাকতেই পারে।

১৫.২০ Swtich Cases If Else (পল্টি ব্যাপার যদি-নাহলে)

Switch case (পল্টি ব্যাপার) ব্যবহার না করে if else (যদি নাহলে) ব্যবহার করলেই তো হয়। তাহলে switch case কোথায় ব্যবহার করবো, আর কোথায় if else ব্যবহার করবো?

```
switch(number)
{
  case -2:
  case -1:
    cout << "negative" << endl;
    break;

  case 0:
    cout << "zero" << endl;
    break;

  case 1:
  case 2:
```

```
        cout << "positive" << endl;
        break;
    }
```

উপরের উদাহরণটি দেখো। এখানে আমরা একটি নম্বর positive (ধনাত্মক), negative (ঋণাত্মক), নাকি zero (শুন্য) নির্ণয় করতে চাই। আমরা যদি আগে থেকে জানি যে নম্বরটি কেবল $-2, -1, 0, 1, 2$ এই পাঁচটি নির্দিষ্ট সংখ্যার একটি হতে পারবে, অন্য আর কিছু নয়, এগুলোর বাইরে নয়, কেবল তাহলেই আমরা উপরের মতো করে switch case (পল্টি ব্যাপার) ব্যবহার করতে পারবো। আবার চাইলে আমরা নীচের মতো করে সমতুল আরেকটি programও লিখতে পারবো, যেখানে আমরা switch case ব্যবহার না করে if else (যদি নাহলে) ব্যবহার করবো। যদি না হলে ব্যবহার করে অবশ্য আরো নানা ভাবেই এটি করা সম্ভব, এটি কেবল একটা উদাহরণ।

```
if (number == -2 || number == -1)
    cout << "negative" << endl;
else if (number == 1 || number == 2)
    cout << "positive" << endl;
else // if (number == 0)
    cout << "zero" << endl;
```

কিন্তু আমাদের নম্বরটি যদি উপরের ওই পাঁচটি সংখ্যার বাইরে অনির্দিষ্ট সংখ্যক নম্বরগুলোর একটি হয়, অথবা অনেক অনেক বেশী সংখ্যকের একটি হয়, তাহলে ঠিক switch ব্যবহার করে আমরা সামলাতে পারবো না। কারণ এ সব ক্ষেত্রে number of cases (ব্যাপারের সংখ্যা) হবে অনেক বেশী বা অসংখ্য। আর একটি ব্যাপার হলো switchএ caseগুলো মুলত মান সমান == হলে কী হবে তার ওপর ভিত্তি করে তৈরী, অন্য কোন ধরনের তুলনা যেমন বড় >, ছোট < ইত্যাদি ব্যবহার করা যায় না। ফলে switch (পল্টি) সাধারণত ব্যবহার করা হয় অল্প কিছু সংখ্যক ও সুনির্দিষ্ট সংখ্যক ব্যাপারের ক্ষেত্রে, আর এ সব ক্ষেত্রে program পড়া সহজ হয়ে যায়। অন্যান্য সকল ক্ষেত্রে সাধারণত if else (যদি নাহলে) ব্যবহার করা হয় কারণ if elseএর সাথে যে কোন শর্ত বা connectives &&, ||, ! ব্যবহার করে আরো জটিল শর্ত ব্যবহার করা যায়।

```
if (number < 0)
    cout << "negative" << endl;
else if (number > 0)
    cout << "positive" << endl;
else // if (number == 0)
    cout << "zero" << endl;
```

১৫.২১ Global & Local Variables (ব্যাপীয় ও স্থানীয় চলক)

Local variable (স্থানীয় চলক) কী? এর বিপরীতে global variableই (ব্যাপীয় চলক) বা কী? Conditional programmingএ (শর্তালী পরিগণনা) local variableএর ব্যবহার দেখাও।

যখন কোন varaible বা constant বাঁকা বন্ধনী যুগলের বাইরে অর্থাৎ কোন blockএর বাইরে থাকে তখন তাকে global variable (ব্যাপীয় চলক) বা global constant (ব্যাপীয় ধ্রুবক) বলা হয়। নীচের programএ (ক্রমলেখ) খেয়াল করো pai আর lowerLimit যে কোন blockএর (মহল্লা) বাইরে, তাই এগুলো যথাক্রমে global constant (ব্যাপীয় ধ্রুবক) এবং

global variable (ব্যাপীয় চলক)। Global variable বা constant declare (ঘোষনা) করার পর থেকে programএর যে কোন জায়গায় ব্যবহার করা যায়। যে কোন constantএর (ধ্রুবক) মান তো declare করার সময় অবশ্যই দিতে হয়, global constantএর (ব্যাপীয় ধ্রুবক) মানও declareএর সময়ই দিয়ে দিতে হয়। আর global variableএর মান declare করার সময় না দিয়ে দিলে এতে default (অগত্যা) শুন্য থাকে; এইটা একটা জরুরী তথ্য।

ফিরিস্তি ১৫.৭: Using Local & Global Variables (স্থানীয় ও ব্যাপীয় চলকরে ব্যবহার)

```
#include <iostream>
#include <cstdlib>

using namespace std;

float const pai = 3.1416; // ব্যাপীয় ধ্রুবক, মান দিতেই হবে
float lowerLimit = 1.00;  // ব্যাপীয় চলক, মান না দিলে শুন্য

int main(void)
{
  float radius;           // স্থানীয় চলক
  float const two = 2.0;  // স্থানীয় ধ্রুবক

  cout << "radius: "; cin >> radius;

  if (radius < lowerLimit)
  {
    cout << "less than lower limit" << endl;
    return EXIT_FAILURE;
  }

  float perimeter = two * pai * radius; // স্থানীয় চলক
  cout << "perimeter: " << perimeter << endl;

  return EXIT_SUCCESS;
}
```

যখন কোন variable(চলক) বা constant(ধ্রুবক) কোন বাঁকা বন্ধনী যুগল {} বা blockএর (মহল্লা) ভিতরে ঘোষিত হয় তখন তাকে local variable (স্থানীয় চলক) বা local constant (স্থানীয় ধ্রুবক) বলা হয়। উপরের programএ (ক্রমলেখ) খেয়াল করো variable radius এবং constant two উভয়ই main functionএর blockএর ভিতরে ঘোষিত হয়েছে, কাজেই এ দুটো যথাক্রমে local variable ও local constant। Local variable বা constant যে কোন block (মহল্লা) বা subblockএর (উপমহল্লা) ভিতরে ঘোষিত হতে পারে। Blockএর ভিতরে আবার block থাকলে ভিতরের blockকে subblock (উপমহল্লা) বলা হয় আর বাইরের blockকে বলা হয় superblock (অধিমহল্লা)। যে কোন constantএর মান তো ঘোষণার সময়ই দিয়ে দিতে হয়, local constantএর মানও তাই declare করার সময়ই দিয়ে দিতে হবে। আর local variableএর মান দিয়ে না দিলে এটাতে উল্টা পাল্টা একটা মান থাকবে। সুতরাং local

variable ব্যবহারের পূর্বে অবশ্যই এতে value assign করে নিতে হবে। Local variable ও constant declare করার পর থেকে ওই blockএর ভিতরে যে কোন খানে ব্যবহার করা যায়, এমনকি subblock বা subsubblock ভিতরেও ব্যবহার করা যায়।

```
int myvar = 2;  // ব্যাপীয় চলক

int main()
{
  cout << myvar << endl;     // ব্যাপীয় চলকের মান 2

  int myvar = 3;             // এখন থেকে স্থানীয় চলক
  cout << myvar << endl;     // স্থানীয় চলকের মান 3
  {
    cout << myvar << endl;   // অধিমহল্লার চলক মান 3

    int myvar = 5;           // উপমহল্লার স্থানীয় চলক
    cout << myvar << endl; // উপমহল্লার স্থানীয় চলক মান 5
  }
  cout << myvar << endl;     // স্থানীয় চলকের মান 3

  // অন্যান্য কিছু এখানে থাকতে পারে, আমরা লিখছি না
}
int yourvar = myvar;     // ব্যাপীয় চলকের মান 2
```

অনেক সময় একটি local (স্থানীয়) variable বা constantএ নাম একটি global (ব্যাপীয়) variable বা constantএর নামের সাথে মিলে যেতে পারে। প্রথম কথা প্রতিটি varaible বা constantএর নাম পুরো program জুড়ে unique (একক) হওয়া উচিৎ, কিন্তু সুবিধার বিচারে অনেক সময় সেটা করা সম্ভব হয় না। এমতাবস্থায় কী করে বুঝবো ব্যবহৃত variable বা constantটি global না local? উপরের program (ক্রমলেখ) খেয়াল করো, সেখানে myvar নাম বারবার ব্যবহার করে অনেগুলো variable declare করা হয়েছে, যার একটি সকল blockএর (মহল্লা) বাইরে তাই global (ব্যাপীয়) আর অন্যগুলো কোন না কোন blockএর ভিতরে তাই local variable। এখন myvar নামের variableকে নানান খানে outputএ (ফলন) দেখানো হয়েছে। কথা হচ্ছে নাম যেহেতু একই, তো আমরা নামটি দিয়ে কখন কোন variableটিকে বুঝবো, কখন কোন মানই বা outputএ দেখতে পাবো?

খেয়াল করে দেখো যেখানে global variableটি declare করা হয়েছে আর মান দেওয়া হয়েছে 2 তারপর থেকে এটির কার্যকারীতা বলবৎ আছে, blockএর বাইরে তো অবশ্যই আছে যেমন একদম নীচে যেখানে int yourvar = myvar; লেখা হয়েছে। আবার blockএর (মহল্লা) ভিতরে local variable ঘোষণার আগে পর্যন্ত এটির কার্যকারীতা রয়েছে ফলে আমরা global variableটির মানটিই অর্থাৎ 2ই দেখতে পাবো। তারপর blockএর ভিতরে যখন একই নাম দিয়ে একটি variable ঘোষনা করা হয়েছে আর মান দেওয়া হয়েছে 3, তখন myvar নামের সাথে local এই variableটির কার্যকারীতা বলবৎ হয়েছে, আর তা জারি আছে block শেষ হওয়া পর্যন্ত, তাছাড়া subblockএর ভিতরে একই নামের আরেকটি variable ঘোষণার আগে পর্যন্তও তা জারি আছে। Programএ (ক্রমলেখ) commentগুলো (টীকা) খেয়াল করো। কোথায় কোন মান outputএ আসবে তা দিয়ে আমরা বুঝার চেষ্টা করছি, কোথায় কোন variableটির

কার্যকারীতা বলবৎ আছে। তাহলে কোন নাম কোন variableটিকে বুঝাই, সেটার জন্য আমাদের দেখতে হবে একই blockএর ভিতরে ওই নামের কোন variable আছে কিনা? যদি থাকে সেই variableটি কার্যকর আছে। আর একই blockএর ভিতরে যদি না থাকে, তাহলে আমরা ঠিক বাইরের blockটি দেখবো, সেখানে একই নামে কোন variable আছে কিনা? যদি থাকে সেটা বলবৎ হবে আর তাও না থাকলে তার ঠিক বাইরে আরো কোন block আছে কিনা তা দেখবো।

```
int number;
cin >> number ;

// নীচের remainder হলো স্থানীয় চলক
if (int remainder = number % 3)
  cout << "nil remainder" << endl;
else
  cout << "remainder " << remainder << endl;
```

সিপিপিতে if else (যদি নাহলে) লেখার সময় যদি { } বাঁকা বন্ধনী যুগল ব্যবহার করে কোন block (মহল্লা) তৈরী করা হয়, তাহলে সেই blockএর ভিতরে ঘোষিত যে কোন variable বা constant তো local variable বা constant হবে। আমরা সেটা আর আলাদা করে দেখাতে চাই না। তবে উপরের program খেয়াল করো if (int remainder = number % 3) লিখেও আমরা remainder নামে একটি variable ঘোষণা করেছি। এই remainder নামের variableও একটি local variable (স্থানীয় চলক) হিসাবে পরিগণিত হয়, আর এটা কার্যকর থাকে কেবল যেখানে লেখা হয়েছে সেখান থেকে শুরু হয়ে ওই if else ladder (মই) বা nesting (অন্তান্তি) যতক্ষণ শেষ না হচ্ছে ততক্ষণ পর্যন্ত, এর বাইরে কোন কার্যকারীতা থাকবে না, ফলে ব্যবহার করলে error message (ত্রুটি বার্তা) পাবে।

If else (যদি নাহলে) এর ক্ষেত্রে ঘোষিত local (স্থানীয়) variableটির মতো আমরা switch caseএর (পল্টি ব্যাপার) ক্ষেত্রেও একই ভাবে local variable ঘোষণা করতে পারি। নীচের programএ খেয়াল করো switch (int remainder = number % 3) লিখে আমরা একটি local variable remainder ঘোষণা করেছি। এই variableটির কার্যকারীতাও কেবল ওই switch blockএর ভিতরেই। বাইরে কোথাও এই variableটিকে ব্যবহার করবার জো নেই। তুমি কিন্তু switch blockটির ভিতরে চাইলে আরো local variable (স্থানীয় চলক) ঘোষণা ও ব্যবহার করতেই পারতে। Constantএর ক্ষেত্রেও একইরকম আলোচনা প্রযোজ্য।

```
int number;
cin >> number;

// নীচের সারিতে remainder স্থানীয় চলক
switch(int remainder = number % 3)
{
  case 0:
    cout << "zero " << remainder << endl;
    break;
  case 1:
    cout << "one " << remainder << endl;
    break;
  case 2:
```

```
        cout << "two " << remainder << endl;
        break;
    }
```

১৫.২২ Exercise Problems (অনুশীলনী সমস্যা)

Conceptual Questions: নীচে কিছু conceptual প্রশ্ন রয়েছে। প্রশ্নগুলোর উত্তর নিজে নিজে বের করবে।

১. Conditional programming (শর্তালি পরিগণনা) কী? অল্প কথায় আলোচনা করো।

২. যদি if এর সাথের শর্ত মিথ্যা হলে সংশ্লিষ্ট নাহলেতে else গিয়ে আবারও শর্তের বিপরীত শর্তটি সত্য কিনা পরীক্ষা করা দরকার নেই। ব্যাখ্যা করো।

৩. যদি নাহলে (if else) দিয়ে program (ক্রমলেখ) লিখতে indentation (ছাড়ন) দেয়া গুরুত্বপূর্ণ কেন? কার জন্য গুরুত্বপূর্ণ মানুষের জন্য নাকি computerএর (গণনি) জন্য?

৪. Relational operators (অন্বয়ী অণুক্রিয়া) কী? এগুলো কী ধরনের ফলাফল দেয়? সি-পিপিতে থাকা কয়েকটি relational operatorএর উদাহরণ দাও।

৫. If else ladderএ (যদি নাহলে মই) শর্তগুলো কী ভাবে সাজাবে, যদি চিন্তার সুবিধা বিবেচনা করো অথবা programএর দক্ষতা বিবেচনা করো?

৬. Nested if else (অন্তান্তি যদি নাহলে) ও if else ladder (যদি নাহলে মই) একটা থেকে আরেকটিতে রূপান্তর সম্ভব, উদাহরণ সহ ব্যাখ্যা করো।

৭. Dangling else (ঝুলন্ত নাহলে) সমস্যাটি কী? এটির সমাধান কী কী ভাবে করা যেতে পারে, উদাহরণ দিয়ে আলোচনা করো।

৮. Empty statement (শুন্য বিবৃতি) কী? কত ভাবে empty statement দেওয়া যায়?

৯. Boolean connectives (বুলক সংযোজক) কী কী, কী ভাবে ফলাফল দেয়?

১০. পূর্ণক (integer) ও ভগ্নক (fractioner) কে সরাসরি Boolean হিসাবে কী ভাবে ব্যবহার করা যায় আলোচনা করো। এতে কী সুবিধা হয়?

১১. Boolean conditionএর (বুলক শর্ত) partial evaluation কী ভাবে কাজ করে?

১২. একাধিক global variable (ব্যাপীয় চলক) ও local variableএর (স্থানীয় চলক) নাম একই হলে কোনটা কার্যকর তা কী ভাবে নির্ধারিত হয়?

১৫.২২. Exercise Problems (অনুশীলনী সমস্যা)

Programming Problems: নীচে আমরা কিছু programming সমস্যা দেখবো। এই সমস্যাগুলো আগে ধৈর্য্য ধরে নিজে নিজে সমাধান করতে চেষ্টা করবে। যখন একেবারেই পারছো না বলে মনে হয় তখনই কেবল সমাধান দেখে নিতে পারো। সমাধানগুলো programmingএর প্রশ্নগুলোর শেষে আছে।

১. নীচের programএর (ক্রমলেখ) output (ফলন) কী তা প্রথমে খাতা কলমে নির্ণয় করো, আর তারপর গণনিতে চালিয়ে তার সাথে মিলাও।

```
int n;  // আদি মান আরোপ করা হয় নি
cout << (n = 4) << endl;
cout << (n == 4) << endl;
cout << (n > 3) << endl;
cout << (n < 4) << endl;
cout << (n = 0) << endl;
cout << (n == 0) << endl;
cout << (n > 0) << endl;
cout << (n && 4) << endl;
cout << (n || 4) << endl;
cout << (!n) << endl;
```

২. নীচের programএ (ক্রমলেখ) কিছু syntactical (গঠনগত) ভুল আছে। ভুলটা কোথায় বলে তুমি মনে করো? ভুলটা এমন ভাবে ঠিক করো যাতে এটির indentation (ছাড়ন) দেখে যা করতে চাওয়া হয়েছিল বলে মনে হয়, programটি (ক্রমলেখ) semantically (অর্থবোধকতায়) যেন একদম তাই করে।

```
if (x >= y)
  sum += x;
  cout << "x large" << endl;
else
  sum += y;
  cout << "y large" << endl;
```

৩. নীচের program (ক্রমলেখ) চালালে কী output (ফলন) পাওয়া যাবে?

```
int n, k = 5;
n = (100 % k ? k + 1 : k - 1);
cout << "n = " << n << " k = " << k << endl;
```

৪. নীচের program (ক্রমলেখ) চালালে কী output (ফলন) পাওয়া যাবে?

```
int found = 0, count = 5;
if (!found || ++ count == 0)
  cout << "danger" << endl;
cout << "count = " << count << endl;
```

৫. নীচের programএ, সম্ভবত বলা যায় যে conditional statementএর (শর্তালি বিবৃতি) একদম প্রথম সারিতেই একটা ভুল আছে। Programটি যে ভাবে লেখা আছে সেরকম অবস্থায়ই যদি execute (নির্বাহ) করা হয় তাহলে output (ফলন) কী হবে? আর যেটা করতে চাওয়া হয়েছিল বলে মনে হয় যদি সেটা করা হয় তাহলে output কী হবে?

```
int n = 5;
if (n = 0) // অণুক্রিয়াটি খেয়াল করো
  cout << "n is zero." << endl;
else
  cout << "n is not zero" << endl;
cout << "square of n " << n * n << endl;
```

৬. নীচের conditional statementএ (শর্তালি বিবৃতি) তে অনেক অপ্রয়োজনীয় শর্ত আছে। অপ্রয়োজনীয় শর্তগুলো বাদ দিয়ে conditional statementটি আবার লেখো।

```
float income;
cout << "monthly income: ";
cin >> income;

if (income < 0)
  cout << "loan will increase." << endl;
else if (income >= 0 && income < 1200)
  cout << "below poverty limit." << endl;
else if (income >= 1200 && income < 2500)
  cout << "slightly well-off." << endl;
else if (income >= 2500)
  cout << "sufficiently well-off." << endl;
```

৭. যদি ভিন্ন ভিন্ন বার চালানোর সময় ০, ১৫, বা ৭ input (যোগান) দেয়া হয় তাহলে নীচের programএর (ক্রমলেখ) output (ফলন) কোন বারে কী হবে । কত input দিলে outputএ "out of range!" আসবে?

```
int n;
cout << "number is: ";
cin >> n;

if (n < 10)
  cout << "less than 10." << endl;
else if (n > 5)
  cout << "larger than 5." << endl;
else
  cout << "out of range!" << endl;
```

৮. নীচের nested if else (অন্তান্তি যদি নাহলে) খেয়াল করো। Indentation (ছাড়ন) যে ভাবে দেয়া হয়েছে তাতে মনে হচ্ছে না যা লিখতে চাওয়া হয়েছে তা লেখা হয়েছে।

```
if (n < 10)
  if (n > 0)
    cout << "positive." << endl;
else
  cout << "_________." << endl;
```

যদি n এর মান ৭ বা ১৫ বা -৩ input (যোগান) দেয়া হয় তাহলে output (ফলন) কী হবে? Statementটির (বিবৃতি) syntax (গঠন) এমন ভাবে ঠিক করো যাতে indentation (ছাড়ন) দেওয়া থেকে যেমনটি লিখতে চাওয়া হয়েছে বলে মনে হয় outputও ঠিক সে রকম আসে। আর সেক্ষেত্রে শুন্যস্থানে কী হবে বলে যৌক্তিক মনে হয় সেটাও ঠিক করো। অন্যদিকে যা লেখা হয়েছে সেটা ঠিকই আছে ধরে নিয়ে কেবল indentationটা (ছাড়ন) ঠিক করো, আর তাতে শুন্যস্থানে কী বসানো যথার্থ হবে তাও নির্ণয় করো।

৯. তিনটি সংখ্যা input (যোগান) নিয়ে কোনটি বড়, কোনটি ছোট outputএ দেখাও।

১০. তিনটি সংখ্যা input (যোগান) নিয়ে তাদের মধ্যে মাঝেরটি outputএ (ফলন) দেখাও।

১১. তিনটি সংখ্যা input (যোগান) নিয়ে তাদেরকে উর্ধক্রমে সাজিয়ে output (ফলন) দাও।

১২. গণিতে প্রাপ্ত নম্বর input (যোগান) নিয়ে সেটা থেকে letter grade (বর্ণ মান) output দাও। ধরো ৯০ বা বেশী হলে A, ৮০ বা বেশী হলে B, ৭০ বা বেশী হলে C, ৬০ বা বেশী হলে D, ৫০ বা বেশী হলে E, আর তারও কম হলে F letter grade পাওয়া যায়।

১৩. একটি দ্বিমাত্রিক (two dimensional) বিন্দুর স্থানাঙ্ক দেওয়া আছে, বিন্দুটি চারটি চতুর্ভাগের (quadrant) ঠিক কোনটিতে পড়বে নির্ণয় করো।

১৪. একটি প্রগমণ ১, ২, ৩, ..., ৯, ১১, ২২, ৩৩, ..., ৯৯ এর ১ম পদ ১, আর ১৮ তম পদ ৯৯। কততম পদ দেখাতে হবে তা input (যোগান) নিয়ে পদটি outputএ (ফলন) দেখাও।

১৫. তোমাকে -১০০ ও ১০০ এর মধ্যে দুটি সংখ্যা input (যোগান) হিসাবে দেওয়া হবে, তুমি ওই দুটি সংখ্যা সহ তাদের মাঝের সকল সংখ্যার যোগফল outputএ (ফলন) দেখাও।

১৬. একটি প্রদত্ত বর্ষ leap year কি না তা নির্ণয়ের programটি তুমি if-else ladder (যদি-নাহলে মই) ব্যবহার করে লিখবে। তবে programটি রচনা করার সময় তোমাকে মনে রাখতে হবে যে এটি ১ থেকে ২০০০ সাল পর্যন্ত প্রতিটি সালের জন্য চালানো হবে। কাজেই তুমি ladderএর শর্তগুলো এমন ভাবে সাজাবে যাতে program দ্রুততম হয়।

১৭. বাংলা বছরের কততম মাস তা input (যোগান) নিয়ে সেই মাসের নাম ও ওই মাসে কত দিন তা outputএ (ফলন) দেখাও। একাজে switch case (পল্টি ব্যাপার) ব্যবহার করো।

১৮. কতটা বাজে সেই সময় ঘন্টায় input (যোগান) নিয়ে মাঝরাত (১-২), প্রভাত (৩-৬), সকাল (৭-১১), দুপুর (১২-১৪), বিকাল (১৫-১৭), সন্ধ্যা (১৮-১৯), রাত (২০-২৮) outputএ দেখাও। একাজে switch case (পল্টি ব্যাপার) ব্যবহার করো।

১৯. এমন একটি program (ক্রমলেখ) লিখো যেটি ১-৫ পর্যন্ত ক্রম অনুযায়ী পাঁচটা কোমল পানীয়ের (পানি, কোক, স্প্রাইট, ফানটা, পেপসি) নামের তালিকা দেখাবে, তারপর ক্রমিক নম্বর input (যোগান) নিয়ে কোমল পানীয়টির নাম outputএ (ফলন) দেখাবে। আর

ক্রমিক নম্বরটি যদি ১-৫ এর বাইরে হয়, তাহলে সে সংক্রান্ত একটি error message (ত্রুটি বার্তা) দেখাবে। তুমি এই programটি একবার switch case (পল্টি ব্যাপার) ব্যবহার করে আবার if else (যদি নাহলে) ব্যবহার করে করো।

২০. একটি সংখ্যার পুরক সংখ্যা নির্ণয় করো। সংখ্যাট এক অঙ্কের হলে তার পুরক সংখ্য ৯ এর সাথে বিয়োগফল, দুই অঙ্কের হলে ৯৯ এর সাথে বিয়োগফল, তিন অঙ্কের হলে ৯৯৯ এর সাথে বিয়োগফল। তিনের চেয়ে বেশী অঙ্কের সংখ্যা input (যোগান) দেওয়া হবে না।

২১. এমন একটি program (ক্রমলেখ) লিখো যেটা ৫ জন লোক যাদের ক্রমিক ১-৫ তাদের কে কতটা করে পরোটা খেয়েছে input (যোগান) নিবে। Programটি তারপর একজনে সর্বোচ্চ কয়টা পরোটা খেয়েছে সেটা outuputএ (ফলন) দেখাবে। আর কোন লোক সর্বোচ্চ সংখ্যক পরোটা খেয়েছে programটি সেটাও দেখাবে, তবে সর্বোচ্চ পরোটা খাওয়া একাধিক ব্যক্তি থাকলে কেবল প্রথমজনের ক্রমিক নম্বর হলেই চলবে।

২২. একজন লোক স্বাভাবিক নিয়ম অনুযায়ী সপ্তাহে ৪০ ঘন্টা কাজ করে, ৪০ ঘন্টার বেশী কাজ করলে অতিরিক্ত সময়টুকুর জন্য স্বাভাবিক নিয়মের চেয়ে ১.৫ গুণ মজুরি পায়। কোন এক সপ্তাহে লোকটি কত ঘন্টা কাজ করেছে আর স্বাভাবিক নিয়মে ঘন্টা প্রতি মজুরি কত তা input (যোগান) নিয়ে ওই সপ্তাহে তার মোট মজুরি কত তা outputএ (ফলন) দেখাও।

২৩. ধরো তুমি চার টুকরো কাগজ নিয়েছো। তোমার ১ম টুকরোতে লেখা আছে ১, ৩, ৫, ৭, ৯, ১১, ১৩, ২য় টুকরোতে আছে ২, ৩, ৬, ৭, ১০, ১১, ১৪, ১৫, ৩য় টুকরোতে আছে ৪, ৫, ৬, ৭, ১২, ১৩, ১৪, ১৫, ৪র্থ টুকরোতে আছে ৮, ৯, ১০, ১১, ১২, ১৩, ১৪, ১৫। তোমার program (ক্রমলেখ) ব্যবহারকারী মনে মনে একটি সংখ্যা ধরবে, আর সেটি ১ম, ২য়, ৩য়, ৪র্থ টুকরোর কোন কোনটিতে আছে input (যোগান) দিবে, তারপর তোমার program ব্যবহারকারী মনে মনে যে সংখ্যাটি ধরেছে সেটি outputএ (ফলন) দেখাবে। এটি খুব সহজ একটি ব্যাপার। যে যে টুকরোতে সংখ্যাটি আছে ওই টুকরোগুলোর প্রথম সংখ্যাগুলো যোগ করলেই ব্যবহারকারীর সংখ্যাটি পাওয়া যাবে। যেমন ব্যবহারকারীর সংখ্যাটি যদি ১, ৩, ৪ নম্বর টুকরোতে থাকে তাহলে সংখ্যাটি ১ + ৪ + ৮ = ১৩।

Programming Solutions: এবার আমরা programming সমস্যাগুলোর সমাধান দেখবো। মনে রাখবে সমাধানগুলো দেয়া হয়েছে তুমি যদি একান্তই নিজে করতে পারছোনা বলে মনে হয়, তখন কেবল একটু সাহায্য যাতে পেতে পারো তাই। কাজেই পারতপক্ষে সমাধান দেখবে না।

১. নীচের programএর (ক্রমলেখ) output (ফলন) কী তা প্রথমে খাতা কলমে নির্ণয় করো, আর তারপর computerএ চালিয়ে তার সাথে মিলাও।

```
int n;  // আদি মান আরোপ করা হয় নি
cout << (n = 4) << endl;
cout << (n == 4) << endl;
cout << (n > 3) << endl;
cout << (n < 4) << endl;
cout << (n = 0) << endl;
cout << (n == 0) << endl;
cout << (n > 0) << endl;
cout << (n && 4) << endl;
```

```
cout << (n || 4) << endl;
cout << (!n) << endl;
```

```
4              // আরোপণ হবে 4
1              // মান আসলেই তো 4
1              // কাজেই 3 এর বেশী
0              // 4 এর সমান, কম তো নয়
0              // আরোপন হবে 0
1              // মান 0 এর সমান, সত্য
0              // মান তো 0, বেশী তো নয়
0              // 0 হলো মিথ্যা তাই ফলাফল মিথ্যা
1              // 4 যেহেতু সত্য, তাই ফলাফল সত্য
1              // 0 নিজে মিথ্যা তাই !0 সত্য
```

২. নীচের programএ (ক্রমলেখ) কিছু syntactical (গঠনগত) ভুল আছে। ভুলটা কোথায় বলে তুমি মনে করো? ভুলটা এমন ভাবে ঠিক করো যাতে এটির indentation (ছাড়ন) দেখে যা করতে চাওয়া হয়েছিল বলে মনে হয়, programটি (ক্রমলেখ) semantically (অর্থবোধকতায়) যেন একদম তাই করে।

```
if (x >= y)
  sum += x;
  cout << "x large" << endl;
else
  sum += y;
  cout << "y large" << endl;
```

উপরের programএ indentation দেখে মনে হয় যদির শর্ত সত্য হলে বা মিথ্যা হলে উভয় ক্ষেত্রে ঠিক তাদের পরের দুই সারিতে থাকা statementগুলো নির্বাহ (execute) হবে। কিন্তু একাধিক statement যদি execute করতে হয় সেক্ষেত্রে আমাদের নীচের programএর মতো করে বাঁকা বন্ধনী দিয়ে compound statement (যৌগিক বিবৃতি) বানিয়ে নিতে হবে। বাঁকা বন্ধনী না দেয়ায় উপরের programটি compile (সংকলন) করতে গেলে error দেখাবে। Errorটা হল সংশ্লিষ্ট if সাপেক্ষে elseটা ঠিক জায়গায় নাই। যদি else টা cout << "x large" << endl; এর আগে থাকে তাহলে গঠনগত (syntactically) ভাবে শুদ্ধ হয়, কিন্তু তাতে অবশ্য আমরা যা করতে চাই তা হতো না।

```
if (x >= y)
{
  sum += x;
  cout << "x large" << endl;
}
else
{
  sum += y;
  cout << "y large" << endl;
```

```
}
```

৩. নীচের program (ক্রমলেখ) চালালে কী output (ফলন) পাওয়া যাবে?

```
int n, k = 5;
n = (100 % k ? k + 1 : k - 1);
cout << "n = " << n << " k = " << k << endl;
```

উপরের programএর output নীচে দেখানো হলো। শুরুতে k এর মান assign (আরোপণ) করা হলো 5। তারপর 100 যেহেতু 5 দ্বারা বিভাজ্য তাই 100 % k হবে শুন্য যাহা Boolean (বুলক) হিসাবে ধরলে মিথ্যা, ফলে ternary operatorএর (তিনিক অণুক্রিয়া) শেষের অংশ k − 1 অর্থাৎ 4 হবে ফলাফল যা n variableএ (চলক) assign (আরোপিত) হবে। সবমিলিয়ে n হলো 4 আর k শুরুতে যা ছিলো তাই অর্থাৎ 5।

```
n = 4 k = 5
```

৪. নীচের program (ক্রমলেখ) চালালে কী output (ফলন) পাওয়া যাবে?

```
int found = 0, count = 5;
if (!found || ++count == 0)
  cout << "danger" << endl;
cout << "count = " << count << endl;
```

উপরের programএর output (ফলন) নীচে দেখানো হলো। Variable found এর মান 0 অর্থ্যাৎ মিথ্যা, ফলে !found হলো সত্য, আর তাই অথবা || এর ফলাফলও সত্য। লক্ষ্য করো এই ফলাফল নির্ধারণে আমাদের কিন্তু || এর পরের অংশ execute (নির্বাহ) করার দরকারই নাই। Partial evaluationএর (আংশিক মূল্যায়ন) কারণে এটি ঘটবে। তাহলে || এর ফলাফল সত্য আসায় outputএ আসবে "danger"। আর ++count যেহেতু নির্বাহিতই হয় নি, তাই count এর মান 5 ই দেখাবো।

```
danger
count = 5
```

৫. নীচের programএ, সম্ভবত বলা যায় যে conditional statementএর (শর্তালি বিবৃতি) একদম প্রথম সারিতেই একটা ভুল আছে। Programটি যে ভাবে লেখা আছে সেরকম অবস্থায়ই যদি execute (নির্বাহ) করা হয় তাহলে output (ফলন) কী হবে? আর যেটা করতে চাওয়া হয়েছিল বলে মনে হয় যদি সেটা করা হয় তাহলে output কী হবে?

```
int n = 5;
if (n = 0) // অণুক্রিয়াটি খেয়াল করো
  cout << "n is zero." << endl;
else
  cout << "n is not zero" << endl;
cout << "square of n " << n * n << endl;
```

উপরের programএর ২য় সারিতে assignment (আরোপন) = operator ব্যবহার করা হয়েছে, সাধারণতো শর্ত পরীক্ষার জন্য সমান (equal) == operator ব্যবহার করা হয়। সুতরাং এটি সম্ভবত একটা ভুল যেটা প্রায়শই আমাদের হয়ে থাকে। যাই হোক programটি যেমন আছে তেমনি চালালে assignmentএর ফলে n এর মান হবে শুন্য আর assignment operatorএর ফলাফলও হবে শুন্য, যা Boolean value (বুলক মান) হিসাবে মিথ্যা। সুতরাং else অংশে থাকা statementটুকু নির্বাহিত হবে, আর আমরা outputএ পাবো n is not zero। বিষয়টি কেমন যেন গোলমেলে তাই না, একদিকে n এর মান আসলেই শুন্য, কিন্তু অন্য দিকে output দেখাচ্ছে n শুন্য নয়! যাইহোক n এর মান শুন্য assignmentএর ফলে if else এর পরের সারিতে থাকা cout এর কারণে outputএ আসবে n er borgo 0। এই outputগুলো নীচে বামদিকে দেখানো হলো, আর ডান দিকে রয়েছে assignment (আরোপণ) = না লিখে আমরা যদি সমান (equality) == লিখি তাহলে output (ফলন) কী হবে তা। লক্ষ্য এবারে n এর মান কিন্তু 5ই থাকছে যা initial assignment করা হয়েছে। ফলে n == 0 মিথ্যা হওয়ায় আগের মতোই n is not zero দেখাবে আর পরের সারিতে 5 এর বর্গ হবে 25, কাজেই ফলনে আসবে square of n 25।

```
n is not zero                    n is not zero
square of n 0                    square of n 25
```

৬. নীচের conditional statement (শর্তালি বিবৃতি) তে অনেক অপ্রয়োজনীয় শর্ত আছে। তো সেই অপ্রয়োজনীয় শর্তগুলো বাদ দিয়ে conditional statementটি আবার লেখো।

```
float income;
cout << "monthly income: ";
cin >> income;

if (income < 0)
  cout << "loan will increase." << endl;
else if (income >= 0 && income < 1200)
  cout << "below poverty limit." << endl;
else if (income >= 1200 && income < 2500)
  cout << "slightly well-off." << endl;
else if (income >= 2500)
  cout << "sufficiently well-off." << endl;
```

অদরকারী শর্তগুলো ছাড়া program (ক্রমলেখ) কেমন হবে তা নীচে দেখানো হলো। যদি income < 0 এই শর্ত মিথ্যা হয়, তাহলে অবশ্যই income >=0 সত্য হবে। কাজেই if else ladderএ (যদি নাহলে মই) elseএর সাথে যে যদি থাকবে সেখানে income >=0 আবার লেখার কোন দরকার নেই। Control (নিয়ন্ত্রণ) ওইখানে যাওয়া মানে ওই শর্ত অবশ্যই সত্য আর এবং (and) && operatorএর (অণুক্রিয়া) একটি operand (উপাদান) সত্য হলে ফলাফল কেবল দ্বিতীয় operandএর ওপর নির্ভর করে। সুতরাং আমরা সরলীকরণ করে কেবল && এর দ্বিতীয় operandটিকেই লিখবো। এই একই ভাবে income >= 1200 আর income >= 2500 লেখার কোন দরকার নাই।

```
float income;
```

```
  cout << "monthly income: ";
  cin >> income;

  if (income < 0)
    cout << "loan will increase." << endl;
  else if (income < 1200) // >= 0 দরকার নেই
    cout << "below poverty limit." << endl;
  else if (income < 2500) // >= 1200 দরকার নেই
    cout << "slightly well-off." << endl;
  else       // >= 2500 দরকার নেই
    cout << "sufficiently well-off." << endl;
```

৭. যদি ভিন্ন ভিন্ন বার চালানোর সময় ০, ১৫, বা ৭ input (যোগান) দেয়া হয় তাহলে নীচের programএর (ক্রমলেখ) output (ফলন) কোন বারে কী হবে । কত input দিলে outputএ "out of range!" আসবে?

```
  int n;
  cout << "number is: ";
  cin >> n;

  if (n < 10)
    cout << "less than 10." << endl;
  else if (n > 5)
    cout << "larger than 5." << endl;
  else
    cout << "out of range!" << endl;
```

Input (যোগান) হিসাবে $0, 15, 7$ দিলে, উপরের program কী output (ফলন) দেবে তা নীচে ৩ স্তম্ভে দেখানো হলো। এই program n এর কোন মানের জন্যই out of range! ফলন দিবে না, নিয়ন্ত্রণ (control) কোন অবস্থাতেই সংশ্লিষ্ট বিবৃতিতে যাবে না। সুতরাং else cout << "out of range!" << endl; অংশটুকু পুরোপুরি অদরকারী আর সে কারণে মুছে দেয়া যায়, তাতে programএর বৈশিষ্ট্যে কোন প্রভাব পড়বে না।

```
number is: 0     number is: 15     number is: 7
less than 10.    larger than 5.    less than 10.
```

৮. নীচের nested if else (অন্তান্তি যদি নাহলে) খেয়াল করো। Indentation (ছাড়ন) যে ভাবে দেয়া হয়েছে তাতে মনে হচ্ছে না যা লিখতে চাওয়া হয়েছে তা লেখা হয়েছে।

```
if (n < 10)
  if (n > 0)
    cout << "positive." << endl;
else
  cout << "----------." << endl;
```

যদি n এর মান ৭ বা ১৫ বা -৩ যোগান (input) দেয়া হয় তাহলে output (ফলন) কী হবে? Statementএর (বিবৃতি) syntax (গঠন) এমন ভাবে ঠিক করো যাতে indentation (ছাড়ন) দেওয়া থেকে যেমনটি লিখতে চাওয়া হয়েছে বলে মনে হয় outputও ঠিক সে রকম আসে। আর সেক্ষেত্রে শুন্যস্থানে কী হবে বলে যৌক্তিক মনে হয় সেটাও ঠিক করো। অন্যদিকে যা লেখা হয়েছে সেটা ঠিকই আছে ধরে নিয়ে কেবল indentationটা (ছাড়ন) ঠিক করো, আর তাতে শুন্যস্থানে কী বসানো যথার্থ হবে তাও নির্ণয় করো।

```
if (n < 10)
  if (n > 0)
    cout << "positive." << endl;
  else
    cout << "_________." << endl;
```

প্রদত্ত programটি (ক্রমলেখ) লেখার সময় এমন ভাবে indentation (ছাড়ন) দেয়া হয়েছে যে মনে হচ্ছে else অংশটুকু প্রথম if এর শর্ত n < 10 মিথ্যা হলে কার্যকর হবে। কিন্তু সিপিপি ভাষায় indentation বা ফাঁকা দেয়া না দেয়া computerএর (গণনি) জন্য কোন ব্যাপার নয়। আর dangling elseএর (ঝুলন্ত নাহলে) আলোচনা থেকে আমরা জানি এই else টি তার নিকটতম পূর্ববর্তী এমন একটি if এর সাথে সংশ্লিষ্ট যে if এর সাথে আর কোন else জুড়ে দেয়া হয় নি। কাজেই, সেই হিসাবে ঠিক উপরে যেমনটি দেখানো হলো, সেভাবে এই else টি দ্বিতীয় if এর শর্ত n > 0 মিথ্যা হলে কার্যকর হবে।

```
positive.          not output          _________.
```

এমতাবস্থায় এই program চালালে আমরা ৭, ১৫, বা -৩ input (যোগান) দিয়ে যে output (ফলন) পাবো তা উপরের তিনটি স্তম্ভে দেখানো হয়েছে। লক্ষ্য করো ১৫ input দিলে আমরা কোন output আসলে পাবো না, কারণ n < 10 শর্তের কোন else নেই। আর শুন্যস্থানটি _____ outputএ আসে যখন সংখ্যাটি শুন্য বা কম হয় অর্থাৎ non-positive হয়। আমরা তাহলে _____ এর স্থানে লিখতে পারি non−positive।

```
if (n < 10)
{
  if (n > 0)
    cout << "positive." << endl;
}
else
  cout << "_________." << endl;
```

Indentation (ছাড়ন) দেয়া দেখে যেমন মনে হয়, programটি (ক্রমলেখ) সেই অনুযায়ী সংশোধন করলে ঠিক উপরের মতো করে বাঁকা বন্ধনী ব্যবহার করতে হবে। সেক্ষেত্রে শুন্যস্থান অংশটি outputএ আসবে যখন n সংখ্যাটি ১০ এর বড় বা সমান, যেমন ধরো ১৫। এক্ষেত্রে আমরা তাই _____ এর বদলে লিখতে পারি "10 or larger"। এবার খেয়াল করো n এর মান যখন শুন্য এর বেশী কিন্তু ১০ এর কম যেমন ৭, তখন কিন্তু আমরা output পাবো positive, আর শুন্য বা কম হলে কোন outputই পাবো না।

৯. তিনটি সংখ্যা input (যোগান) নিয়ে কোনটি বড়, কোনটি ছোট outputএ দেখাও।

ফিরিস্তি ১৫.৮: Small and Big of Three Numbers (তিনটি সংখ্যার বড়-ছোট)

```
int a, b, c;            // চাইলে ভগ্নকও নিতে পারো
cout << "three numbers are? ";
cin >> a >> b >> c;     // যোগান নাও

int large, small;       // চলক ঘোষণা
if (a > b)              // a যদি বড় হয় b এর চেয়ে
  large = a, small = b;
else                    // b যদি বড় হয় a এর চেয়ে
  large = b, small = a;
if (large < c)          // c যদি large এর চেয়ে বড় হয়
  large =   c;
else if (small > c)     // c যদি small এর চেয়ে ছোট হয়
  small = c;

cout << "large " << large << " ";
cout << "small " << small << endl;
```

উপরের program খেয়াল করো। প্রথম দুটি সংখ্যা a ও b কে তুলনা করে বড় ও ছোট নির্ধারণ করা হয়ছে। তারপর c কে তুলনা করা হয়েছে সেটা আরো বড় কিনা দেখতে, যদি তা না হয় তাহলে সেটা আরো ছোট কিনা সেটা পরীক্ষা করা হয়েছে। লক্ষ্য করো c কে তুলনা করা সময় একটা else লাগানো হয়েছে, কারণ largeএর বড় হলে তো আর smallএর ছোট কিনা পরীক্ষা করার দরকার নেই। তুমি কিন্তু চাইলে নীচের মতো করেও program লিখতে পারো। প্রথমে ধরে নাও তোমার সংখ্যা একটাই কাজেই aই বড়, আবার aই ছোট। এরপর তাদের সাথে b কে তুলনা করো। আর শেষে তাদের সাথে c কে তুলনা করো।

```
int large = a, small = a; // ধরে নেই a–ই বড় ও ছোট
if (large < b)            // b যদি তার চেয়েও বড় হয়
  large =   b;
else if (small > b)     // b যদি তার চেয়েও ছোট হয়
  small = b;
if (large < c)            // c যদি তার চেয়েও বড় হয়
  large =   c;
else if (small > c)     // c যদি তার চেয়েও ছোট হয়
  small = c;
```

১০. তিনটি সংখ্যা input (যোগান) নিয়ে তাদের মধ্যে মাঝেরটি output দেখাও।

ফিরিস্তি ১৫.৯: Median of Three Numbers (তিনটি সংখ্যার মধ্যক)

```
// ধরো চলক তিনটি a, b, c ঘোষণা করে যোগান নেয়া হয়েছে

if (a > b)                // ক্রম হলো a > b
  if (c > a)              // ক্রম হলো  c > a > b
    cout << a << endl;
```

```
    else if (b > c)          //  ক্রম হলো a > b > c
      cout << b << endl;
    else                     // ক্রম হলো a >= c >= b
      cout << c << endl;
  else                       // ক্রম হলো a <= b
    if (c < a)               // ক্রম হলো c < a <= b
      cout << a << endl;
    else if c > b)           // ক্রম হলো a <= b < c
      cout << b << endl;
    else                     // ক্রম হলো a <= c <= b
      cout << c << endl;
```

উপরের ক্রমলেখতে প্রথমে a ও b তুলনা করা হয়েছে। তারপর c তাদের বড়টির চেয়ে বড় কিনা, নাহলে ছোটটির চেয়ে ছোট কিনা পরীক্ষা করা হয়েছে, আর তাও না হলে সেটি উভয়ের মাঝামাঝি। এভাবে তিনটি সংখ্যার ক্রম জানা হয়ে গেলে মাঝেরটি outputএ (ফলন) দেখানো হয়েছে। এটি nesting (অন্তান্তি) ও ladderএর (মই) চমৎকার উদাহরণ।

১১. তিনটি সংখ্যা input (যোগান) নিয়ে তাদেরকে উর্ধক্রমে সাজিয়ে output (ফলন) দাও।

ফিরিস্তি ১৫.১০: Three Numbers in Ascending Order (তিনটি সংখ্যার উর্ধক্রম)

```
// ধরো চলক তিনটি a, b, c ঘোষণা করে যোগান নেয়া হয়েছে
if (a > b)                   //  ক্রম হলো a > b
  if (c > a)                 //  ক্রম হলো  c > a > b
    cout << b << " " << a << " " << c << endl;
  else if (b > c)            //  ক্রম হলো a > b > c
    cout << c << " " << b << " " << a << endl;
  else                       // ক্রম হলো a >= c >= b
    cout << b << " " << c << " " << a << endl;
else                         // ক্রম হলো a <= b
  if (c < a)                 // ক্রম হলো c < a <= b
    cout << c << " " << a << " " << b << endl;
  else if c > b)             // ক্রম হলো a <= b < c
    cout << a << " " << b << " " << c << endl;
  else                       // ক্রম হলো a <= c <= b
    cout << a << " " << c << " " << b << endl;
```

উপরের ক্রমলেখতে প্রথমে a ও b তুলনা করা হয়েছে। তারপর c তাদের বড়টির চেয়ে বড় কিনা, নাহলে ছোটটির চেয়ে ছোট কিনা পরীক্ষা করা হয়েছে, আর তাও না হলে সেটি উভয়ের মাঝামাঝি। এভাবে তিনটি সংখ্যার ক্রম জানা হয়ে গেলে তাদের মানের উর্ধক্রমে (ascending order) outputএ (ফলন) দেখানো হয়েছে। এটি if elseএর (যদি নাহলে) nesting (অন্তান্তি) ও ladderএর (মই) এক সাথে ব্যবহারের চমৎকার উদাহরণ।

১২. গণিতে প্রাপ্ত নম্বর input (যোগান) নিয়ে সেটা থেকে letter grade (বর্ণ মান) output দাও। ধরো ৯০ বা বেশী হলে A, ৮০ বা বেশী হলে B, ৭০ বা বেশী হলে C, ৬০ বা বেশী হলে D, ৫০ বা বেশী হলে E, আর তারও কম হলে F letter grade পাওয়া যায়।

ফিরিস্তি ১৫.১১: Letter Grades from Numbers (নম্বর হতে বর্ণমান)

```
cout << "maths number? ";
int number; cin >> number;

if (number >= 90)
  cout << "letter grade A" << endl;
else if (number >= 80)
  cout << "letter grade B" << endl;
else if (number >= 70)
  cout << "letter grade C" << endl;
else if (number >= 60)
  cout << "letter grade D" << endl;
else if (number >= 50)
  cout << "letter grade E" << endl;
else // ৫০ এর ছোট
  cout << "letter grade F" << endl;
```

উপরের program switch case (পল্টি ব্যাপার) দিয়ে করা সম্ভব নয়, কারন এখানে >= তুলনা ব্যবহার করতে হবে। Switch case কেবল সমান == তুলনায় ব্যবহার করা যায়।

১৩. একটি দ্বিমাত্রিক (two dimensional) বিন্দুর স্থানাঙ্ক দেওয়া আছে, বিন্দুটি চারটি চতুর্ভাগের (quadrant) ঠিক কোনটিতে পড়বে নির্ণয় করো।

এই programএ (ক্রমলেখ) আমরা কেবল চতুর্ভাগ (quadrant) বিবেচনা না করে বরং, বিন্দুটি কোন অক্ষের ওপরে কিনা, হলে ধনাত্মক দিকে না ঋণাত্মক দিকে, অথবা স্থানাংকরে মূল বিন্দুতে কিনা তাও বিবেচনা করবো। যে কোন একটি প্রদত্ত বিন্দুর x বা y দুটোই আলাদা আলাদা ভাবে ধনাত্মক বা ঋণাত্মক বা শুন্য এই তিন রকম হতে পারে। কাজেই একসাথে বিবেচনা করলে আমরা মোট নয় রকম combination (সমাবেশ) পাবো।

ফিরিস্তি ১৫.১২: Quadrant of a Point (বিন্দুর চতুর্ভাগ নির্ণয়)

```
float x, y;
cout << "abscissa x? ";
cin >> x;
cout << "ordinate y? ";
cin >> y;

if (x > 0)
  if (y > 0)
    cout << "first quadrant" << endl;
  else if (y < 0)
    cout << "fourth quadrant" << endl;
  else    // y শুন্য
    cout << "on positive x axis" << endl;
else if (x < 0)
```

```
    if (y > 0)
      cout << "second quadrant" << endl;
    else if (y < 0)
      cout << "third quadrant" << endl;
    else
      cout << "on negative x axis" << endl;
  else    // x শুন্য
    if (y > 0)
      cout << "on positive y axis" << endl;
    else if (y < 0)
      cout << "on negative y axis" << endl;
    else    // y শুন্য
      cout << "coordinate origin" << endl;
```

১৪. একটি প্রগমণ ১, ২, ৩, ..., ৯, ১১, ২২, ৩৩, ..., ৯৯ এর ১ম পদ ১, আর ১৮ তম পদ ৯৯। কততম পদ দেখাতে হবে তা input (যোগান) নিয়ে পদটি outputএ (ফলন) দেখাও।

```
cout << "which term: " << endl;
int n; cin << n;

if (n < 0)
  cout << "out of range" << endl;
else if (n <= 9)  // এক অঙ্কের সংখ্যা
  cout << n << endl;
else if (n <= 18) // দুই অঙ্কের সংখ্যা
  cout << ((n-9) * 11) << endl;
else
  cout << "out of range" << endl;
```

১৫. তোমাকে -১০০ ও ১০০ এর মধ্যে দুটি সংখ্যা input (যোগান) হিসাবে দেওয়া হবে, তুমি ওই দুটি সংখ্যা সহ তাদের মাঝের সকল সংখ্যার যোগফল outputএ (ফলন) দেখাও।

নীচের সংশ্লিষ্ট program (ক্রমলেখ) দেখানো হলো। যে সংখ্যা দুটি input (যোগান) নেয়া হবে, সেগুলো অবশ্যই -১০০ ও ১০০ এর ভিতরে হতে হবে। আমরা তাই আগে পরীক্ষা করে দেখবো। যদি n1 বা n2 যে কোনটি -100 এর ছোট বা 100 এর বড় হয়, তাহলে error message (ত্রুটি বার্তা) দেখিয়ে বিফল হয়ে control ফেরত যাবে। খেয়াল করো আমাদের কিন্তু শর্তগুলোকে অথবা || দিয়ে যুক্ত করতে হবে।

```
cout << "two number are? "; // যাচনা
int n1, n2; cin >> n1 >> n2;     // যোগান

// -১০০ ও ১০০ এর মধ্যে কিনা পরীক্ষা করতে হবে
if (n1 < -100 || n1 > 100 ||
        n2 < -100 || n2 > 100)
{
```

```
    cout << "number out of range" << endl;
    return EXIT_FAILURE;
  }

  int s, n;    // (প্রথম পদ + শেষ পদ) আর পদসংখ্যা

  s = n1 + n2;  // প্রথম পদ + শেষ পদ।

  if (n1 > n2)     // কোনটা ছোট কোনটা বড়
    n = n1 - n2 + 1; // পদসংখ্যা
  else
    n = n2 - n1 + 1;  // পদসংখ্যা

  cout << "sum " << s*n/2;  // ফলন
```

এবার আমরা জানি কোন সমান্তর প্রগমণের সংখ্যাগুলোর যোগফল হলো (প্রথম সংখ্যা + শেষ সংখ্যা) * পদসংখ্যা / ২। সংখ্যা দুটো যোগান নেয়ার সময় ব্যবহারকারী যে কোনটিকে আগে input দিতে পারে, মানে কোনটা বড় কোনটা ছোট আমরা নিশ্চিত থাকবো না। (প্রথম সংখ্যা + শেষ সংখ্যা) এই যোগফল s বের করতে এতে কোন সমস্যা হবে না, তবে পদসংখ্যা n বের করতে গেলে আমাদের জানতে হবে কোনটা বড় কোনটা ছোট। ধরো ৭ আর ১৩ নিজেদের সহ তাদের মধ্যে কয়টা সংখ্যা আছে সেটা বের করা যায় ১৩ - ৭ + ১ হিসাব করে, যেখানে ১৩ হলো বড় আর ৭ হলো ছোট। তো n1 আর n2 এর নিজেদের সহ তাদের মাঝে মোট কয়টি সংখ্যা আছে তা বের করতে আমাদের জানতে হবে কোনটি বড়। তো আমরা একটি যদি নাহলে (if else) ব্যবহার করে দেখবো n1 > n2 কিনা, যদি হয় তাহলে পদসংখ্যা n1 - n2 + 1 আর যদি না হয় তাহলে পদসংখ্যা হবে n2 - n1 + 1। সবশেষে যোগফল হলো s * n / 2 আমরা যেটা outputএ (ফলন) দেখাবো।

১৬. একটি প্রদত্ত বর্ষ অধিবর্ষ কি না তা নির্ণয়ের programটি (ক্রমলেখ) তুমি if else ladder (যদি-নাহলে মই) ব্যবহার করে লিখবে। তবে programটি রচনা করার সময় তোমাকে মনে রাখতে হবে যে এটি ১ থেকে ২০০০ সাল পর্যন্ত প্রতিটি সালের জন্য চালানো হবে। কাজেই তুমি ladderএর শর্তগুলো এমন ভাবে সাজাবে যাতে program দ্রুততম হয়।

যদি ১ থেকে ২০০০ সাল পর্যন্ত প্রতিটি সালের জন্য চালানো হয় তাহলে আমরা প্রথমে প্রত্যেক রকমের সালের হিসাব করি। মোটামুটি প্রতি চারটি সালের তিনটি অধিবর্ষ নয়, একটি অধিবর্ষ। কাজেই সবচেয়ে বেশী সংখ্যক ২০০০ / ৪ * ৩ = ১৫০০ টি সাল আছে যে গুলো ৪ দিয়ে বিভাজ্য হয় না, এগুলোর কোনটিই অধিবর্ষ নয়। বাঁকী ৫০০ টি সাল ৪ দিয়ে বিভাজ্য। এদের মধ্যে যেগুলো ১০০ দিয়ে বিভাজ্য নয় যেমন ১৯৯৬ এমন ২০টি ছাড়া বাঁকী ৪৮০ টি অধিবর্ষ। আর ওই ২০টি সালের মধ্যে যে ১৫টি ৪০০ দিয়ে বিভাজ্য নয় সেগুলো অধিবর্ষ নয়, আর বাঁকী ৪টি সাল যে গুলো ৪০০ দিয়ে বিভাজ্য সেগুলো অধিবর্ষ।

```
if (year % 4 != 0)          // ৪ দিয়ে বিভাজ্য নয়
  cout << "leap year no" << endl;
else if (year % 100 != 0) // ১০০ দিয়ে বিভাজ্য নয়
  cout << "leap year yes" << endl;
else if (year % 400 != 0) // ৪০০ দিয়ে বিভাজ্য নয়
```

```
    cout << "leap year no" << endl;
  else  //  if (year % 400 = 0) 400 দিয়ে বিভাজ্য
    cout << "leap year yes" << endl;
```

দ্রুততম গতির programএর (ক্রমলেখ) জন্য যে রকমের সাল সবচেয়ে বেশী সেগুলো নির্ণয় করতে সবচেয়ে কম সংখ্যক শর্ত পরীক্ষণ ব্যবহার করতে হবে। কাজেই আমরা ১৫০০ সাল যেগুলো ৪ দিয়ে বিভাজ্য নয় সেগুলোকে প্রথম শর্ত পরীক্ষা করেই বের করতে চাইবো। উপরের program খেয়াল করো, আমরা তাই করেছি। এরপরে রয়েছে যে ৪৮০টি বছর যেগুলো ৪ দিয়ে বিভাজ্য কিন্তু ১০০ দিয়ে বিভাজ্য নয়। আমরা এগুলোকে দুইবার শর্ত পরীক্ষা করে বের করতে চাই। একটা শর্ত হচ্ছে ৪ দিয়ে বিভাজ্য নাহওয়া কাজেই প্রথম শর্তের else হিসাবে থাকবে সেটা, আরেকটি শর্ত হলো ১০০ দিয়ে বিভাজ্য না হওয়া। উপরের programএর if else ladderএ (যদি নাহলে মই) দেখো else if দিয়ে এটা করা হয়েছে। এরপর থাকে ১৫ টি সাল যেগুলো ১০০ দিয়ে বিভাজ্য কিন্তু ৪০০ দিয়ে বিভাজ্য নয় এই ১৫ টি সাল, এগুলো নির্ণয় করা হয়েছে আরেকটি else if লাগিয়ে অর্থাৎ মোট তিনটি শর্ত পরীক্ষণ শেষে। আর সবশেষে ৪০০ দিয়ে বিভাজ্য সেই সালগুলো এসেছে সবশেষের else দিয়ে, এগুলোর জন্য তিনটি শর্ত পরীক্ষণই লেগেছে, কারণ শেষের শর্ত মিথ্যা হলেই তো এগুলো নির্ণীত হবে। তাহলে মোট শর্ত পরীক্ষা লাগলো কতগুলো? ১৫০০ * ১ + ৪৮০ * ২ + ১৫ * ৩ + ৫ * ৩ = ২৫২০ টি। তুমি আরো নানান ভাবে চেষ্টা করে দেখতে পারো, এর চেয়ে কম শর্ত পরীক্ষা করে করতে পারো কিনা! পারবে না!

```
  if (year % 4 != 0 ||
        (year % 100 == 0 && year % 400 != 0))
    cout << "leap year no" << endl;
  else
    cout << "leap year yes" << endl;
```

একই program আমরা if else ladder (যদি নাহলে মই) ব্যবহার না করে ঠিক উপরের programএর মতো Boolean connectives (বুলক সংযোজক) ব্যবহার করে করতে পারি। Boolean connectivesএর আংশিক মূল্যায়ন (partial evaluation) মনে আছে? অথবা || ক্ষেত্রে যে কোন একটি operand (উপাদান) সত্যি হলেই অন্যটি মূল্যায়ন ছাড়াই আমরা ফলাফল সত্য বলে ধরে নিতে পারি। আর && এর ক্ষেত্রে যে কোন একটি operand (উপাদান) মিথ্যা হলেই ফলাফল মিথ্যা বলে ধরে নেয়া যায়। কোন সাল অধিবর্ষ নয় যখন সালটি ৪ দ্বারা বিভাজ্য নয় অথবা ১০০ দ্বারা বিভাজ্য হলেও ৪০০ দ্বারা বিভাজ্য নয় তখন। যদির সাথে শর্ত হিসাবে সেটিই লাগানো হয়েছে দেখো। অন্যদিকে কোন সাল অধিবর্ষ হতে গেলে || এর ফলাফল মিথ্যা হতে হবে, তারমানে বাম ও ডানের উভয় operand (উপাদান) মিথ্যা হতে হবে অর্থাৎ year % 4 == 0 এবং (year % 100 == 0 && year % 400 != 0) মিথ্যা হতে হবে। এখানে year % 4 == 0 মিথ্যা হওয়া মানে বছরটি ৪ দ্বারা বিভাজ্য হওয়া আর (year % 100 == 0 && year % 400 != 0) মিথ্যা হতে গেলে && এর দুপাশের যেকোন একটি মিথ্যা হলেই হবে। তো (year % 100 == 0 মিথ্যা হওয়া মানে বছরটি ১০০ দ্বারা বিভাজ্য না হওয়া আর year % 400 != 0 মিথ্যা হওয়া মানে ৪০০ দিয়ে বিভাজ্য হওয়া।

১৭. বাংলা বছরের কততম মাস তা input (যোগান) নিয়ে সেই মাসের নাম ও ওই মাসে কত দিন তা outputএ (ফলন) দেখাও। একাজে switch case (পল্টি ব্যাপার) ব্যবহার করো।

ফিরিস্তি ১৫.১৩: Bengali Month Names (বাংলা মাসের নাম)

```
int mash; cin >> mash;  // চাইলে যাচনা করতে পারো

switch(mash)
{
  case 1: cout << "boishakh 31" << endl; break;
  case 2: cout << "joistho 31" << endl; break;
  case 3: cout << "ashar 31" << endl; break;
  case 4: cout << "shrabon 31" << endl; break;
  case 5: cout << "vadro 31" << endl; break;
  case 6: cout << "arshin 30" << endl; break;
  case 7: cout << "kartik 30" << endl; break;
  case 8: cout << "ogrohayon 30" << endl; break;
  case 9: cout << "poush 30" << endl; break;
  case 10: cout << "magh 30" << endl; break;
  case 11: cout << "falgun 30" << endl; break;
  case 12: cout << "choitro 30" << endl; break;
  default: cout << "ojana mash" << endl; break;
}
```

১৮. কতটা বাজে সেই সময় ঘন্টায় input (যোগান) নিয়ে মাঝরাত (০-২), প্রভাত (৩-৬), সকাল (৭-১১), দুপুর (১২-১৪), বিকাল (১৫-১৭), সন্ধ্যা (১৮-১৯), রাত (২০-২৪) ফলনে দেখাও। একাজে switch case (পল্টি ব্যাপার) ব্যবহার করো।

```
int vartime;
cin >> vartime; // যাচনা করতে পারো

switch(vartime)
{
  case 0: case 1: case 2:
    cout << "midnight" << endl; break;
  case 3: case 4: case 5: case 6:
    cout << "dawn" << endl; break;
  case 7: case 8: case 9: case 10: case 11:
    cout << "morning" << endl; break;
  case 12: case 13: case 14:
    cout << "midday" << endl; break;
  case 15: case 16: case 17:
    cout << "afternoon" << endl; break;
  case 18: case 19:
    cout << "evening" << endl; break;
  case 20: case 21: case 22: case 23
    cout << "night" << endl; break;
  default:
```

```
        cout << "unknown time" << endl;
    }
```

১৯. এমন একটি program (ক্রমলেখ) লিখো যেটি ১-৫ পর্যন্ত ক্রম অনুযায়ী পাঁচটা কোমল পানীয়ের (পানি, কোক, স্প্রাইট, ফানটা, পেপসি) নামের তালিকা দেখাবে, তারপর ক্রমিক নম্বর input (যোগান) নিয়ে কোমল পানীয়টির নাম outputএ (ফলন) দেখাবে। আর ক্রমিক নম্বরটি যদি ১-৫ এর বাইরে হয়, তাহলে সে সংক্রান্ত একটি error message (ত্রুটি বার্তা) দেখাবে। তুমি এই programটি একবার switch case (পল্টি ব্যাপার) ব্যবহার করে আবার if else (যদি নাহলে) ব্যবহার করে করো।

```
cout << "list" << endl;
cout << "1 water" << endl;
cout << "2 coke" << endl;
cout << "3 sprite" << endl;
cout << "4 fanta" << endl;
cout << "5 pepsi" << endl;
cout << endl;

cout << "choice: " << endl;
int choice; cin >> choice;

cout << "choice ";
switch(choice)
{
  case 1: cout << "water" << endl; break;
  case 2: cout << "coke" << endl; break;
  case 3: cout << "sprite" << endl; break;
  case 4: cout << "fanta" << endl; break;
  case 5: cout << "pepsi" << endl; break;
  default: cout << "unknown" << endl; break;
}
```

উপরের programএর (ক্রমলেখ) switch case (পল্টি ব্যাপার) অংশটি if else (যদি নাহলে) ব্যবহার করে লিখলে নীচের মতো হবে।

```
if (choice == 1)
  cout << "water" << endl;
else if (choice == 2)
  cout << "coke" << endl;
else if (choice == 3)
  cout << "sprite" << endl;
else if (choice == 4)
  cout << "fanta" << endl;
else if (choice == 5)
  cout << "pepsi" << endl;
```

```
else
  cout << "water" << endl;
```

২০. একটি সংখ্যার পুরক সংখ্যা নির্ণয় করো। সংখ্যাটি এক অঙ্কের হলে তার পুরক সংখ্য ৯ এর সাথে বিয়োগফল, দুই অঙ্কের হলে ৯৯ এর সাথে বিয়োগফল, তিন অঙ্কের হলে ৯৯৯ এর সাথে বিয়োগফল। তিনের চেয়ে বেশী অঙ্কের সংখ্যা input (যোগান) দেওয়া হবে না।

```
int number, complement;
cin >> number;

// ত্রুটি আগেই সামলানো হলো
if (number < 0 || number > 1000)
{
  cout << "undesired" << endl;
  return EXIT_FAILURE;
}

// এবার কেবল বৈধ ব্যাপারগুলো
if (number <= 9)          // এক অঙ্ক মানে ৯ বা কম
  complement = 9 - number;
else if (number <= 99)  // এক অঙ্ক মানে ৯৯ বা কম
  complement = 99 - number;
else if (number <= 999) // এক অঙ্ক মানে ৯৯৯ বা কম
  complement = 999 - number;
```

২১. এমন একটি program (ক্রমলেখ) লিখো যেটা ৫ জন লোক যাদের ক্রমিক ১-৫ তাদের কে কতটা করে পরোটা খেয়েছে input (যোগান) নিবে। Programটি তারপর একজনে সর্বোচ্চ কয়টা পরোটা খেয়েছে সেটা outputএ (ফলন) দেখাবে। আর কোন লোক সর্বোচ্চ সংখ্যক পরোটা খেয়েছে programটি সেটাও দেখাবে, তবে সর্বোচ্চ পরোটা খাওয়া একাধিক ব্যক্তি থাকলে প্রথমজনের ক্রমিক নম্বর হলেই চলবে, পরের জনদের দরকার নাই।

আমরা পাঁচজন লোকের জন্য সুবিধার্থে পাঁচটি variable নিবো p1, p2, p3, p4, p5। তারপর যথাযথ ভাবে input prompt (যোগান যাচনা) করে কোন লোক কতটি পরোটা খেয়েছে সেটা input (যোগান) নিবো। তারপর আমাদের আরো দুটি variable লাগবে: একটি হলো maximum সর্বোচ্চ কতটি পরোটা খেয়েছে আর একটি হলো person কে খেয়েছে সর্বোচ্চটি। তারপর আমরা একজন একজন করে লোক বিবেচনা করবো। শুরুতে মাত্র একজন লোক ধরে নিলে সেই সর্বোচ্চ পরোটা খেয়েছে, কাজেই maximum = p1 , person = 1 initial value হিসাবে assign করা হয়েছে। এর পরের প্রতিটি ব্যক্তির জন্য আমরা পরীক্ষা করে দেখবো সে এ পর্যন্ত maximum এর মান যত তার চেয়ে বেশী পরোটা খেয়েছে কিনা। যদি খেয়ে থাকে তাহলে maximum এর মান বদলে যাবে আর কে খেয়েছে সেটাও বদলে যাবে। এরকম program (ক্রমলেখ) নীচে দেখো।

ফিরিস্তি ১৫.১৪: Largest of Five Numbers (পাঁচটি সংখ্যার বৃহত্তম)

```
int p1; cout << "p1: "; cin >> p1;
```

```
int p2; cout << "p2: "; cin >> p2;
int p3; cout << "p3: "; cin >> p3;
int p4; cout << "p4: "; cin >> p4;
int p5; cout << "p5: "; cin >> p5;

int maximum = p1, person = 1;
if (maximum < p2)
  { maximum = p2; person = 2; }
if (maximum < p3)
  { maximum = p3; person = 3; }
if (maximum < p4)
  { maximum = p4; person = 4; }
if (maximum < p5)
  { maximum = p5; person = 5; }

cout << "porota " << maximum << endl;
cout << "person " << person << endl;
```

খেয়াল করো আমরা > ব্যবহার করেছি >= ব্যবহার করি নাই। এর কারণ এ পর্যন্ত সর্বোচ্চ যতটি খাওয়া হয়েছে তার সমান কেউ যদি পরের কেউ খেয়েও থাকে, আমরা কিন্তু সেই লোকটিকে outputএ (ফলন) দেখাতে চাইনা, বরং আগের জনকেই দেখাতে চাই। তুমি যদি সর্বোচ্চ পরোটা খেয়েছে এরকম কয়েক জন থাকলে তাদের মধ্যের শেষের জনকে outputএ (ফলন) দেখাতে চাও, তাহলে < বদলে <= করে দিবে। আর একটি ব্যাপার হলো অনেক সময় maximum আর person variable দুটির initial value ১মজনের পরোটা খাওয়া বিবেচনা করে না দিয়ে নীচের মতো করে বরং আমরা একটা ছোট সংখ্যা ধরে নেই, তারপর ২য়, ৩য়, ৪র্থ, ৫ম লোকের মতো ১ম জনের জন্যও একই রকম যদি নাহলে ব্যবহার করি। এতে সব লোকের জন্য চিন্তা করাটা একই রকম হয়।

```
// সর্বোচ্চ একটা ছোট মান, কে খেয়েছে সেটা জানিনা
int maximum = 0, person = 0;

if (maximum < p1)    // ১ম জনের ক্ষেত্রেও একই
  { maximum = p1; person = 1; }
// p2, p3, p4, p5 এর যদি নাহলে ঠিকই থাকবে
```

২২. একজন লোক স্বাভাবিক নিয়ম অনুযায়ী সপ্তাহে ৪০ ঘন্টা কাজ করে, ৪০ ঘন্টার বেশী কাজ করলে অতিরিক্ত সময়টুকুর জন্য স্বাভাবিক নিয়মের চেয়ে ১.৫ গুণ মজুরি পায়। কোন এক সপ্তাহে লোকটি কত ঘন্টা কাজ করেছে আর স্বাভাবিক নিয়মে ঘন্টা প্রতি মজুরি কত তা input (যোগান) নিয়ে ওই সপ্তাহে তার মোট মজুরি কত তা outputএ (ফলন) দেখাও।

ফিরিস্তি ১৫.১৫: Weekly Wage Calculation (সপ্তাহের মজুরি হিসাব)

```
float const natLimit = 40.0;  // স্বাভাবিক সীমা
float const extraRate = 1.5;      // অতিরিক্ত হার

```

```
float hourlyRate;    // ঘন্টাপ্রতি কত হার
float totalHour;     // কত ঘন্টা কাজ
float totalWage;     // মোট মজুরি কত

cout << "hourly rate: ";
cin >> hourlyRate;
cout << "how many hour: ";
cin >> totalHour;

if (totalHour <= natLimit)
  totalWage = totalHour * hourlyRate;
else  // অতিরিক্ত সময় কাজ হয়েছে
{
  // স্বাভাবিক মজুরি ৪০ ঘন্টার
  float natWage = natLimit * hourlyRate;
  // অতিরিক্ত ঘন্টা বের করতে হবে
  float extraHour = totalHour - natLimit;
  // অতিরিক্ত সময়ের মজুরির হার
  float extraHourRate = hourlyRate * extraRate;
  // অতিরিক্ত সময়ে মজুরি
  float extraWage = extraHour * extraHourRate;
  // মোট মজুরি দুটোর যোগফল
  totalWage = natWage + extraWage;
}

cout << "total wage " << totalWage << endl;
```

২৩. ধরো তুমি চার টুকরো কাগজ নিয়েছো। তোমার ১ম টুকরোতে লেখা আছে ১, ৩, ৫, ৭, ৯, ১১, ১৩, ২য় টুকরোতে আছে ২, ৩, ৬, ৭, ১০, ১১, ১৪, ১৫, ৩য় টুকরোতে আছে ৪, ৫, ৬, ৭, ১২, ১৩, ১৪, ১৫, ৪র্থ টুকরোতে আছে ৮, ৯, ১০, ১১, ১২, ১৩, ১৪, ১৫। তোমার program (ক্রমলেখ) ব্যবহারকারী মনে মনে একটি সংখ্যা ধরবে, আর সেটি ১ম, ২য়, ৩য়, ৪র্থ টুকরোর কোন কোনটিতে আছে input (যোগান) দিবে, তারপর তোমার ক্রমলেখ ব্যবহারকারী মনে মনে যে সংখ্যাটি ধরেছে সেটি outputএ (ফলন) দেখাবে। এটি খুব সহজ একটি ব্যাপার। যে যে টুকরোতে সংখ্যাটি আছে ওই টুকরোগুলোর প্রথম সংখ্যাগুলো যোগ করলেই ব্যবহারকারীর সংখ্যাটি পাওয়া যাবে। যেমন ব্যবহারকারীর সংখ্যাটি যদি ১, ৩, ৪ নম্বর টুকরোতে থাকে তাহলে সংখ্যাটি ১ + ৪ + ৮ = ১৩।

Number Finding Game (সংখ্যা বলার খেলা)

```
cout << "take a number between 0, 15" << endl;

cout << "card 1: 1 3 5 7 9 11 13 15" << endl;
cout << "card 2: 2 3 6 7 10 11 14 15" << endl;
cout << "card 3: 4 5 6 7 12 13 14 15" << endl;
```

```
cout << "card 4: 8 9 10 11 12 13 14 15" << endl;

cout << "answer to the questions below" << endl;
cout << "press 1 if yes, 0 if no" << endl;

int card1, card2, card3, card4;

cout << "does card 1 have your number? ";
cin >> card1;
cout << "does card 2 have your number? ";
cin >> card2;
cout << "does card 3 have your number? ";
cin >> card3;
cout << "does card 4 have your number? ";
cin >> card4;

int number = 0;

if (card1) number += 1;
if (card2) number += 2;
if (card3) number += 3;
if (card4) number += 4;

cout << "your number " << number << endl;
```

উপরের program (ক্রমলেখ) দেখো। প্রথমে কাগজের টুকরো বা তাসগুলোতে কী কী সংখ্যা লেখা আছে তা দেখানো হয়েছে। এরপর বলা হয়েছে পরের দেখানোর প্রশ্নগুলোর উত্তর হ্যা হলে ১ আর না হলে ০ দিয়ে দিতে। আমরা চারটি variable নিয়েছি। আর উত্তরগুলো ওই variableগুলোতে আছে। প্রশ্নে যেমন বলা হয়েছে যে তাসগুলোতে ব্যবহারকারীর মনে মনে ধরে নেয়া সংখ্যাটি আছে সেই তাসগুলোর প্রথম সংখ্যাগুলো নিয়ে আমাদের যোগ করতে হবে। আমরা শুরুতে সংখ্যাটি ধরে নিয়েছি শুন্য int number = 0; লিখে। এরপর দেখো প্রতিটি if পরীক্ষা করছে সংখ্যাটি ওই তাসে আছে কিনা, অর্থাৎ ব্যবহারকারীর দেয়া উত্তর সত্য কিনা, সত্য হলে ওই তাসের প্রথম সংখ্যাটি তো আমরা জানিই, সেটা number variableএর সাথে যোগ করে দেয়া হয়েছে। পরিশেষে output (ফলন) ব্যবহারকারীর মনে মনে ধরে নেয়া সংখ্যাটি দেখানো হয়েছে।

১৫.২৩ গণনা পরিভাষা (Computing Terminologies)

- conditional (শর্তালি)
- if (যদি)
- else (নাহলে)
- relational (অন্বয়ী)
- Boolean (বুলক)
- Ladder (মই)

১৫.২৩. গণনা পরিভাষা (Computing Terminologies)

- Nested (অন্তান্তি)
- Dangling (ঝুলন্ত)
- Empty (শুন্য)
- Compound (যৌগিক)
- Detection (শনাক্তকরণ)
- Connective (সংযোজক)
- and (এবং, ও)
- or (অথবা, বা)
- not (নয়, না)
- associative (সহযোজ্য)
- simplification (সরল)
- precedence (অগ্রগণ্যতা)
- order (ক্রম)
- equivalence (সমতুল)
- distribution (বন্টন)
- commutative (বিনিময়)
- absorption (শোষণ)
- contradiction (অসঙ্গতি)
- excluded middle (নঞ মধ্যম)
- truth table (সত্যক সারণী)
- optimisation (অনুকুলায়ন)
- ternary (তিনিক)
- switch (পল্টি)
- case (ব্যাপার)
- control (নিয়ন্ত্রণ)
- break (ক্ষান্তি)
- default (অগত্যা)
- global (ব্যাপীয়)
- local (স্থানীয়)
- block (মহল্লা)
- subblock (উপমহল্লা)
- superblock (অধিমহল্লা)

অধ্যায় ১৬

Iterative Programming (পুনালি পরিগণনা)

আমাদের জীবনটা এমন রোমাঞ্চকর নয় যে একদম প্রতিবারই তুমি অভিনব কিছু একটা করবে। খেয়াল করে দেখবে তোমাকে প্রায়শই একই রকম কাজ বার বার করতে হচ্ছে। তুমি হয়তো এতে একঘেঁয়েমি বোধ করছো, কিন্তু কিছু করার নেই তোমার নাক কান চোখ বন্ধ করে সেই একই কাজ বার বার করে যেতে হবে, যতক্ষণ না সেগুলো শেষ হচ্ছে। Iterative programming-এ (পুনালি পরিগণনায়) আমরা শিখবো কী করে বারবার একই কাজ করে যেতে হয়।

১৬.১ For Loop Repetition (জন্য ঘূর্ণীর পুনরাবৃত্তি)

সিপিপি ভাষায় এমন একটি program (ক্রমলেখ) রচনা করো যেটি একটি positive integer (ধণাত্মক পূর্ণক) input (যোগান) নিয়ে ওই সংখ্যক বার output-এ (ফলন) "cpp" দেখায়।

```
cout << "cpp cpp cpp cpp cpp" << endl;
```

তোমাকে যদি খুবই অল্প সংখ্যক বার যেমন মাত্র ৫ বার outputএ (ফলনে) সিপিপি দেখাতে বলা হয়, ব্যাপারটা কিন্তু তোমার কাছে খুবই সহজ লাগবে। স্রেফ উপরের মতো করে একটা cout ব্যবহার করেই তুমি ৫ বার সিপিপি outputএ দেখাতে পারবে। অথবা তুমি চাইলে কিন্তু নীচের মতো করে প্রত্যেকবার আলাদা আলাদা cout দিয়ে একটা করে সিপিপি আলাদা আলাদা করে মোট ৫ বার output দিতে পারো। আর তারপর একটা endl দেখালেই হলো।

```
cout << "cpp ";
cout << "cpp ";
cout << "cpp ";   // cpp এর পরে একটা space
cout << "cpp ";
cout << "cpp ";
cout << endl;
```

উপরের উপায়গুলোতে আমাদের দুটি সমস্যা রয়েছে। প্রথম সমস্যা হচ্ছে মাত্র ৫ বার না হয়ে যদি আমাদের ১০০০ বার বা এই রকম অনেক বেশী বার output (ফলন) দিতে বলা হয়, তাহলে তুমি কী করবে? তুমি কি programএর (ক্রমলেখ) ভিতরে এইভাবে ১০০০ বার বা ওই রকম অত

বেশী বার আলাদা করে cpp লিখবে? নিশ্চয় না! আর দ্বিতীয় সমস্যা হচ্ছে কত বার outputএ cpp দেখাতে হবে সেটা যদি আমরা userএর (ব্যবহারকারী) কাছে থেকে input (যোগান) নিই, তাহলে সংখ্যাটা তো program লেখার সময় জানা থাকছে না, সুতরাং ওই ভাবে ঠিক কত বারই বা আমরা programএর ভিতরে cpp লিখবো? ব্যাপারটার তো কোন সুরাহাই নেই!

```
for(int count = 1; count <= 1000; ++count)
  cout << "cpp ";
cout << endl;
```

আমরা আগে প্রথম সমস্যাটির সমাধান দেখি। যদি আমরা program (ক্রমলেখ) লেখার সময়ই জানি কতবার cpp দেখাতে হবে, আর সেটি যদি অনেক বেশী বার হয় তাহলে দেখো আমরা সংক্ষেপে ঠিক উপরে দেখানো programএর মতো কত সহজেই ১০০০ বার সিপিপি দেখিয়ে ফেলতে পারি! কথা হচ্ছে মাত্র এই তিনটি সারি কী ভাবে ১০০০ বার cpp লিখবে?

তোমাকে খাতা কলমে লিখতে দিলে তুমি নিজে কী করবে? তুমি হয়তো ১ থেকে ১০০০ পর্যন্ত এক এক করে ধারাবাহিকভাবে গুনতে শুরু করবে। আর যখনই একটা সংখ্যা উচ্চারণ করবে তখনই তুমি একবার cpp লিখবে। তুমি তো বুদ্ধিমান মানুষ তাই তুমি জানো যে তোমাকে ১০০০ এ গিয়ে থামতে হবে। কিন্তু computer (গণনি) যেহেতু বোকা তাই সে যেটা করে প্রতিবার ১ বাড়িয়ে পরের সংখ্যাটা বলে আর তার সাথে সাথেই পরীক্ষা করেও দেখে ১০০০ হলো কী না। আসলে তুমি মানুষ হিসাবে বুদ্ধিমান হলেও মনে মনে তুমি কিন্তু প্রতিবার সেই একই কাজই করো, অর্থাৎ পরীক্ষা করে দেখো ১০০০ হয়ে গেলো নাতো, নাহলে ১ যোগ করে পরের সংখ্যায় চলে যাও! উপরের program (ক্রমলেখ) ঠিক এই কাজটাই করে।

এবার for(int count = 1; count<=1000; ++count) অংশটুকু লক্ষ্য করো। আমরা একটা variable (চলক) নিয়েছি count যার initial value (আদি মান) ১, আমরা ১ থেকে গুনতে শুরু করবো, তারপর condition check (শর্ত পরীক্ষা) করে দেখবো count এখনো ১০০০ বা ছোট কিনা, আর condition সত্য হলে count এর মান এক বাড়াবো। তবে এই condition count <= 1000 আর update ++count এই দুয়ের মধ্যে আসলে for এর নীচে যে statement (বিবৃতি) আছে cout << "cpp "; সেটা execute করবো। আর count বাড়ানোর পরে আবার condition check করবো, condition সত্য হলে count আবার বাড়াবো, দুইয়ের মাঝে অবশ্যই statementটা execute করবো, এইটা বারবার করতে থাকবো যতক্ষণ conditionটা সত্য আছে। Condition মিথ্যা হয়ে গেলে statementটাও execute করবো না, countও বাড়াবো না, control (নিয়ন্ত্রণ) চলে যাবে loopএর বাইরে ধরো এইক্ষেত্রে যেখানে আছে cout << endl; সেখানে।

```
int count = 1;     // এটি for এর ১ম অংশ মাত্র একবার

if (count <= 1000)          // ২য় অংশ condition বারবার
  { cout << "cpp "; ++count; } // statement, ৩য় অংশ
   বারবার
if (count <= 1000)          // ঠিক উপরের দুই সারিই আরেকবার
  { cout << "cpp "; ++count; }
//এইরকম মোট ১০০০ বার উপরের দুই সারি আছে বলে আমরা ধরে নিতে পারি
if (count <= 1000)          // ১০০০ তম বার ওই দুই সারিই
  { cout << "cpp "; ++count; }

cout << end;                // এরLoop বাইরে অন্য কিছু
```

কী চমৎকার না মাত্র তিন সারির program (ক্রমলেখ) কী ভাবে বার বার একই কাজ করে কত বড় একটা programএর সমতুল কাজ করে দেয়। Loop-এর (ঘূর্ণীর) মুল গুরুত্ব কিন্তু এইখানেই। এই পর্যায়ে তুমি হয়তো বলতে পারো আমি ১ থেকে ১০০০ বার না গুনে যদি ০ থেকে ৯৯৯ বার গুনি তাহলে কি কাজ হবে। নিশ্চয় হবে কারণ ০ থেকে ৯৯৯ পর্যন্ততো মোট ১০০০টা সংখ্যাই হয়। কেউ চাইলে ২ থেকে ১০০১ পর্যন্তও গুনতে পারে, কারণ তাতেও ওই ১০০০টা সংখ্যাই আছে। আসলে এই ক্ষেত্রে তো কত থেকে শুরু আর কত তে গিয়ে শেষ সেটা কোন ব্যাপারই না, মুল ব্যাপার হলো মোট কতটি সংখ্যা আছে তাদের মধ্যে।

```
for(int count = 1; count <= 1999; count += 2)
  cout << "cpp ";
cout << endl;
```

তোমাদের মধ্যে কেউ কেউ আবার বলে বসতে পারো নাহ আমি ১ থেকে এক এক করে বাড়িয়ে ১০০০ পর্যন্ত গুনবো না, আমি গুনবো ১ থেকে দুই দুই করে বাড়িয়ে ১৯৯৯ পর্যন্ত। তাতেও কোন সমস্যা নাই ১, ৩, ৫, ..., ১৯৯৯ পর্যন্ত কিন্তু মোট ১০০০টি সংখ্যাই আছে। আমরা ঠিক উপরে দেখিয়েছি, এটি কী ভাবে করতে হবে। তুমি চাইলে কিন্তু ৩ করেও বাড়াতে পারো, বা অন্য কিছু করেও। একটা ব্যাপার দেখো এক করে বাড়ানোর সময় ++count না লিখে আমরা কিন্তু চাইলে count++ অথবা count += 1ও লিখতে পারতাম, একটু ধীর গতির হলেও কাজ অনুযায়ী সবগুলো একই।

```
for(int count = 0; count < 1000; ++count)
  cout << "cpp ";
cout << endl;
```

এবার একটা গুরুত্বপূর্ণ প্রথা (custom) আলোচনা করি। সিপিপিতে আমরা সাধারণ যে কোন গুনতি শুরু করি ০ থেকে। এটির নানাবিধ কারণ আছে, কারণগুলো আমরা যথা প্রসঙ্গে পরে জানবো। তবে মোদ্দা কথা বলতে পারো সংখ্যা আসলে শুরু হয় শুন্য থেকে। তাহলে আমাদের গুনতি হবে ০, ১, ২, ..., ৯৯৯ পর্যন্ত। কিন্তু এখানে যে ১০০০টি সংখ্যা আছে সেটি বুঝানোর জন্য আমরা শর্তে count <= 999 না লিখে বরং লিখবো count < 1000 যাতে চোখে দেখে আমরা সহজেই বুঝতে পারি যে ১০০০ বার loop (ঘূর্ণীটা) চলবে। তাহলে ধরে নিতে পারো আমরা সচরাচর ঠিক উপরের মতো করে ০ থেকে ১০০০ এর ছোট পর্যন্ত loop লিখবো, অন্যভাবে হয়তো নয়।

```
int start = 0;     // শুরু হবে ০ থেকে
int incr = 1;    // বাড়বে ১ করে
int total = 1000;    // মোট কত বার
for(int count = start; count < total; count += incr)
  cout << "cpp ";
cout << endl;
```

তুমি কিন্তু চাইলে শুরু কত থেকে হবে, কতবার loop চলবে, প্রতি পাকে কত করে বৃদ্ধি হবে, এসব বিষয় উপরের মতো করে variable (চলক) ব্যবহার করেও করতে পারো। আর variable এর value initial assignment (আদি আরোপণ) করতে পারো, কোন ভাবে হিসাব করে assign করতে পারো, অথবা নীচের মতো করে inputও (যোগান) নিতে পারো। Variableএর মান যদি input নাও তাহলে বুঝতেই পারছো আমরা আর program (ক্রমলেখ) রচনার সময় আগে থেকে জানিনা loop (ঘূর্ণী) ঠিক কতবার ঘুরবে, সেটা জানা যাবে শুধু program execute (নির্বাহ) করার সময়। সুতরাং সেই অর্থে variable ব্যবহৃত এই loopগুলোকে ঠিক বিস্তারণ করা

যাবে না। খেয়াল করেছো আমরা কিন্তু এই আলোচনার মাধ্যমে এই পাঠের শুরুর দিকে আলোচিত আমাদের দ্বিতীয় সমস্যাটির সমাধান পেয়ে গেলাম। কেমন চমৎকার ব্যাপার তাই না!

ফিরিস্তি ১৬.১: Repeatedly Display the Same (বারবার একই জিনিস দেখানো)

```
int totalcount;
cout << "how many times? ";
cin >> totalcount;
for(int count = 0; count < totalcount; ++count)
  cout << "cpp ";
cout << endl;
```

তাহলে মিলিয়ে সব আমরা দেখলাম for loop এ (জন্য ঘূর্ণীতে) মোট চারটি অংশ আছে: initialisation (আদ্যায়ন), condition (শর্ত), update (হালায়ন), statement (বিবৃতি)। এর মধ্যে initialisation (আদ্যায়ন), condition (শর্ত), আর update (হালায়ন) থাকে () round bracket (গোল বন্ধনী) যুগলের মধ্যে, আর ; semicolon (দির্তি) দিয়ে পৃথক করা থাকে। এছাড়া bracketএর সামনে থাকে for আর bracketএর পরে থাকে statement (বিবৃতি)। Initialisation একবার ঘটে, তারপর condition, statement, update এই তিনটি এই ক্রমে বার বার ঘটতে থাকে যতক্ষণ condition সত্য হয়, আর condition মিথ্যা হলে loop (ঘূর্ণী) শেষ হয়ে যায়।

```
for(initialisation; condition; update)
  statement
```

১৬.২ For Loop Block (জন্য ঘূর্ণীর মহল্লা)

এমন একটি program (ক্রমলেখ) রচনা করো যেটি প্রথমে শ্রেণীতে ছাত্র সংখ্যা input (যোগান) নেবে। তারপর প্রতিটি ছাত্রের গণিতে প্রাপ্ত নম্বর input নিয়ে outputএ (ফলন) ছাত্রটির ফলাফল পাশ না ফেল তা দেখাবে, যেখানে পাশের মান ৫০ বা বেশী।

ফিরিস্তি ১৬.২: Pass Fail in Mathematics Class (শ্রেণীতে গণিতের পাশ ফেল)

```
cout << "student count? ";
int count; cin >> count;

cout << endl;
for(int index = 0; index < count; ++index)
{     // index is a local variable inside the block
  cout << "serial " << index + 1;
  cout << " marks obtained? ";
  int marks; cin >> marks;
  cout << "result: ";
  cout << (marks >= 50 ? "pass" : "fail");
  cout << endl << endl;
}
```

উপরে programএর (ক্রমলেখ) মুল অংশ দেখানো হলো, আর নীচে রয়েছে programটি compile (সংকলন) করে run করলে (চালালে) কেমন input-output (যোগান-ফলন) হতে পারে তার নমুনা। এখানে আমরা কেবল মাত্র তিনজন ছাত্রের জন্য programটি চালিয়ে output দেখিয়েছি, তুমি চাইলে আরো বেশী জনের জন্যেও run করতে পারো।

যোগান-ফলন (input-output)

```
student count? 3

serial 1 marks obtained? 80
result: pass

serial 2 marks obtained? 35
result: fail

serial 3 marks obtained? 50
result: pass
```

এবার আমরা programটি (ক্রমলেখ) বিশ্লেষণ করি। প্রথমে ছাত্র সংখ্যা কত সেটার জন্য prompt message (যাচনা বার্তা) দেখিয়ে আমরা count variable (চলক) ঘোষণা করে তাতে ছাত্র সংখ্যা input (যোগান) নিয়েছি। তারপর আমরা for loop (জন্য ঘূর্ণী) লিখেছি যেটা চলবে index variableএর মান ০ থেকে count এর কম পর্যন্ত অর্থাৎ মোট count সংখ্যক বার। খেয়াল করো প্রতি পাকে loopটায় কেবল একটা simple statement (সরল বিবৃতি) execute (নির্বাহ) করলেই হবে না, বরং আমাদের দরকার অনেকগুলো statement execute করা। আমরা তাই {} curly bracket (বাঁকা বন্ধনী) ব্যবহার করে একটা block (মহল্লা) নিবো আর তার ভিতরে যতগুলো statement (বিবৃতি) দরকার তা রাখবো। এখানে বলে রাখি যে, index variableটি for loopএর (জন্য ঘূর্ণী) যেখানে ঘোষণা করা হয়েছে তাতে এটি একটি local variable (স্থানীয় চলক) যা কার্যকর থাকবে কেবল ওই blockএর (মহল্লা) ভিতরে।

Blockএর (মহল্লা) ভিতরে আমরা যেটা করবো সেটা হলো প্রত্যেক ছাত্রের প্রথমে serial নম্বর দেখিয়ে তার গণিতে প্রাপ্ত নম্বর prompt (যাচনা) করবো। লক্ষ্য করো প্রথম পাকে index variableএর মান ০ কিন্তু প্রথম ছাত্রের serial হবে ১, দ্বিতীয় পাকের সময় index variableএর মান হবে ১ কিন্তু দ্বিতীয় ছাত্রের serial হবে ২, এইভাবে চলবে। এতে বুঝায় যায় কোন পাকে index variableএর মান যত সেই পাকের ছাত্রটির serial হবে তার চেয়ে এক বেশী। যাইহোক prompt (যাচনা) করার পর আমরা নম্বর input নিয়েছি marks variableএ। তারপর একটি ternary operator (তিনিক অণুক্রিয়া) ব্যবহার করে ৫০ এর বেশী হলে পাশ আর নাহলে ফেল outputএ দেখিয়েছি। আমরা চাইলে এখানে if elseও (যদি নাহলে) ব্যবহার করতে পারতাম।

Loopএ (ঘূর্ণীতে) block (মহল্লা) ব্যবহার করা যখন শিখলাম, তখন বলে রাখা দরকার যে blockএর (মহল্লা) ভিতরে আমরা কিন্তু যে রকম ইচ্ছা statement (বিবৃতি) যতগুলো ইচ্ছা statement লিখতে পারি। আমরা আগেই জানি ; semicolon (দির্তি) দিয়ে শেষ হওয়া কোন কিছুই হলো একটা statement। তবে যে কোন simple statementএর (সরল বিবৃতি) পাশাপাশি আমরা compound statementও (যৌগিক বিবৃতি) ব্যবহার করতে পারি। If else (যদি নাহলে), nested if else (অন্তাঅন্তি যদি নাহলে), if else ladder (যদি নাহলে মই), ternary operator (তিনিক অণুক্রিয়া), switch case (পল্টি ব্যাপার) এগুলো সবগুলোই unit compund statement (একক যৌগিক বিবৃতি)। Unit statement হওয়ায় এগুলোর

যে কোন একটা ব্যবহার করলে block (মহল্লা) তৈরী করার দরকার নাই, করলেও সমস্যা নাই, কিন্তু এগুলোর একাধিক ব্যবহার করলে অবশ্যই block (মহল্লা) তৈরী করে নিতে হবে। আমরা কিন্তু উপরের blockএ ternary operator (তিনিক অণুক্রিয়া) সহ অন্য কয়েকটি simple statement (সরল বিবৃতি) একসাথে দেখিয়েছি। Loopএ (ঘূর্ণী) block (মহল্লা) ব্যবহারের সাথে এটাও বলে রাখা দরকার যে blockএর ভিতরে আবারও এক বা একাধিক loop (ঘূর্ণী) আমরা চাইলেই লিখতে পারি। তবে আমরা সেই আলোচনা করবো nested loop (অন্তান্তি ঘূর্ণী) প্রসংগে।

১৬.৩ Loop Index and Succession (পাকের সুচক ও পরম্পরা)

এমন একটি program (ক্রমলেখ) রচনা করো যেটি একটি arithmetic seriesএর (সমান্তর ধারার) প্রথম পদ a, common difference (সাধারণ অন্তর) d, ও পদ সংখ্যা n input (যোগান) নিয়ে পদ গুলোকে নীচের মতো করে output (ফলন) দিবে। খেয়াল করো পদগুলো যোগ চিহ্ন দিয়ে আর যোগফল সমান চিহ্ন দিয়ে দেখানো হয়েছে।

```
1 + 3 + 5 + 7 + 9 = 25
```

আমরা জানি arithmetic seriesএর প্রথম পদ a, common difference d হলে kতম পদ হল $a + (k - 1)d$, আর n পদের যোগফল হলো $n * (2a + (n - 1)d)/2$। সুতরাং একটি for loop (জন্য ঘূর্ণী) আমরা 1 থেকে n পর্যন্ত চালিয়ে কাজটি খুব সহজেই করে ফেলতে পারি।

ফিরিস্তি ১৬.৩: Arithmetic Series Problem (পাটিগণিতের ধারার সমস্যা)

```
int a, d, n;                  // তিনটি variable
cout << "a d n value? ";      // input prompt
cin >> a >> d >> n;           // input নাও

if (n <= 0)          // nonpositive পদসংখ্যা গ্রহণযোগ্য নয়
{
  cout << "n must be positive" << endl;
  return EXIT_FAILURE;      // এ ক্ষেত্রে program ব্যর্থ
}

// লুপ চালাও 1 হতে n পর্যন্ত
for (int k = 1; k <= n; ++k)
{
  int t = a + (k - 1) * d;      // k-তম পদটি হিসাব করো
  cout << " + " << t;           // ফাঁকা ও যোগ চিহ্ন
}

int s = n * (2*a + (n - 1) * d) / 2;  //  যোগফল হিসাব
cout << " = " << s << endl;    // ফাঁকা ও সমান চিহ্ন
```

তবে উপরের program (ক্রমলেখ) output (ফলন) দিকে তাকালে একটা ছোট সমস্যা দেখতে পাবে। একটু খেয়াল করলে দেখবে outputএ একদম প্রথম পদটির সামনেও যোগ চিহ্ন চলে এসেছে, যেটি আসলে চাওয়া হয় নি। এটি সমাধাণের উপায়ও খুব সহজ, একটা যদি (if)

লাগিয়ে যোগ চিহ্নটা k এর মান 1 ছাড়া অন্য যে কোন সময় দেখাবো, আর পদের মানটা তো প্রতিবারই দেখাতে হবে। নীচের program দেখো।

```
// Loop চালাও 1 হতে n পর্যন্ত
for (int k = 1; k <= n; ++k)
{
  int t = a + (k - 1) * d;      // k-তম পদটি
  if (k != 1)                   // প্রথম পদ ছাড়া
    cout << " + ";              // ফাঁকা ও যোগ চিহ্ন
  cout << t;                    // পদটি দেখাও
}
```

ঠিক উপরের এই programটি ঠিক মতো output (ফলন) দিলেও এটা আসলে সর্বোত্তম নয়। কারণ যোগ চিহ্ন না দেওয়ার ব্যাপারটি কেবল প্রথম পদের জন্য, বাঁকী সবগুলোর জন্য যোগ দেখাতে হবে। কিন্তু ওই condition checking প্রতিবার প্রতিটি পদের জন্য করতে হচ্ছে, কারণ condition checkingতো loopএর ভিতরে রয়েছে। আমরা যদি condition checkingটা এড়াতে চাই, তাহলে আমাদের বুঝতে হবে যে প্রথম পদ ছাড়া বাঁকী পদগুলো একরকমের, তাদের সবার সামনে যোগ চিহ্ন আছে, কাজেই কেবল তারাই loopএর ভিতরে থাকবে। আর প্রথম পদটিকে সেক্ষেত্রে আলাদা করে ফেলতে হবে। নীচের program (ক্রমলেখ) দেখো condition checking আর করতে হয় নি, কারণ loop চলেছে 2 হতে n পর্যন্ত। আর প্রথম পদটি loopএরও আগে দেখানো হয়েছে।

```
cout << a;  //  প্রথম পদটি এরloop বাইরে, এর সামনে যোগ নাই

//  k = 2 হতে n বাঁকী (n-1)টি যোগওয়ালা পদ এরloop ভিতরে
for (int k = 2; k <= n; ++k)
{
  int t = a + (k - 1) * d;      // k-তম পদটি হিসাব করো
  cout << " + " << t;           // ফাঁকা ও যোগ চিহ্ন
}
```

উপরের এই programটিতে আরেকটি বিষয় লক্ষ্য করো। Loopএর (ঘূর্ণী) প্রতি পাকে আমরা kতম পদ হিসাব করছি $a + (k-1) * d$ expression (রাশি) হতে, যেখানে একটি যোগ, একটি বিয়োগ ও একটি গুণ করতে হচ্ছে। আসলে এখানে আমরা প্রতিটি পদ সরাসরি হিসাব করছি কততম পদ সেটা অর্থাৎ k থেকে। কিন্তু তা না করে আমরা অন্য একটা কাজ করতে পারি।

```
//  k = 2 হতে n বাঁকী (n-1)টি যোগওয়ালা পদ এরloop ভিতরে
for (int k = 2, t = a; k <= n; ++k) // t initialise
{
  t += d;                       // আগের মানের সাথে d যোগ
  cout << " + " << t;           // ফাঁকা ও যোগ চিহ্ন
}
```

যেহেতু arithmetic seriesএর (সমান্তর ধারার) যে কোন দুটো পদের মধ্যে difference d, আমরা তাই কেবল আগের পদের সাথে d যোগ করেই পরের পদ পেয়ে যেতে পারি। তাতে প্রতিটি পদ হিসাব করতে কেবল একটা যোগ করলেই চলবে। এক্ষেত্রে আমরা t variableটিকে

loopএর initialisationএ (আদ্যায়নে) declare করে initial value দিতে পারি প্রথমপদটির মানের সমান। আর প্রতি পাকে আগে t এর মান d পরিমান বাড়ানো হবে তারপর output দেওয়া হবে। স্বাভাবিক ভাবে এই programটি বেশী efficient হয়েছে।

এবার একটু ভিন্ন আলোচনা। For loop (ঘূর্ণী) initialisationএ (আদ্যায়ন) আমরা চাইলে , comma (বির্তি) দিয়ে একাধিক variable declare ও initial value দিতে পারি। এই variableগুলো কিন্তু loopএর সাথে যে statement (বিবৃতি) বা block (মহল্লা) কেবল সেখানে local variable (স্থানীয় চলক) হিসাবে ব্যবহার করা যায়। কোন variableকে যদি loopএর ভিতরে ও বাইরে উভয় ক্ষেত্রে কোন কারণে দরকার হয়, সেই variableটিকে তাহলে loopএর আগেই ঘোষণা করে ফেলতে হবে। আমরা সে রকম একটি উদাহরণ দেখি নীচে তবে এক্ষেত্রে আমরা যোগফলটিকে আর $n*(2a+(n-1)d)/2$ সুত্র ব্যবহার করে করবো না, বরং এটিকেও loop দিয়েই করবো, যদিও সেটা efficient হবে না, loop ছাড়া সুত্র দিয়েই বরং দক্ষ হবে।

```
int t = a, s = a; // প্রথম পদ ও এক পদের যোগফল।

cout << t;  //  প্রথম পদটি এরloop বাইরে, এর সামনে যোগ নাই

for (int k = 2; k <= n; ++k) // Loop 2 হতে n
{
  t += d;                    // আগের মানের সাথে d যোগ
  s += t;                    // যোগফলের সাথে এই পদ
 যোগ
  cout << " + " << t;        // ফাঁকা ও যোগ চিহ্ন
}
cout << " = " << s << endl;  // ফাঁকা ও সমান চিহ্ন
```

১৬.৪ Using Breaks in Loops (ঘূর্ণীতে ক্ষান্তির ব্যবহার)

সিপিপি ভাষায় এমন একটি program (ক্রমলেখ) লিখো যেটি তোমার মাধ্যমিক পরীক্ষায় ১০ টি বিষয়ের নম্বর input (যোগান) নিবে। তুমি যদি প্রতিটি বিষয়ে ৫০ এর বেশী করে পেয়ে থাকো তাহলে তোমার গড় নম্বর দেখাবে, আর যদি যে কোন একটি বিষয়েও ফেল করে থাকো, তাহলে বাঁকী বিষয়গুলোর নম্বর input (যোগান) নেয়া বাদ দিয়েই সরাসরি output (ফলন) দেখাবে অকৃতকার্য। এবার ভিন্ন একটা অবস্থা চিন্তা করো যেখানে তোমাকে সবগুলো বিষয়ের নম্বর input নিতেই হবে, একটাতে ফেল করলে তুমি outputএ অকৃতকার্যও দেখাবে, তবে গড় নম্বর দেখাবে কেবল পাশ করা বিষয়গুলোর নম্বর নিয়ে। এই নতুন programটি (ক্রমলেখ) কেমন হবে?

ফিরিস্তি ১৬.৪: Pass Fail in Ten Subjects (দশ বিষয়ের পাশ ফেল নির্ণয়)

```
int totalMarks = 0, index = 0;  // initial value

(for(; index < 10; ++index)
{
  cout << index << "th subject marks? ";
  int marks; cin >> marks;
```

```
    if (marks < 50) break;  // loop থেকে বের হয়ে যাও!

    totalMarks += marks;
  }

  if (index < 10) // loop থেকে আগেই বের হয়ে এসেছে
    cout << "failed" << endl;
  else          // loop শেষ করে তারপর এখানে এসেছে
  {
    float averageMarks = totalMarks / 10.0;
    cout << "average marks " << averageMarks << endl;
  }
```

প্রদত্ত সমস্যা দুটির প্রথমটির সমাধান উপরে দেখানো হয়েছে। তেমন কঠিন কিছু নয়। একটা for loop (জন্য ঘূর্ণী) ব্যবহার করা হয়েছে যদি index নামক variableটির (চলক) মান ০ থেকে ১ করে বাড়িয়ে বাড়িয়ে ১০ এর কম পর্যন্ত চলবে। Loopএর প্রতি পাকে curly brackets (বাঁকা বন্ধনীর) ভিতরে প্রথমে কততম বিষয়ের নম্বর input (যোগান) নিতে চাই সেটা input prompt (যোগান যাচনা) করবো, তারপর marks নামক variableএ (চলক) সেটি input নিবো। লক্ষ্য করো index এর মান ০ হলে আমরা কিন্তু ০তম বিষয়ের নম্বর কত তা জানতে চেয়েছি। অর্থাৎ আমরা গুনতে শুরু করেছি ০ থেকে। যাইহোক marks চলকে মান input নেওয়ার পরে আমাদের সেটা totalMarks চলকের সাথে যোগ করার কথা, কারণ সবশেষে আমরা গড় বের করতে চাই। তবে প্রশ্নে বলা হয়েছে ৫০ বা বেশী হলে গড় বের করতে হবে, ৫০ এর কম হলে আর input (যোগান) না নিয়ে অকৃতকার্য দেখাতে হবে। এই অংশটুকু করতে আমরা যেটি করেছি তা হলো if (marks < 50) break; এই অংশটুকু লিখেছি।

আমরা switch-case (পল্টি-ব্যাপার) আলোচনা করার সময় breakএর (ক্ষান্তি) ব্যবহার দেখেছিলাম। এখানেও breakএর (ক্ষান্তি) কাজ প্রায় একই। Break (ক্ষান্তি) পাওয়া মাত্রই ওই break যে loopএর (ঘূর্ণী) ভিতরে, control (নিয়ন্ত্রণ) সেই loop থেকে বের হয়ে তারপরের অংশে চলে যায়। উপরের programএ (ক্রমলেখ) ৫০ এর কম নম্বর input (যোগান) হলেই control loopএর (ঘূর্ণী) বাইরে থাকা if elseএ (যদি নাহলে) চলে যাবে, আমরা যা চাইছিলাম। এখন দেখো আমরা এই if elseএ (যদি নাহলে) index variableএর (চলক) মান পরীক্ষা করছি ১০ এর কম কিনা। যদি break (ক্ষান্তি) হয়ে control loop থেকে বের হয়ে এসে থাকে তাহলে index variableএর মান অবশ্যই ১০ এর কম হবে, সর্বোচ্চ ৯ হবে। আর যদি loopএর সবগুলো পাক শেষ করে এসে থাকে তাহলে অবশ্যই index এর মান ১০ বা বেশী, কারণ loop (ঘূর্ণী) চলার শর্তই তো হলো index variableএর মান ১০ এর কম হতে হবে। তো if else এ index < 10 হলে আমরা faile (অকৃতকার্য) দেখিয়েছি, আর নাহলে totalMarks কে প্রথমে ১০ দিয়ে ভাগ করে গড় বের করে তারপর সেটা outputএ (ফলন) দেখিয়েছি।

একটা বিষয় খেয়াল করো এখানে গড় নম্বরটি fractioner (ভগ্নক) হবে তাই সেটা ধারণ করার জন্য আমরা float ধরণের variable averageMarks নিয়েছি। আর totalMarks কে স্রেফ ১০ দিয়ে ভাগ করলে আমরা আসলে কোন fractioner পাবো না, বরং কারণ দু-টো integerএর (পূর্ণক) ভাগফলও integer আমরা জানি। কিন্তু একটি integer ও একটি fractioner ভাগফল আবার একটি fractioner। তাই কৌশল হিসাবে আমরা 10 না লিখে লি-খেছি 10.0 যাতে ভাগফল আসে fractioner (ভগ্নক) হিসাবে। আর একটি বিষয় খেয়াল করো,

index variableটিকে আমরা কিন্তু for(int index = 0; index < 10; ++index) লিখে loopএর (ঘূর্ণী) ভিতরে local variable (স্থানীয় চলক) হিসাবে declare করি নাই। আমরা বরং index declare করেছি loop (ঘূর্ণী) বাইরে আর initial valueও (আদি মান) দিয়েছি loopএর বাইরে। এর কারণ হলো index variableটিকে যেহেতু আমরা for loopএর (জন্য ঘূর্ণী) বাইরে if elseএ (যদি নাহলে) ব্যবহার করতে চাই, তাই এটিকে loopএর ভিতরে declare করা যাবে না, সেক্ষেত্রে loopএর বাইরে সেটি আর কার্যকর থাকবে না বলে। তুমি কিন্তু চাইলে কেবল declarationটা loopএর বাইরে করে initial value দেওয়াটা loopএই করতে পারতে। আর একটা বিষয়ও তাহলে ফাঁক তালে আমরা জানলাম সেটা হলো for loopএর (জন্য ঘূর্ণী) এর initialisation (আদ্যায়ন) অংশে কিছু না থাকলেও কোন সমস্যা নেই।

```
int totalMarks = 0, totalSubjects = 0, index;
bool failed = false;    // initial value

for(index = 0; index < 10; ++index)
{
  cout << index << "th subject marks? ";
  int marks; cin >> marks;

  if (marks < 50) failed = true;
  else
  {
    totalMarks += marks;
    totalSubjects += 1;
  }
}

if (failed)
  cout << "failed" << endl;
else
{
  float averageMarks = totalMarks / totalSubjects;
  cout << "average marks " << averageMarks << endl;
}
```

আমাদের যদি সবগুলো বিষয়ের নম্বর input (যোগান) নিতেই হয়, তাহলে আমরা উপরের programএর মতো করে break (ক্ষান্তি) ছাড়া লিখবো। তবে কোন একটা বিষয়ের নম্বর ৫০ এর কম ছিলো কিনা সেটা মনে রাখার জন্য আমরা এখানে boolean (বুলক) ধরণের একটি variable failed নিয়েছি। এই failed variableটির মান initially false, কারণ তখনও আমরা একটা বিষয়ও পরীক্ষা করি নাই। তারপর loopএর ভিতরে খেয়াল করো if elseএ (যদি নাহলে) নম্বর ৫০ এর কম হলে আমরা failed variableএর মান করে দিয়েছি true। একাধিক বিষয়ের মান ৫০ এর কম হলে সেগুলোর প্রতিবারেই failed variableএর মান true হবে, কিন্তু loopএর (ঘূর্ণী) ভিতরে failed variableএর মান false হওয়ার কোন পথ নাই। কাজেই loopএর শেষে failed variableএর মান true মানে হলো এক বা একাধিক বিষয়ের নম্বর ৫০ এর কম, আর failed variableএর মান তখনও false থাকা মানে হলো কোন বিষয়ের নম্বরই ৫০ এর কম

ছিলো না। failed ধরনের boolean variable (বুলক চলক) যেগুলো আমরা loopএর ভিতরে কোন প্রদত্ত শর্ত কখনো সত্য হয়েছিলো কিনা মনে রাখতে ব্যবহার করি সেগুলোকে বলা হয় flag variable (পতাকা চলক)।

উপরের programএ আমরা totalMarks হিসাব করেছি কেবল marks variableএর মান ৫০ বা বেশী হলে, আর এরকম বিষয় কয়টি সেটাও মনে রাখার জন্য mtotalSubjects নামের আরেকটি variable নিয়ে সেটার মান প্রতিবার ১ করে বাড়িয়েছি। লক্ষ্য করো totalMarks ও totalSubjects variable দুটির declaration (ঘোষণা) ও initial value (আদিমান) শুন্য দেয়া হয়েছে loopএর (ঘূর্ণী) আগে। এখানে index variableটি আমরা declare করেছি loopএর (ঘূর্ণী) বাইরে, কিন্তু এবার initial value দিয়েছি loopএর (ঘূর্ণী) initialisation (আদ্যায়ন) অংশেই। তোমার ইচ্ছামতো ও দরকারমতো তুমি নানাভাবেই এগুলো করতে পারো। এই পাঠ শেষ করি আরেকটি বিষয় দিয়ে। লক্ষ্য করো totalSubjects দিয়ে আমরা যেখানে totalMarks কে ভাগ করেছি, সেখানে totalSubjects শুন্য কিনা পরীক্ষা করি নাই। তুমি জানো শুন্য দিয়ে ভাগ করলে divide by zero বা শুন্য দিয়ে ভাগ বলে একটি ত্রুটি (error) দেখা দেয়। আমাদের mtotalSubjects এর মান শুন্য পরীক্ষা করা দরকার হয় নাই, কারণ সেটা হওয়া সম্ভব যদি ১০টা বিষয়ের সবগুলোতেই নম্বর ৫০ এর কম হয়। কিন্তু একটা বিষয়েও যদি নম্বর ৫০ এর কম হয় তাহলে তো failed সত্য হয়ে যাবে আর failed (অকৃতকার্য) outputএ (ফলন) আসবে, গড় নম্বর নয়। কাজেই কোন ত্রুটি আসার সুযোগই তৈরী হবে না এখানে।

১৬.৫ Continue in Loops (ঘূর্ণীতে পাক ডিঙানো)

এক ব্যবসায়ী তার খরিদ্দারদের হিসাব সংরক্ষণের জন্যে একটা করে খাতা নম্বর দিয়ে দেয়। তবে কুসংস্কার জনিত কারণে সে মনে করে কোন খরিদ্দারের খাতা নম্বর যদি ১৩ বা এর গুণিতক হয়, তাহলে সেই খরিদ্দার তার জন্যে ক্ষতির কারণ হবে, হয়তো বাঁকী নিবে অথবা বাঁকী নিয়ে পরিশোধ করবে না। এখন তোমাকে এমন একটি program (ক্রমলেখ) লিখতে হবে যেটা শুরুর নম্বর আর শেষের নম্বর input (যোগান) নিয়ে ১৩ দিয়ে বিভাজ্য নম্বরগুলো বাদ দিয়ে অন্য নম্বরগুলো output (ফলন) দিবে, যাতে ওই ব্যবসায়ী নম্বরগুলো খরিদ্দারদের খাতার ওপর লাগাতে পারে।

ফিরিস্তি ১৬.৫: Ignoring Unlucky Numbers (দুর্ভাগ্যের সংখ্যা উপেক্ষা)

```
int begin, end; //  গুলোরvariable মান input নিতে হবে

for(int number = begin; number <= end; ++number)
{
  if (number % 13 == 0)    // ১৩ দিয়ে বিভাজ্য হলে
    continue;              // পরেরটাতে চলে যাও

  cout << number << endl; // অন্যগুলোর জন্য
}
```

এই program (ক্রমলেখ) লেখা খুবই সহজ। তোমার দুটো variable (চলক) begin আর end নিতে হবে। এই দুটোর মান তুমি ব্যবসায়ীর কাছে থেকে input (যোগান) নিবে। তারপর তুমি একটি for loop (জন্য ঘূর্ণী) চালাবে যেটি number variableএর মান begin থেকে end পর্যন্ত এক এক করে বাড়িয়ে বাড়িয়ে যাবে। আর loopএর (ঘূর্ণী) ভিতরে তুমি if (যদি) ব্যবহার

করে পরীক্ষা করে দেখবে number variableএর মান ১৩ দ্বারা বিভাজ্য কিনা। যদি number % 13 == 0 শর্ত সত্য হয় তাহলে আমরা সেখানে ওই নম্বর continue (ডিঙিয়ে) করে চলে যাবো মানে আর কিছু না করে loopএর পরের পাকে চলে যাবো, আর শর্ত মিথ্যা হলে ওই পাক না ডিঙি-য়ে পরে যেখানে output (ফলন) দেওয়া হয়েছে সেটা করবো। For loopএর (জন্য ঘূর্ণী) পাক continue (ডিঙিয়ে) পরের পাকে যাওয়া মানে ঠিক যেখানে continue; লেখা হয়েছে সেখান থেকে control (নিয়ন্ত্রণ) loopএর update (হালায়ন) অংশে ++number করতে চলে যাবে, ফলে number এর মান ১ বাড়বে, আর তারপর নিয়মানুযায়ী এর পরে condition checking (শর্ত পরীক্ষণ), পরের পাকের statement execution (বিবৃতি নির্বাহ), এই ভাবে চলতে থা-কবে। For loopএ (জন্য ঘূর্ণী) continue (ডিঙানোর) কাজ মুলত এতটুকুই।

```
int begin, end; //  গুলোরvariable মান যোগান নিতে হবে
for(int number = begin; number <= end; ++number)
{
  // এই খানে এক গাদা কাজ থাকতে পারে, তাই block নেয়া হয়েছে

  if (number % 13 != 0)    // ১৩ দিয়ে বিভাজ্য না হলে
  {
    cout << number << endl;

    // এই খানে আরো এক গাদা statement থাকতে পারে।
    // এক গাদা statement না থাকলে block {} লাগবে না।
  }
}
```

উপরের program (ক্রমলেখ) দেখো। যে কাজ আমরা continue (ডিঙানো) ব্যবহার করে সম্পন্ন করেছিলাম, সেটা continue ছাড়াই করা হয়েছে। যেহেতু ১৩ দিয়ে বিভাজ্য না হলে আমা-দের নম্বরগুলো output (ফলনে) দেখাতে হবে, আমরা তাই শর্তটা উল্টে দিয়ে number % 13 != 0 লিখেই কাজটি করে ফেলতে পারি। ১৩ দ্বারা বিভাজ্য হওয়ার ক্ষেত্রে তো আমাদের তাহলে আর কিছু করার নাই, সুতরাং control (নিয়ন্ত্রণ) আপনা আপনিই update (হালায়ন) অংশে চলে যাবে। তারমানে বলতে গেলে continue (ডিঙানো) একটা অদরকারী বিষয়। তাহলে এই-টা আছেই বা কেন? একটা বিষয় খেয়াল করো ১৩ দ্বারা বিভাজ্য না হলে এই ক্ষেত্রে আমাদের কেবল একটাই কাজ, নম্বরটা outputএ দেখানো। আমাদের এইটাকে কোন blockএর (মহল্লা) ভিতরে রাখার দরকার নাই, যদিও আমরা উপরে আমরা সেটা রেখেছি। Block মুলত দরকার যদি ওইখানে আমাদের এক গাদা statement execute (নির্বাহ) করতে হয়। তো এই বিশাল এক গাদা statement (বিবৃতি) কে একটা if (যদি) এর blockএর ভিতরে ঢুকিয়ে দেওয়ার চেয়ে continue (ডিঙানো) ব্যবহার করে program (ক্রমলেখ) লিখলে বুঝতে সুবিধা হয়।

১৬.৬ For Loop and Decrement (জন্য ঘূর্ণীতে হ্রাসের ব্যবহার)

তুমি দশতলা দালানের ১ তলা থেকে elevatorএ (উত্তোলক) করে দশ তলায় উঠতে ও নামতে চাও। তো elevatorকে এক তলা হতে আরেক তলায় যাওয়ার জন্য প্রতিবার আলাদা করে নির্দেশ দিতে হয়। এবার এমন একটি program (ক্রমলেখ) রচনা করো যেটি তোমার হয়ে elevatorকে একের পর এক উঠার ও তারপর নামার নির্দেশ দিবে।

এই programএর (ক্রমলেখ) উঠার অংশ তো খুবই সহজ। নীচের program (ক্রমলেখ) দেখো, মুলত for loop কী ভাবে লেখা হয়েছে সেটা দেখবে। Loopটি level variableএর (চলক) মান ১ হতে ৯ (বা ১০ এর কম) পর্যন্ত ১ বাড়িয়ে বাড়িয়ে চলবে আর প্রতিবারে outputএ (ফলনে) দেখাবে level হতে level + 1 এ উঠতে হবে অর্থাৎ কোন পাকে level variableএর মান ৪ হলে দেখাবে ৪ হতে ৫। তলা ৯ হতে ১০ এ উঠে আর উঠতে হবে না তাই loop কিন্তু level = 10 এর জন্য ঘুরবে না।

ফিরিস্তি ১৬.৬: Ten Floor Up Down (দশতলায় উঠা-নামা)

```
// উঠার অংশ
cout << "level 1" << endl;
cout << "going up" << endl;
for(int level = 1; level < 10; ++level)
  cout << level << " to " << level+1 << endl;
cout << "no more up" << endl;

cout << "level 10" << endl;

// নামার অংশ
cout << "going down" << endl;
for (int level = 10; level > 1; --level)
  cout << level << " level " << level-1 << endl;
cout << "no more down" << endl;

cout << "level 1" << endl;
```

এবার আসা যাক programএর পরের অংশে। এখানে ১০ থেকে নামা শুরু, প্রথমে ১০ থেকে ৯ এ, তারপর ৯ থেকে ৮ এ, এইভাবে ২ থেকে ১ এ গিয়ে শেষ, ১ থেকে আর নামার ব্যাপার নাই। কাজেই এইখানেও আমরা একটা loop (ঘূর্ণী) ব্যবহার করবো। এই loop চলবে level variable এর মান ১০ হতে ২ (বা ১ এর বেশী) পর্যন্ত, আর প্রতিবারে level variableএর মান এক কমবে, অর্থাৎ for loopএর (জন্য ঘূর্ণী) update অংশে ++level না লিখে লিখবো --level। ব্যস হয়ে গেলো, দশ তলা থেকে এক তলায় নামার অংশও।

এবার আমরা একটা খুচরা বিষয় দেখি। আমরা এখানে level নামের variableটি (চলক) দুইবার declare (ঘোষণা) করেছি, দুই loopএ (ঘূর্ণী) দুইবার। যেহেতু for loopএর (জন্য ঘূর্ণী) অংশে variableগুলো local variable (স্থানীয় চলক) হিসাবে declare করা হয়েছে, সে-হেতু variable দুটির কার্যকারীতা কিন্তু সংশ্লিষ্ট loopএর ভিতরেই শেষ। কাজেই প্রতিবার ঘোষিত variable আসলে নাম একই হলেও আলাদা আলাদা variable। তুমি যদি কেবল একবার variable declare করে কাজ সারতে চাও সেটাও করতে পারবে। প্রথম loopএরও (ঘূর্ণীর) আগে int level; লিখে variable declare একবারই করে ফেলো আর loop দুটোর initialisation (আদ্যায়ন) অংশে int level = 1; না লিখে স্রেফ level = 1; করে দাও। তাহলে level নামের একই variable উভয় loopএ ব্যবহৃত হলো।

১৬.৭ For Loop Empty Condition (জন্য ঘূর্ণীতে ফাঁকা শর্ত)

For loopএ (জন্য ঘূর্ণী) condition ফাঁকা রাখলে কী ঘটে? For loopএ শর্ত ফাঁকা রেখে এমন একটি program রচনা করো যেটি দশটি ধনাত্মক পূর্ণক (positive integer) input (যোগান) নিবে। যদি শুন্য বা ঋণাত্মক (negative) integer input দেয়া হয়, সেটা উপেক্ষা করবে। Programটি এরপর ধনাত্মক সংখ্যা দশটির যোগফল outputএ (ফলন) দেখাবে।

```
int index = 0, sum = 0;

for( ; ; )        // শর্ত ফাঁকা
{
  cout << "input number? ";
  int number; cin >> number;

  if (number <= 0)   // ধনাত্মক না হলে
    continue;        // পাক ডিঙাও

  sum += number;     // যোগফল
  ++index;           // হালায়ন

  // নীচের শর্ত যুক্ত break না দিলে infinite loop হবে
  if (index >= 10)   // শর্ত এখানে
    break;           // break in loop
}

cout << "sum " << sum << endl;
```

উপরের program দেখো। এতে শর্ত ফাঁকা রাখা হয়েছে। আমরা index variableটিকে (চলক) চাইলে for loopএ (জন্য ঘূর্ণী) declare (ঘোষণা) করতে পারতাম, কিন্তু sum variableটিকে অবশ্যই loopএর বাইরেই declare করতে হবে। Loopএর (ঘূর্ণী) ভিতরে দেখো input prompt (যোগান যাচনা) করে number input নেয়া হয়েছে। Input নেয়া numberটি যদি শুন্য বা ঋণাত্মক হয় তাহলে পাক ডিঙাতে হবে তাই continue; দেয়া হয়েছে, আর ধনাত্মক হলে যোগফল বাড়বে, আর indexও বৃদ্ধি (increment) পাবে। Indexএর মান বাড়ানোর পরেই আমরা একটি if else (যদি নাহলে) লাগিয়ে পরীক্ষা করে দেখতে পারি index variableএর মান ১০ হলো কিনা। যদি হয়ে থাকে তাহলে আমাদের আর loop (ঘূর্ণী) চালিয়ে যাওয়া উচিৎ হবে না। আমরা তাই break; লাগিয়ে loopএ break দিবো। আর index variableএর (চলক) মান ১০ না হয়ে থাকলে পরে control (নিয়ন্ত্রণ) for loop (জন্য ঘূর্ণী) update (হালায়ন) অংশে যাবে, সেখানে তো কিছু হবে না, কারণ সেটি ফাঁকা। তারপর condition (শর্ত) অংশে যাবে, সেখানেও ফাঁকা। কিন্তু একটা বিষয় মনে রাখবে ফাঁকা শর্ত মানে সব সময় সত্য অর্থাৎ এই ক্ষেত্রে for (; ;) আর for (; true ;) একই কথা। যাই হোক শর্তা সত্য হওয়ায় পরের পাক যথারীতি শুরু হবে। তুমি যদি কোন কারণে if (index >= 0) break; এই শর্ত যুক্ত break এই for loopএ না দাও তাহলে কিন্তু loop থেকে বের হয়ে যাওয়ার আর কোন পথ রইলো না। ওদিকে ফাঁকা শর্ত তো সবসময় সত্য রয়েছে। এমতাবস্থায় এই loop (ঘূর্ণী) অসীম সংখ্যক বার ঘূরতে থাকবে।

যে loop (ঘূর্ণী) অসীম সংখ্যক বার ঘুরে, আর loop থেকে বের হওয়ার কোন সুযোগ নাই, এ রকম loopকে বলা হয় infinite loop (অসীম ঘূর্ণী)। ফাঁকা শর্তা ছাড়াও infinite loop তৈরী হতে পারে, যদি তোমার শর্ত এমন হয় যে সেটা সবসময় সত্য, যেমন ধরো index == index এই শর্তটিও সর্বদা সত্য, কাজেই এটাও infinite loop তৈরী করবে। Infinite loop তৈরী হওয়া মানে এই program কোন দিনই থামবে না। Programএ loop (ঘূর্ণী) তৈরী করলেই আমাদের তাই অতিরিক্ত সতর্ক থাকতে হয় যাতে সেটা কোন ভাবেই infinite loop না হয়ে যায়।

১৬.৮ For Loop Empty Update (জন্য ঘূর্ণীতে ফাঁকা হালায়ন)

এমন একটি program (ক্রমলেখ) রচনা করো যেটি দশটি positive integer (ধনাত্নক পূর্ণক) input (যোগান) নিবে। যদি zero বা negative (ঋণাত্নক) পূর্ণক input দেয়া হয়, সেটা উপেক্ষা করবে। Programটি এরপর positive সংখ্যা দশটির যোগফল outputএ (ফলন) দেখাবে।

এই programটি লেখা একদমই সোজা। এখানে মুলত আমরা দেখতে চাই যে for loopএ (জন্য ঘূর্ণী) initialisation (আদ্যায়ন) অংশের পাশাপাশি update (হালায়ন) অংশও ফাঁকা রাখা যায়। নীচের program (ক্রমলেখ) দেখো। এখান আমরা দুটো variable (চলক) count আর sum নিয়েছি। দুটোরই initial value (আদি মান) শুন্য, কারণ এখন একটা সংখ্যাও input (যোগান) নেয়া হয় নি, আর তাই যোগফলও এই অবস্থায় শুন্য। Variable count কে তুমি চাইলে অবশ্য initialisation অংশেও ঘোষণা করে initial value দিতে পারতে যেমন for (int count = 0; count < 10;) কিন্তু variable sum কে অবশ্যই loopএর (ঘূর্ণী) বাইরে declare করতে হবে, কারণ আমরা output (ফলন) দেখাবো তো loopএর বাইরে।

```
int count = 0, sum = 0;

for( ; count < 10; )   // ফাঁকা হালায়ন
{
  cout << "number? ";
  int number; cin >> number;

  if (number <= 0)     // ধনাত্নক না হলে
    continue;          // পাক ডিঙাও

  sum += number;
  ++count;             // হালায়ন
}

cout << "sum " << sum << endl;
```

উপরের programএ for loopএ (জন্য ঘূর্ণী) আমরা কেবল condition (শর্ত) অংশটি রেখেছি, মোট নম্বর ১০ টি হলো কিনা তা পরীক্ষা করতে। Loopএর (ঘূর্ণী) ভিতরে input prompt (যোগান যাচনা) করে নম্বরটি input (যোগান) নেয়া হয়েছে। তারপর দেখো number variableএর মান যদি শুন্য বা কম হয় তাহলে continue; দিয়ে পাক ডিঙাতে বলা হয়েছে। ধনাত্নক সংখ্যা ছাড়া অন্য রকমের সংখ্যা আসলে আমরা উপেক্ষা করতে চাই, এ কারণে এ ব্যবস্থা।

১৬.৮. For Loop Empty Update (জন্য ঘূর্ণীতে ফাঁকা হালায়ন)

আমরা জানি for loopএ (জন্য ঘূর্ণী) continue; করলে control সরাসরি update (হালায়ন) অংশে চলে যায়। তো আমাদের এই loopএ update অংশতো ফাঁকা রেখেছি, কাজেই count variableএর মান যা ছিলো তাই থাকলো। ফলে আবার input prompt করে নম্বর input নেওয়া হবে। যতক্ষণ শুন্য বা তার কম কোন সংখ্যা input (যোগান) দেয়া হচ্ছে ততক্ষণ এইভাবে চলতে থাকবে count variableএর মান বাড়বে না। এবার ধরো positive নম্বরটি input দেয়া হলো, তাহলে number <= 0 এই শর্তটি মিথ্যা হবে, ফলে control (নিয়ন্ত্রণ) আর পাক ডিঙাবে না, পরের সারিতে গিয়ে নম্বরটিকে sum এর সাথে যোগ করবে, আর count variableএর (চলক) মানও এক বাড়বে। এই ভাবে দশটি ধনাত্নক সংখ্যা হলেই কেবল count variableএর মান বেড়ে দশ হওয়া সম্ভর আর তাতে loop (ঘূর্ণী) থেকে বের হয়ে যাওয়া সম্ভর, negative সংখ্যা বা শুন্য দিয়ে শর্ত মিথ্যা করা সম্ভব হবে না। তাহলে আমরা দেখলাম loopএর (ঘূর্ণী) update অংশকে ফাঁকা রেখে শর্ত সাপেক্ষে update করতে চাইলে সেটা আমরা statement (বিবৃতি) অংশে নিতে পারি।

তবে একটা ব্যাপার এখানে স্মরণ করা দরকার। ধরো দুষ্টামি করে আমরা কখনোই positive সংখ্যা input (যোগান) না দিয়ে কেবলই শুন্য বা negative সংখ্যা input দিতে থাকলাম। এই অবস্থায় কী ঘটবে? তাহলে তো count variableএর (চলক) মান কখনো বাড়বে না, ফলে count < 10 শর্তটি মিথ্যা হওয়ার কোন সম্ভাবনা থাকছে না। এই অবস্থায় কিন্তু for loopটি (জন্য ঘূর্ণী) অসীম সংখ্যক বার ঘুরতে থাকবে। যে loop (ঘূর্ণী) অসীম সংখ্যক বার ঘুরে, আর loop থেকে বের হওয়ার কোন সুযোগ নাই, এ রকম loopকে বলা হয় infinite loop (অসীম ঘূর্ণী)। Infinite loop তৈরী হওয়া মানে এই program কোন দিনই থামবে না। Loop (ঘূর্ণী) লিখলেই আমাদের তাই সতর্ক থাকতে হয় যাতে সেটা কোন ভাবেই inifinite loop না হয়ে যায়।

যদিও এই programএ (ক্রমলেখ) দশটি ধনাত্নক সংখ্যা input (যোগান) দিয়ে দিয়ে loop থেকে বের হওয়ার আমাদের সুযোগ আছে, তবে সেটা কেবল সম্ভব যদি input দাতার (user who is giving the input) সদিচ্ছা থাকে আর আমরা তার ওপরে আস্থা রাখতে পারি। তুমি যদি input দাতার ওপরে আস্থাশীল না হও তাহলে একটা কাজ করতে পারো। সেটা হলো সর্বোচ্চ কত বার তুমি input (যোগান) চাইবে সেটা নির্দিষ্ট করে দিতে পারো। যেমন ১০ টি সংখ্যা input নেয়ার জন্য ধরো আমরা ধরে নিলাম যে সর্বোচ্চ ১৫ বার input দেয়া আমরা মেনে নেবো। নিতান্ত যদি ভুল করে ঋণাত্নক বা শুন্য কেউ দেয়, সেই রকম ভুল আমরা এক্ষেত্রে ৫ বারের বেশী হতে দিবো না। তাহলে আমরা নীচের মতো করে program (ক্রমলেখ) লিখতে পারি।

```
int attempt = 0, count = 0, sum = 0;

for( ; attempt < 15 && count < 10; )
{
  cout << "number? ";
  int number; cin >> mombor;

  ++attempt;

  if (number <= 0)      // ধনাত্নক না হলে
    continue;           // পাক ডিঙাও

  sumber += number;
  ++count;              // হালায়ন
}
```

```
if (count == 10)
  cout << "sum " << sum << endl;
else
  cout << "max attempts finished" << endl;
```

উপরের এই programএ আমরা attemp নামের আরেকটি variable নিয়েছি যেটি দিয়ে সর্বমোট কয়বার input (যোগান) দেয়া হলো সেটা হিসাব রাখবো। প্রতিবার নম্বর input দে-ওয়া মাত্রই attempt variableএর (চলক) মান এক বাড়বে, নম্বরটি ধনাত্নক, ঋণাত্নক, শুন্য যাই হোক, এ কারণে এটি কিন্তু if (number <= 0) continue; এর আগে দেয়া হয়েছে। আর attempt variableএর মান যাতে ১৫ হওয়া পর্যন্ত loop ঘুরে তাই আমরা এবার loopএর (ঘূর্ণী) শর্ত অংশটি বদলে লিখেছি attempt < 15 && number < 10। এর মানে হলো যে কোন একটি শর্ত ভঙ্গ হলেই loop আর ঘুরবে না, কাজেই ১৫ বারের বেশী চেষ্টা করা সম্ভব হবে না, আবার ১০ টির বেশী ধনাত্নক নম্বরও input (যোগান) দেয়া সম্ভব হবে না। তাহলে একটা ব্যা-পার আমরা দেখলাম, loopএর condition (শর্ত) অংশে আমরা চাইলে একাধিক condition boolean connectives (বুলক সংযোজক) যেমন and &&, or ||, not ! দিয়ে সংযুক্ত করে দিতে পারি। সবশেষে দেখো loopএর (ঘূর্ণী) বাইরে আমরা যোগফল output (ফলন) দিয়েছি যদি count variableএর মান ১০ হয়ে থাকে। আর না হয়ে থাকলে মানে ১৫ বারের চেষ্টায়ও ১০ টি ধনাত্নক সংখ্যা নেয়া সম্ভব হয় নাই সেক্ষেত্রে একটা message (বার্তা) দেখানো হয়েছে যে সর্বোচ্চ চেষ্টা শেষ হয়ে গেছে।

১৬.৯ For Loop Empty Statement (জন্য ঘূর্ণীতে ফাঁকা বিবৃতি)

এমন একটি program (ক্রমলেখ) লিখো যেটি শুন্য থেকে প্রতি পাকে ৩ করে বাড়িয়ে সর্বোচ্চ ১০ ধাপ সামনে যাবে, আর এই ভাবে যদি এমন কোন সংখ্যা পায় যেটি ৭ দ্বারা বিভাজ্য তাহলে থেমে যাবে। আমরা এই রূপে পাওয়া সর্বশেষ সংখ্যাটি output জানতে চাই।

```
int number = 0; // count দিয়ে loop
for(int index = 0; count < 10; ++count)
{
  if (number % 7 == 0)  // number দিয়ে break
    break;
  number += 3;
}
cout << number << endl;
```

Break (ক্ষান্তি) ব্যবহার করে এই programটি লেখা বেশ সহজ। Loop (ঘূর্ণী) count variableএর (চলক) মান শুন্য থেকে দশের কম পর্যন্ত চালাও আর প্রতি পাকে numberএর মান ৭ দ্বারা বিভাজ্য হলো কিনা পরীক্ষা করে দেখো। ৭ দ্বারা বিভাজ্য হলে loopএ break (ক্ষান্তি) দাও। আর না হলে numberএর মান ৩ বাড়াও। তুমি কিন্তু চাইলে এই একই কাজ নীচের মতো করেও করতে পারো যেখানে আমরা loop (ঘূর্ণী) চালিয়েছি number variable ব্যবহার করে। লক্ষ্য করো number variableএর মান শুন্য থেকে শুরু করে প্রতিবারে ৩ করে বাড়বে, আর loop চলবে যতক্ষণ number ৭ দ্বারা বিভাজ্য নয় ততক্ষণ, ৭ দ্বারা বিভাজ্য হওয়া মাত্র loop

শেষ হয়ে যাবে। Statement অংশে দেখো আমরা count variableএর মান ১০ বা বেশী হলে loopএ break (ক্ষান্তি) দিয়েছি, আর না হলে count variableএর মান এক বাড়বে।

```
int number, count = 0;  // number দিয়ে loop
for(number = 0;  number % 7 != 0; number += 3)
{
  if (count >= 10)       // count দিয়ে brea
    break;
  ++count;
}
cout << number << endl;
```

তাহলে উপরের দুটি programএ আমরা একবার একটা variableকে (চলক) loopএ (ঘূর্ণী) আর অন্য variableটিকে break (ক্ষান্তি) এ ব্যবহার করেছি, আর আরেকবার ঠিক উল্টোটা করেছি। আমরা কি চাইলে উভয় variableকে loopএ ব্যবহার করতে পারি না। অবশ্যই পারি। নীচের program (ক্রমলেখ) খেয়াল করো। এখনো number ও count দুটো variableকেই loopএ ব্যবহার করছি। Initialisation (আদ্যায়ন) অংশে দুটোর মানই শুরু হয়েছে শুন্য থেকে, আমরা comma (বির্তি) , ব্যবহার করেছি এখানে। Condition (শর্ত) অংশে আমরা দুটো শর্ত দিয়েছি এবং && দিয়ে জোড়া দিয়ে যাতে loop ততক্ষণ চলে যতক্ষণ উভয় শর্ত সত্য হয়। যে কোন একটি শর্ত মিথ্যা হলেই loop শেষ হয়ে যাবে। আর শর্ত দুটি হলো count < 10 ও number % 7 == 0, প্রথম শর্তটি সর্বোচ্চ দশ ধাপের জন্য আর দ্বিতীয় শর্তটি ৭ দ্বারা বিভাজ্য না হওয়া পর্যন্ত। আর loopএর (ঘূর্ণী) update (হালায়ন) অংশে দেখো আমরা count আর number দুটোকেই বাড়িয়েছি, একটাকে এক করে, আরেকটাকে তিন করে। তো এ সবের ফলে আমাদের statement (বিবৃতি) অংশে কিন্তু আর কিছুই করার থাকছে না। আমরা তাই statement অংশে একটা empty statement (ফাঁকা বিবৃতি) দিয়েছি কেবল একটা semicolon (দির্তি) ব্যবহার করে। তুমি চাইলে {} ফাঁকা blockও ব্যবহার করতে পারো।

```
int number, count;
for(count = 0, number = 0;           //
 initialisation
    count < 10 && number % 7 != 0;   // condition
    ++count, number += 3)            // update
 ;                                   // empty
 statement
cout << number << endl;
```

তাহলে আমরা দেখলাম, for loopএর (জন্য ঘূর্ণী) statement (বিবৃতি) ফাঁকা থাকতে পারে। আর initialisation (আদ্যায়ন) ও update (হালায়ন) অংশে একাধিক variable ব্যবহার করা যাবে, উপরোন্ত increment বা decrement যে কেবল এক করে করতে হবে তাও না, বরং অন্য যে কোন পরিমান increment বা decrement করা যাবে। এছাড়া condition (শর্ত) অংশেও দরকার মতো একাধিক শর্ত boolean connectives (বুলক সংযোজক) যেমন এবং && অথবা || দিয়ে জোড়া দেওয়া যাবে।

১৬.১০ Statement and Update (বিবৃতি হালায়ন মিথস্ক্রিয়া)

সাধারণত for loopএর (জন্য ঘূর্ণী) variableটির (চলক) মান আমরা হয় statementএ (বিবৃতি) অথবা updateএ (হালায়ন) পরিবর্তন করি। কিন্তু উভয় অংশে পরিবর্তন করলে কী হবে?

```
for(int index = 0; index < 10; ++index)
{
  cout << index << " ";
  if (index == 5)
    ++index;
}
cout << endl;
```

উপরের programএ (ক্রমলেখ) আমরা (for loop)এর (জন্য ঘূর্ণী) variable (চলক) index এর মান statement (বিবৃতি) ও update (হালায়ন) উভয়খানে পরিবর্তন করেছি। Updateএ নিয়মিত পরিরর্তন হিসাবে এক করে বাড়বে প্রতিবার, আর statementএ (বিবৃতি) যদি current value (চলতি মান) হয় তাহলে এক বাড়বে। এর ফলে কী হবে? খেয়াল করো index এর মান ৫ ছাড়া অন্য কিছু হলে কেবল update অংশে এক বাড়বে, কিন্তু index এর মান ৫ হলে, তখন statementএ if (index == 5) শর্ত সত্য হওয়ায় সেখানে index এর মান এক বাড়বে, আবার updateএ তো আরো এক বাড়বেই। ফলে পরের পাকে যখন index চলকের মান outputএ (ফলন) আসবে তখন সেটা ৬ না হয়ে ৭ হবে। কাজেই নীচে যেমন দেখানো হলো আমরা outputএ ৬ দেখতে পাবো না।

```
0 1 2 3 4 5 7 8 9
```

For loopএ (জন্য ঘূর্ণী) update (হালায়ন) অংশে incrementটা (বৃদ্ধি) থাকুক, কিন্তু statement (বিবৃতি) অংশে আমরা increment না করে যদি decrement (হ্রাস) করি, তাহলে অবস্থা কী দাঁড়াবে? ধরা যাক নীচের programএর (ক্রমলেখ) মতো statementএ index variableএর মান ৫ হলে এক না বাড়িয়ে আমরা বরং এক কমালাম। খেয়াল করো এই ক্ষেত্রে index variableএর মান ৫ হওয়ার পরে যদির শর্ত সত্য হওয়ায় এক কমে ৪ হবে, কিন্তু updateএ গিয়ে আবার এক বেড়ে হয়ে যাবে ৫, outputএ (ফলন) ৫ দেখানোর পরে আবার যদির শর্ত সত্য হওয়ায় এক কমে হবে ৪, updateএ গিয়ে হবে ৫, ফলে আবার outputএ ৫ দেখাবে। কাজেই 0 1 2 3 4 5 করে একবার ৫ দেখানোর পরে এই for loop (জন্য ঘূর্ণী) তারপরের প্রতি পাকেই কেবল ৫ই দেখিয়ে যাবে। আর এই loopএ (ঘূর্ণী) বের হওয়ার কোন উপায় নাই। কাজেই এটি একটি infinite loopএ (অসীম ঘূর্ণী) পরিণত হবে। তুমি জানো আমরা সবসময় চাই infinite loop এড়িয়ে যেতে।

```
for(int index = 0; index < 10; ++index)
{
  cout << index << " ";
  if (index == 5)
    --index;
}
cout << endl;
```

১৬.১১ Unnecessary For Loop (অদরকারী জন্য ঘূর্ণী)

এমন একটি program রচনা করো যেটি ০ থেকে ৯ পর্যন্ত প্রতিটি অঙ্ককে কথায় লিখবে। আর অঙ্কগুলোকে পরপর অঙ্কেই লিখলে যে program হতো তার সাথে তফাৎটা কেমন দাঁড়ায়?

```
0 1 2 3 4 5 6 7 8 9
```

অঙ্কগুলোকে উপরের মতো করে পরপর অঙ্কেই লিখলে ব্যাপারটা তো খুবই সহজ। এই রকম program (ক্রমলেখ) আমরা নীচে দেখালাম। একটা for loop (জন্য ঘূর্ণী) variable (চলক) digitএর মান ০ থেকে ৯ পর্যন্ত এক করে বাড়িয়ে যাবে, আর প্রতি পাকে আমরা cout ব্যবহার করে digitটাকে দেখাবো। এখানে cout জানে কোন অঙ্ককে কীভাবে লিখতে হয়!

```
for(int digit = 0; digit <= 9; ++digit)
  cout << digit << " ";
cout << endl;
```

এবার অঙ্কগুলোকে যদি অঙ্কে না লিখে আমরা কথায় লিখি, তাহলে কী করবো? এখানে যেহেতু আমরা ০ থেকে ৯ পর্যন্ত অনেকবার কথায় লিখছি, অনেকে তাই এখানে for loop (জন্য ঘূর্ণী) ব্যবহার করতে উদ্বুদ্ধ হয়। ফলে তারা নীচের মতো করে program তৈরী করে। এখানে cout কিন্তু কোন অঙ্ককে কী ভাবে লিখতে হবে তা জানেনা, আমাদের তাই একটি switch case (পল্টি ব্যাপার) ব্যবহার করে প্রতিটি অঙ্ককে আলাদা আলাদা করে বলে দিতে হয়েছে।

```
for(int digit = 0; digit <= 9; ++digit)
{
  swtich(digit)
  {
    case 0: cout << "shunyo "; break;
    case 1: cout << "ek "; break;
    case 2: cout << "dui "; break;
    case 3: cout << "tin "; break;
    case 4: cout << "char "; break;
    case 5: cout << "panch "; break;
    case 6: cout << "soy "; break;
    case 7: cout << "shat "; break;
    case 8: cout << "aat "; break;
    case 9: cout << "noy "; break;
  }
}
cout << endl;
```

কিন্তু উপরের programএ (ক্রমলেখ) for loop (জন্য ঘূর্ণী) ব্যবহার আসলে অদরকারী। কারণ loopএর (ঘূর্ণী) প্রতি পাকে আসলে switch caseএর (পল্টি ব্যাপার) আলাদা আলাদা ব্যাপার execute (নির্বাহিত) হচ্ছে। আমাদেরকে তো প্রতিটি পাকের জন্যে সেই আলাদা আলাদা কাজই করতে হলো, তাহলে আর loop কেন ব্যবহার করবো, কাজগুলো নীচের মতো করে সরাসরি একের পর এক লিখলেই তো হয়ে যায়। মনে রাখবে আমরা loop তখনই ব্যবহার করবো যখন প্রতি পাকের জন্য এই রকম আলাদা আলাদা কিছু করতে হয় না। যেমন কথায় না দেখিয়ে অঙ্কে

দেখালে কিন্তু আমাদের for loop ব্যবহার করলেই চলতো কারণ cout ব্যবহারের কারণে প্রতিটি অঙ্ক দেখানোতে যে ভিন্নতা সেটা আমাদের সরাসরি সামলাতে হয় না।

```
cout << "shunyo ";
cout << "ek ";
cout << "dui ";
cout << "tin ";
cout << "char ";
cout << "panch ";
cout << "soy ";
cout << "shat ";
cout << "aat ";
cout << "noy ";
cout << endl;
```

১৬.১২ General Purpose For Loop (জন্য ঘূর্ণীর সাধারণ ব্যবহার)

For loop (জন্য ঘূর্ণী) আমরা এ পর্যন্ত কেবল order (ক্রম), progression (প্রগমণ), series (ধারা) ইত্যাদির ক্ষেত্রে ব্যবহার করেছি। For loop (জন্য ঘূর্ণী) কী এ সব ছাড়া যে কোন শর্ত পরীক্ষার মাধ্যমে সাধারণ ভাবে একটি loop তৈরীতে ব্যবহার করা যায়?

সিপিপি ছাড়া অন্যান্য ভাষায় for loop (জন্য ঘূর্ণী) কেবল order (ক্রম), progression (প্রগমণ), series (ধারা) এসবেই ব্যবহার করা যায়। তবে সিপিপির for loop আসলে অনেক শক্তিশালী, এটাকে যে কোন রকম loop তৈরীতে ব্যবহার করা যায়। বস্তুত সিপিপিতে for loop (জন্য ঘূর্ণী) দিয়েই সকল রকমের loop তৈরী করা যায়, কাজেই অন্য কোন loop দরকার হয় না, যদিও সিপিপিতে আরো দুটি loop (ঘূর্ণী) আছে, যে গুলো আমরা পরের পাঠগুলোতে দেখবো।

```
int number = 1, count = 0, sum = 0;

for( ; number != 0; )    // আদ্যায়ন হালায়ন ফাঁকা
{
    cin >> number;       // নম্বর যোগান নাও
    if (number)          // নম্বর শুন্য না হলে
    {
      count += 1;        // গুনতি এক বাড়বে
      sum += number;     // যোগফলেও যোগ হবে
    }
}

cout << "count " << count << " ";
cout << "sum " << sum << endl;
```

উপরের programএ (ক্রমলেখ) খেয়াল করো। এখানে আমরা কিছু নম্বর input (যোগান) নিয়ে তাদের যোগফল বের করতে চাই। তবে কয়টি নম্বর input নিবো আমরা সেটা আগে থেকে জানিনা । ব্যবহারকারী যতগুলো ইচ্ছে নম্বর input দিতে থাকবে, যখন সে আর কোন নম্বর input দিতে চায় না তখন সে একটা শুন্য input দিয়ে সেটা জানাবে। আমরা তারপর কয়টি নম্বর input নিয়েছি আর তাদের যোগফল কত সেটা outputএ (ফলন) দেখাবো।

তো এই programটি (ক্রমলেখ) খুবই সহজ। আমরা একটি for loop (জন্য ঘূর্ণী) নিয়েছি, কিন্তু এটাকে আমরা কোন variableএর (চলক) মান বাড়িয়ে বা কমিয়ে ঘুরাবো না। আমরা মুলত শর্তটা ব্যবহার করবো loop (ঘূর্ণী) তৈরীতে। তিনটি variable (চলক) number, count আর sum নেয়া হয়েছে যাদের আদিমান (initial value) দেওয়া হয়েছে ঘূর্ণীর বাইরে। ঘূর্ণীর পরে যেহেতু count ও sum ফলনে (output) দেখানো হবে, তাই ওগুলো অবশ্যই ঘূর্ণীর (loop) বাইরে ঘোষণা (declare) করতে হবে। কিন্তু number চলকটি ঘূর্ণীর ভিতরে ঘোষণা ও আদ্যায়ন (initialisation) করা যেতে পারতো। যাইহোক ঘূর্ণীর ভিতরে নম্বরটি যোগান নিয়ে যদি শুন্য না হয় তাহলে গুনতি এক বাড়িয়ে যোগফলের সাথে নম্বরটি যোগ করা হয়েছে।

একটা বিষয় খেয়াল করো আমাদের ঘূর্ণীতে (loop) শর্ত number != 0 অর্থাৎ number এর মান শুন্য ছাড়া অন্য কিছু হলে কেবল ঘূর্ণীর বিবৃতি (statement) নির্বাহিত (execute) হবে। তো প্রথমবার আমরা তো অবশ্যই ঘূর্ণীর ভিতরে ঢুকতে চাই, কিন্তু number তো তখন পর্যন্ত একটাও যোগান (input) নেয়া হয় নাই। ঘূর্ণীর ভিতরে যদি আমাদের ঢুকতেই হয়, আমাদের সেক্ষেত্রে কোন ভাবে শর্ত সত্য করে দিতে হবে, number চলকরে আদিমান শুন্য ছাড়া একটা কিছু দিয়ে রাখতে হবে। আমরা number এর আদি মান দিয়েছি 1, তুমি চাইলে শুন্য ছাড়া অন্য যে কোন কিছু দিতে পারতে। চলক count আর sum এর আদিমান তো শুন্যই দিতে হবে, সেটা বুঝতেই পারছো, যেহেতু তখনও আমাদের একটাও নম্বর যোগান নেওয়া হয় নাই।

আসলে জন্য ঘূর্ণীতে (for loop) শর্ত পরীক্ষণ হয় সাধারণত বিবৃতিতে (statement) ঢুকার আগে, অথচ আমরা এখানে শর্ত পরীক্ষা করতে চাই বিবৃতি অংশের পরে, কারণ বিবৃতিতে আমরা যে নম্বরটি যোগান নিবো সেটা আমরা পরীক্ষা করতে চাই পরের পাকে ঢুকার আগে। পরের পাকের আগে পরীক্ষণ মানে আগের পাকের পরে আর কী! আর সে কারণে জোর করে number চলকের আদি মান 1 দিয়ে প্রথমবার শর্ত সত্য বানিয়ে আমরা খানিকটা চালাকি করেছি!

১৬.১৩ For Loop Variations (জন্য ঘূর্ণীর নানান বাহার)

For loop (জন্য ঘূর্ণী) কত ভাবে লেখা যায়? এর মধ্যে নিশ্চয় বুঝে ফেলেছো for loopএর for (; ;) bracketsএর (বন্ধনী) ভিতরের semicolon (দির্তি) ; দুটো কেবল আবশ্যিক। তাহলে initialisation (আদ্যায়ন), condition (শর্ত), update (হালায়ন) কত ভাবে বিন্যাস করা সম্ভব?

```
for (আদ্যায়ন; শর্ত; হালায়ন)          আদ্যায়ন
  বিবৃতি                                 for ( ; শর্ত; হালায়ন)
                                           বিবৃতি
```

উপরে বামপাশে for loopএর (জন্য ঘূর্ণী) সাধারণ অবস্থা দেখানো হয়েছে, আর ডান পাশে দেখানো হয়েছে যে চাইলেই initialisation (আদ্যায়ন) অংশটি loopএর (ঘূর্ণী) বাইরে নিয়ে যাওয়া যায়। তাতে ফলাফল একই থাকবে। তবে তফাৎ অল্প একটুই আছে সেটা হলো বামপাশের initialisationএ যদি variable declare করা সেটি কেবল loopএর জন্য local variable

(স্থানীয় চলক), তাই loopএর পরে আর ব্যবহার করা যায় না। কিন্তু ডানপাশের initialisationএ যদি variable declare (চলক ঘোষণা) করা হয় সেটা loopএর (ঘূর্ণী) পরেও ব্যবহার করা যাবে। আমরা এর পরে থেকে initialisation (আদ্যায়ন) loopএর (ঘূর্ণী) ভিতরেই হয়তো লিখবো, তবে আরেকবার বলেই দিচ্ছি variable declare করার ব্যাপারটিতে কোন সমস্যা না থাকলে তুমি চাইলেই সেটি loopএর আগেই লিখতে পারবে। সুতরাং initialisation কোথায় থাকলো সে বিষয়ে ভিন্নতা এখানের পরে আর দেখাবো না।

```
for (আদ্যায়ন; শর্ত; হালায়ন)
  বিবৃতি
```

```
for (আদ্যায়ন; শর্ত; )
  {
    বিবৃতি
    হালায়ন
  }
```

এরপর উপরে দেখো আমরা কী ভাবে update (হালায়ন) অংশটকু যথাস্থানে না লিখে তার বদলে statementএর (বিবৃতি) সাথে দিয়ে দিয়েছি। এতে loopএর (ঘূর্ণী) ফলাফল একই থাকবে, কারণ বিবৃতির পরপরই তো update execute (নির্বাহিত) হতো, এখনো তাই হচ্ছে। কাজেই তুমি চাইলে যে কোন সময় এইটা করতে পারো। আমরা এরপরে update (হালায়ন) যথাস্থানেই রাখবো, এ সংক্রান্ত variationগুলো (ভেদন) আর দেখাবো না।

```
for (আদ্যায়ন; ; হালায়ন)
{
  if (!শর্ত) break;
  বিবৃতি
}
```

```
for (আদ্যায়ন; true; হালায়ন)
{
  if (!শর্ত) break;
  বিবৃতি
}
```

এবার আমরা condition (শর্ত) অংশের ভিন্নতা দেখবো। উপরে দেখো আমরা শর্তটিকে statement (বিবৃতি) অংশে নিয়ে গিয়েছি। ফলে condition অংশে বাম পাশের মতো হয় ফাঁকা রাখা হবে, না হয় ডানপাশের মতো true লিখে দেওয়া হবে। দুটো মুলত একই কথা কারণ ফাঁকা শর্ত মানে সত্য। Condition যখন statement অংশে গেছে তখন দেখো আমরা ! লাগিয়ে উল্টো শর্ত দিয়ে loopএ (ঘূর্ণী) break (ক্ষান্তি) দিয়েছি যাতে শর্ত মিথ্যা হলেই loop থেকে control (নিয়ন্ত্রণ) বের হয়ে যায়। সুতরাং এই variationগুলো (ভেদন) for loopএর (জন্য ঘূর্ণী) সাধারণ অবস্থার সমার্থক।

```
for (আদ্যায়ন; শর্ত১; হালায়ন)
{
  if (শর্ত২)
    বিবৃতি
}

for (আদ্যায়ন; ; হালায়ন)
{
  if (!শর্ত১) break;
  if (শর্ত২) বিবৃতি
}
```

```
for (আদ্যায়ন; শর্ত১; হালায়ন)
  {
    if (!শর্ত২) continue;
    বিবৃতি
  }

for (আদ্যায়ন; ; হালায়ন)
{
  if (!শর্ত১) break;
    if (!শর্ত২) continue;
  বিবৃতি
}
```

উপরে দেখো statement অংশে আমরা আরেকটি শর্ত ব্যবহার করে statement (বিবৃতি) execute করেছি। তো আমরা চাইলে এখানে উপরের ডান পাশের মতো করে শর্ত২ এর বিপরীত শর্ত দিয়ে পাক continue (ডিঙাতে) পারি। তাতে statement আর if (যদি) এর অধীনে থাকছে না। উদাহরণগুলোর শর্ত১ কে যদি আমরা statement অংশে নামিয়ে দেই তাহলে আমরা যা পাবো তাও উপরে দেখানো হয়েছে।

```
for (আদ্যায়ন; ; )
{
  if (!শর্ত১) break;
  if (শর্ত২) বিবৃতি
  হালায়ন
}
```

```
for (আদ্যায়ন; ; )
{
 if (!শর্ত১) break;
   if (!শর্ত২)
   {
     হালায়ন
     continue;
   }
   বিবৃতি
   হালায়ন
}
```

সবশেষে আমরা উপরে একটু দেখি update (হালায়ন) যদি statement (বিবৃতি) অংশে ঢুকে যায় তাহলে break (ক্ষান্তি) আর continue (ডিঙানো) এর সাথে কী মিথস্ক্রিয়া ঘটে। বিশেষ করে ডানপাশে দেখো শর্ত২ সত্য না হলে আমাদের প্রথমে updateএর (হালায়ন) কাজটুকু করতে হবে, তারপর আমরা continue করতে (ডিঙাতে) পারবো। এর কারণ for(আদ্যায়ন;;) এখানে যেহেতু update অংশটুকু ফাঁকা আর continue করলে স্বাভাবিক ভাবে for loopএ (জন্য ঘূর্ণী) update অংশটুকুতেই control (নিয়ন্ত্রণ) চলে যায়, সেহেতু updateএর কাজটুকু আমাদের continue; এর আগেই সেরে ফেলতে হবে। এ ছাড়া খেয়াল করো updateএর কাজটুকু আমরা statementএর পরেও করেছি, সেটাতো স্বাভাবিক ভাবেই হওয়ার কথা।

১৬.১৪ Precondition in While Loop (পূর্ব শর্তের ক্ষণ ঘূর্ণী)

সিপিপিতে একটি program (ক্রমলেখ) লিখো যেটি userএর (ব্যবহারকারী) কাছে থেকে দুটি positive integer (ধনাত্মক পূর্ণক) input (যোগান) নিয়ে তাদের গরিষ্ঠ সাধারণ গুণনীয়ক বা গসাগু নির্ণয় করবে। এই programটি তুমি while loop (ক্ষণ ঘূর্ণী) ব্যবহার করে লিখবে।

নীচে দেখানো program খেয়াল করো। আমরা প্রথমে variable declare (চলক ঘোষণা) করে input prompt (যোগান যাচনা) করে integer (পূর্ণক) দুটি input (যোগান) নিয়েছি। এরপর আমরা if (যদি) লাগিয়ে শর্ত পরীক্ষা করেছি, দেখেছি integerদুটির যে কোনটি শুন্য বা তার কম কিনা। কারণ শুন্য বা negative সংখ্যার জন্য আমরা গসাগু নির্ণয় করবো না, সেক্ষেত্রে বরং আমরা error message (ত্রুটি বার্তা) দেখিয়ে program ব্যর্থতার (failure) সাথে শেষ করবো। এরপরে রয়েছে loop (ঘূর্ণী) দিয়ে আমাদের গসাগু নির্ণয়ের মুল অংশটুকু।

ফিরিস্তি ১৬.৭: HCF of Two Numbers (দুটি সংখ্যার গসাগু)

```
int integer1, integer2;    // চলক ঘোষণা
cout << "two positive integers: ";  // যোগান যাচনা
cin >> integer1 >> integer2;     // যোগান নেয়া
```

```
if (integer1 <= 0 || integer2 <= 0) // ধনাত্মক না হলে
{
  cout << "zero or negative integer!" << endl;
  return EXIT_FAILURE;
}

// গসাগু নির্ণয়ের মুল অংশ
int remainder = integer1 % integer2;  // ভাগশেষ নির্ণয়
while(remainder)  // ভাগশেষ শুন্য না হলে
{
  integer1 = integer2;     // ভাজকই হবে নতুন ভাজ্য
  integer2 = remainder;    // ভাগশেষ হবে নতুন ভাজক
  remainder = integer1 % integer2;  // আবার ভাগশেষ
}

cout << "HCF: " << integer2 << endl;  // ভাজকই গসাগু

return EXIT_SUCCESS;
```

গসাগু নির্ণয় করতে গেলে আমাদের প্রথমে প্রদত্ত সংখ্যা দুটির একটিকে দিয়ে আরেকটিকে ভাগকরে ভাগশেষ বের করতে হয়। ভাগশেষ শুন্য হওয়া মানে আমাদের আর ভাগ করতে হবে না, আর সেক্ষেত্রে ভাজক যে পূর্ণকটি সেটিই হলো আমাদের গসাগু। তো খেয়াল করো আমরা কিন্তু integer1 কে integer2 দিয়ে ভাগ করে ভাগশেষ নিয়েছি remainder variableএ। এক্ষেত্রে আমরা সিপিপির % operator ব্যবহার করে ভাগশেষ নির্ণয় করেছি। আমরা এখানে একটি নতুন ধরণের loop (ঘূর্ণী) ব্যবহার করেছি যেটি হলো while loop (ক্ষণ ঘূর্ণী)। এই while loopএর শর্ত দেয়া হয়েছে remainder অর্থাৎ ভাগশেষ যতক্ষণ সত্য (বা শুন্য নয়) ততক্ষণ loop (ঘূর্ণী) চলবে। তুমি কিন্তু চাইলে শর্ত হিসাবে (remainder) না লিখে (remainder != 0) লিখতে পারতে। যাইহোক শর্তটি সত্য না হলে অর্থাৎ remainder শুন্য হলে loopএর বাইরে blockএর (মহল্লা) {} পরে দেখো আমরা integer2কে গসাগু হিসাবে output দেখিয়েছি।

এখন কথা হচ্ছে ভাগশেষ শুন্য না হলে আমাদের কী করতে হবে? সেটা আমরা blockএর (মহল্লা) ভিতরে লিখেছি। আমাদের আগের ভাজকটি হবে নতুন ভাজ্য, তাই আমরা প্রথমে লিখেছি integer1 = integer2;, এতে কিন্তু আগের ভাজ্যটি হারিয়ে গেলো, আমাদের আসলে সেটি আর দরকার নাই। তারপর দেখো integer2 = remainder; লিখে আমরা ভাগশেষটিকে নতুন ভাজক হিসাবে নিয়ে নিলাম। তুমি নিশ্চয় বুঝতে পারছো কেন integer1 = integer2; আগে আর integer2 = remainder; পরে লিখতে হয়েছে। যদি উল্টোটা করা হতো তাহলে কিন্তু integer2 = remainder; এর কারণে integer2 যেটি কিনা আমাদের নতুন ভাজ্য হবে সেটির মান হারিয়ে যেতো, ফলে ঠিক পরপরই integer1 = integer2; করলে আমরা যে ভাজ্যটি পেতাম সেটা আসলে remainderএরই মান। নতুন ভাজ্য ও নতুন ভাজক ঠিক করার পরে দেখো এবার আমরা আবার ভাগশেষ নির্ণয় করেছি remainder = integer1 % integer2 লিখে। এই ভাগশেষটি শুন্য হলে আমাদের loop (ঘূর্ণী) থেকে বের হয়ে যেতে হবে, আর শুন্য না হলে আবারও loopএর (ঘূর্ণী) blockএর (মহল্লা) ভিতরে যা আছে তা করতে হবে।

তুমি হয়তো এবার প্রশ্ন করতে পারো, আচ্ছা আমরা কি এই program for loop (জন্য ঘূর্ণী)

দিয়ে লিখতে পারতাম। কেন নয়? নীচে দেখো আমরা একই program for loop (জন্য ঘূর্ণী) ব্যবহার করে লিখেছি, মুলত while loopএর (ক্ষণ ঘূর্ণী) বদলে for loopটা (জন্য ঘূর্ণী) এখানে দেখানো হয়েছে। Initialisation (আদ্যায়ন) অংশে আছে প্রথমবার ভাগশেষ নির্ণয়ের ব্যাপারটা, Condition (শর্ত) অংশে আছে ভাগশেষ শুন্য না হওয়ার শর্ত যেটা কিনা এখানে remainder != 0 লিখা হয়েছে কিন্তু কেবল remainder লিখলেও চলতো। আর update (হালায়ন) অংশে আবার ভাগশেষ নির্ণয়ের অংশটুকু দিয়েছি। For loopএর (জন্য ঘূর্ণী) নানান বাহার আমরা যে গুলো দেখেছিলাম তুমি সেগুলো এখানেও নিজে নিজে প্রয়োগ করতে পারো।

```
for(int remainder = integer1 % integer2;   // আদ্যায়ন
    remainder != 0;                        // শর্ত
    remainder = integer1 % integer2)       // হালায়ন
{
  integer1 = integer2;     // ভাজকই হবে নতুন ভাজ্য
  integer2 = remainder;    // ভাগশেষ হবে নতুন ভাজক
}
```

এবার তাহলে প্রশ্ন করতে পারো, for loop (জন্য ঘূর্ণী) দিয়েই যদি এতো সুন্দর কাজ হয় তাহলে while loopএর (ক্ষণ ঘূর্ণী) দরকার কী? সত্যি বলতে আসলে দরকার নাই। সিপিপিতে for loop (জন্য ঘূর্ণী) এতটাই শক্তিশালী যে আর কোন loop দরকার নাই। তবে for loop-টা তবুও বেশীর ভাগ ক্ষেত্রে ক্রম, প্রগমন, ধারা ইত্যাদির ক্ষেত্রে বেশী ব্যবহার করা হয়, আর while loop ব্যবহার করা সাধারণ ক্ষেত্রে। While loopএ (ক্ষণ ঘূর্ণী) loopএর (ঘূর্ণী) অংশ হিসাবে initialisation (আদ্যায়ন) নাই, তাই সেরকম কিছু দরকার হলে ওই অংশটুকুকে রা-খতে হবে loopএরও আগে। While loopএ (ক্ষণ ঘূর্ণী) update (হালায়ন) অংশও আলাদা করে নাই, কাজেই সেটি চলে যাবে statementএর অংশ হিসাবে। এর মানে তোমাকে অবশ্যই statementএর ভিতরে updateএর কাজ করে দিতে হবে যাতে loopটা infinite loopএ পরিণত না হয়। একটা বিষয় খেয়াল করো শর্ত প্রথমেই মিথ্যা হলে while loop (ক্ষণ ঘূর্ণী) এক-বারও না ঘুরতে পারে, এটা অবশ্য for loopএর (জন্য ঘূর্ণী) জন্যেও সত্য। এই উভয় loop প্রথমে শর্ত পরীক্ষা করে, শর্ত সত্য হলে তারপর ঘুরতে যায়। তোমাকে যদি নুন্যতম একবার কাজ করতেই হয় সেটা তাহলে loop আগে বা পরে করে ফেলতে হবে। গসাগু নির্ণয়ের ক্ষেত্রে আমাদের যেমন কমপক্ষে একবার ভাগশেষ করতেই হবে, যেটি আমরা while loopএ (ক্ষণ ঘূর্ণী) loopএর আগেই আর for loopএ (জন্য ঘূর্ণী) initialisationএ (আদ্যায়ন) করে ফেলেছি।

```
for(initialisation;                initialisation
    condition; update)             while(condition)
  statement                          statement+update
```

১৬.১৫ Post-condition in Do Loops (উত্তর শর্তের করো ঘূর্ণী)

দুটো positive integerএর গসাগু নির্ণয়ের programটি (ক্রমলেখ) তুমি আরেকবার লিখো, কিন্তু এবার তুমি while loop (ক্ষণ ঘূর্ণী) বা for loop (জন্য ঘূর্ণী) ব্যবহার না করে তার বদলে ব্যবহার করবে do loop (করো ঘূর্ণী)। Do loop হলো সিপিপিতে তৃতীয় ও শেষ প্রকারের loop।

আমরা আগের পাঠেই এই programটি আলোচনা করেছি ক্ষণ while loop (ঘূর্ণী) ব্যব-হার করে, এখানে তাই আগে সেটিই আরেকবার একটু দেখে নেই। দুটো variable integer1,

integer2 তুমি ঘোষণা করে সেগুলোতে positive integer input (যোগান) নিবে। দরকার হলে আগের পাঠ থেকে input (যোগান) নেয়ার ও তারপর integer দুটি positive কিনা পরীক্ষা করার ব্যাপারটি দেখে নিতে পারো। তাহলে আমরা এবার গসাগু নির্ণয়ের মুল অংশটায় যেতে পারি।

```
int integer1, integer2; // ধনাত্মক মান তুমি input নিবে
int remainder = integer1 % integer2;  // ভাগশেষ নির্ণয়
while(remainder)  // ভাগশেষ শুন্য না হলে
{
  integer1 = integer2;     // ভাজকই হবে নতুন ভাজ্য
  integer2 = remainder;    // ভাগশেষ হবে নতুন ভাজক
  remainder = integer1 % integer2;  // আবার ভাগশেষ
}

cout << "HCF: " << integer2 << endl;  // ভাজকই গসাগু
```

উপরের এই programটির (ক্রমলেখ) loopটাকে (ঘূর্ণী) যদি আমরা বিস্তার করি, মানে loop না লিখে প্রত্যেক পাকে যা হতো সেগুলো যদি বার বার লিখি তাহলে কেমন হতো সেটা আমরা নীচে দেখালাম। খেয়াল করো প্রথম ভাগশেষ নির্ণয়টা কিন্তু উপরের loopএর বাইরে ছিলো, আর তারপর নতুন ভাজ্য, নতুন ভাজক, আর আবার ভাগশেষ নির্ণয়ের statementগুলো (বিবৃতি) ছিলো loopএর ভিতরে, তাই ওগুলো নীচের বিস্তারণে বারবার এসেছে।

```
remainder = integer1 % integer2;  // ভাগশেষ নির্ণয়
integer1 = integer2;     // ভাজকই হবে নতুন ভাজ্য
integer2 = remainder;    // ভাগশেষ হবে নতুন ভাজক
remainder = integer1 % integer2;  // আবার ভাগশেষ
integer1 = integer2;     // ভাজকই হবে নতুন ভাজ্য
integer2 = remainder;    // ভাগশেষ হবে নতুন ভাজক
remainder = integer1 % integer2;  // আবার ভাগশেষ
integer1 = integer2;     // ভাজকই হবে নতুন ভাজ্য
integer2 = remainder;    // ভাগশেষ হবে নতুন ভাজক
remainder = integer1 % integer2;  // আবার ভাগশেষ
.....................................................
```

এখন উপরের এই বিস্তারণ দেখে কারো মনে কিন্তু অন্য রকম করে loop লেখার সাধ জাগতে পারে। কেউ হয়তো বলতে পারে loopটা (ঘূর্ণী) কেন প্রথম ভাগশেষ নির্ণয়কে বাদ দিয়ে শুরু হয়েছে। Loopটাতো বরং প্রথম ভাগশেষ নির্ণয় থেকেই শুরু হতে পারতো। কথা সত্য, আর তাইতো আমাদের নতুন ধরনের একটি loopএর (ঘূর্ণী) উদ্ভব হয়েছে, যেটি হলো do loop (করো ঘূর্ণী)।

নীচের programএ (ক্রমলেখ) দেখো আমরা গসাগু নির্ণয় করেছি do loop (করো ঘূর্ণী) ব্যবহার করে। এখানে প্রথমবার ভাগশেষ নির্ণয় করা হয়েছে loopএর (ঘূর্ণী) ভিতরেই। আর তারপর নতুন ভাজ্য ও নতুন ভাজক নির্ধারণ করা হয়েছে। Loopএর শর্ত পরীক্ষণ তারও পরে while (remainder); যেখানে লেখা হয়েছে সেখানে। শর্ত যদি সত্য হয় তাহলে loopএর পরের পাক শুরু হবে, অর্থাৎ control (নিয়ন্ত্রণ) লাফ দিয়ে doএর পরে যে block (মহল্লা) {} শুরু হয়েছে সেখানে চলে যাবে। তবে একটা গুরুত্বপূর্ণ বিষয় এখানে উল্লেখ করতে হবে এখানে সেটা হলো outputএ (ফলন) কিন্তু এখন গসাগু integer2 হবে না, বরং গসাগু হবে integer1। এর কারণ

হলো blockএর ভিতরে remainder variableএর মান শুন্য হোক বা না হোক আমরা কিন্তু ভাজকটাকে নতুন ভাজ্য হিসাবে ধরে নিয়েছি, ফলে integer2 এর মান এখন integer1এ আছে।

```
int integer1, integer2; // ধনাত্নক মান তুমি যোগান নিবে
int remainder;  // এই variable বাইরেই ঘোষণা করতে হবে!
do
{
  remainder = integer1 % integer2;  // ভাগশেষ নির্ণয়
  integer1 = integer2;     // ভাজকই হবে নতুন ভাজ্য
  integer2 = remainder;    // ভাগশেষ হবে নতুন ভাজক
}
while (remainder);

// সতর্কতা: এখানে গসাগু কিন্তু integer1, integer2 নয়
cout << "HCF: " << integer1 << endl;  // ভাজকই গসাগু
```

তাহলে while loop (ক্ষণ ঘূর্ণী) আর (do loop) করো ঘূর্ণীর তফাৎ হলো আগেরটিতে শর্ত পরীক্ষা পাকে ঢুকার আগে হয়, শর্ত সত্য হলে পাকে ঢুকে, আর পরেরটিতে শর্ত পরীক্ষা পাক শেষ করে হয়, শর্ত সত্য হলে পরের পাকে ঢুকে। এর মানে do loopএর (করো ঘূর্ণী) প্রথম পাক বাদ দিলে ওইটা while loop (ক্ষণ ঘূর্ণী) হয়ে যেতে পারে অথবা উল্টোটা।

১৬.১৬ Break and Continue Again (আবার ক্ষান্তি ও ডিঙানো)

এমন একটি program (ক্রমলেখ) তৈরী করো যেটি ব্যবহারকারীকে serial অনুযায়ী একটি menu (প্রাপণ্য) দেখাবে যোগ, বিয়োগ, গুণ, ভাগফল, বা ভাগশেষ কলন (calculate) করার জন্য। ব্যবহারকারী যত নম্বরের কলন করতে চাইবে তার জন্য দুটি integer (পূর্ণক) inupt (যোগান) নিয়ে হিসাব করে output (ফলন) দেখাবে। ব্যবহারকারী যতক্ষণ একের পর এক কলন করে যেতে চায় তুমি ততক্ষণ menu দেখিয়ে, input নিয়ে, কলন করে যাবে।

ফিরিস্তি ১৬.৮: Rudimentary Toy Calculator (অনুন্নত খেলনা কলনি)

```
// একটা infinite loop তৈরী করবো
while(true) // অথবা for (; ;)
{
  // menu দেখাও
  cout << "toy calculator" << endl;
  cout << "0. exit" << endl;
  cout << "1. plus + " << endl;
  cout << "2. minus –" << endl;
  cout << "3. times *" << endl;
  cout << "4. quotient /" << endl;
  cout << "5. remainder %" << endl;

  int choice;     // পছন্দ variable
```

```
    cin >> choice;  // input

    if (choice == 0)  // পছন্দ শুন্য হলে
      break;          // break from loop

    // উল্টাপাল্টা পছন্দ হলে continue
    if (choice < 1 || choice > 5)
      continue;

    // integer দুটি prompt দিয়ে input নাও
    cout << "two integers: ";
    int integer1, integer2, result;
    cin >> integer1 >> integer2;

    // পছন্দ অনুযায়ী ফলাফল কলন করো
    switch(choice)
    {
      case 1: result = integer1 + integer2; break;
      case 2: result = integer1 - integer2; break;
      case 3: result = integer1 * integer2; break;
      case 4: result = integer1 / integer2; break;
      case 5: result = integer1 % integer2; break;
    }

    // output দেখাও
    cout << "result: " << result << endl;
  }

  // বিদায় সম্ভাষণ
  cout << "try me again" << endl;
```

উপরের program (ক্রমলেখ) দেখো, আমরা একটা infinite loop নিয়েছি while (true) { } লিখে তুমি চাইলে কিন্তু for (; ;) { } লিখে এমন কি do { } while(true); লিখেও infinite loop তৈরী করতে পারতে। Infinite loop তৈরী করলে আমাদের অবশ্যই loopএর ভিতরে কোন ভাবে loop থেকে break (ক্ষান্তি) দেওয়ারও ব্যবস্থা রাখতে হবে।

যাইহোক loopএর (ঘূর্ণী) ভিতরে দেখো আমরা প্রথমে কত serial নম্বর input (যোগান) দিলে কী করা হবে সেটি দেখিয়েছি, যেখানে ১ হলে যোগ, ২ হলে বিয়োগ, ৩ হলে গুণ, ৪ হলে ভাগফল, ৫ হলে ভাগশেষ, আর ০ হলে বের হয়ে যাওয়া আছে। Userএর (ব্যবহারকারী) পছন্দ prompt (যাচনা) করে choice variableএ নেয়া হয়েছে। এবার দেখো choiceএর মান ০ হলে control (নিয়ন্ত্রণ) loop থেকে break দিয়ে বের হবে, তাহলে infinite loop (অসীম ঘূর্ণী) আর হচ্ছে না। আর choice এর মান ০ না হলে তারপর আমরা পরীক্ষা করে দেখেছি সেটি ১ এর কম বা ৫ এর বেশী কিনা। যদি সেরকম হয় তাহলে এইরকম উল্টাপাল্টা পছন্দের জন্য আসলে আমাদের কিছু করার নেই, আমরা কলন করবো কেবল ১ হতে ৫ পর্যন্ত serial নম্বরের জন্য। তা-

হলে এই রকম ক্ষেত্রে আমাদের কলন করা বাদ দিয়ে সরাসরি পরের পাকে চলে যেতে হবে, অর্থাৎ আবার menu (প্রাপণ্য) দেখিয়ে পছন্দ input নিতে হবে। আমরা এই কাজটি করেছি continue ব্যবহার করে পাক ডিঙিয়ে। এরপরে দেখে আমরা একটি switch-case (পল্টি-ব্যাপার) ব্যবহার করে পছন্দ অনুযায়ী ফলাফল কলন করেছি, তারপর output (ফলন) দিয়েছি।

While loop (ক্ষণ ঘূর্ণী) বা do loopএ (করো ঘূর্ণী) break (ক্ষান্তি) দেয়া ঠিক for loopএ (জন্য ঘূর্ণী) break দেয়ার মতোই। Control (নিয়ন্ত্রণ) loop (ঘূর্ণী) থেকে বের হয়ে loopএর বাইরে যা আছে সেখানে চলে যাবে। তবে while loop (ক্ষণ ঘূর্ণী) বা do loopএ (করো ঘূর্ণী) পাক continueএর (ডিঙানো) সাথে for loopএ (জন্য ঘূর্ণী) পাক continueএর (ডিঙানো) কিঞ্চিত তফাৎ আছে। তফাৎটা হলো continue এর পরে for loopএ (জন্য ঘূর্ণী) control (নিয়ন্ত্রণ) update (হালায়ন) অংশে চলে যায়। কিন্তু while loop (ক্ষণ ঘূর্ণী) বা do loopএ (করো ঘূর্ণী) এ update অংশতো আলাদা করে নাই। Update সাধারণত statementএর অংশ হিসাবেই করা হয় যেমন প্রতিবার এখানে choice এর মান input নেয়া হয়েছে।, কাজেই control while বা do loopএর ক্ষেত্রে সরাসরি চলে যায় condition checking (শর্ত) অংশে।

১৬.১৭ Loop and If Interaction (ঘূর্ণী যদি মিথস্ক্রিয়া)

এমন একটি program (ক্রমলেখ) লিখো যেটি একটি positive integer (ধনাত্মক পূর্ণক) input (যোগান) নিয়ে ১ থেকে সেই integer (পূর্ণক) পর্যন্ত জোড় সংখ্যাগুলো একদিকে আর বিজোড় সংখ্যাগুলো আরেকদিকে যোগ করবে। এই programএ (ক্রমলেখ) মুলত আমরা loopএর (ঘূর্ণী) ভিতরে if-else (যদি-নাহলে) ব্যবহার না করতে চেষ্টা করবো।

```
cout << "positive integer ";
int integer; cin >> integer;

int evenSum = 0, oddSum = 0;
for(int index = 1; index <= integer; ++index)
{
  if (index % 2 != 0) // বিজোড়
    oddSum += index;
  else     // জোড়
    evenSum += index;
}

cout << evenSum << " " << oddSum << endl;
```

উপরের programএ (ক্রমলেখ) আমরা প্রথমে একটি positive integer (ধনাত্মক পূর্ণক) input (যোগান) নিয়েছি, তারপর জোড় সংখ্যাগুলোর যোগফলের জন্য evenSum আর বিজোড় সংখ্যাগুলোর যোগফলের জন্য oddSum চলক (variable) নিয়েছি। তারপর একটি for loop (জন্য ঘূর্ণী) চলেছে variable index এর মান ১ থেকে integerএর মান পর্যন্ত। Loopএর ভিতরে index বিজোড় হলে বা (index % 2 != 0) শর্ত সত্য হলে indexএর মান oddSum এর সাথে যোগ হবে আর শর্ত মিথ্যা হলে evenSumএর সাথে যোগ হবে।

উপরের ওই program (ক্রমলেখ) আমাদের সঠিক ফলাফল দিবে তবে একটা বিষয় খেয়াল করো indexএর মান জোড় নাকি বিজোড় এইটা কেন আমাদের loopএর (ঘূর্ণী) প্রতি পাকে (lap)

পরীক্ষা করতে হবে? এইটা তো আমরা আসলে আগে থেকে জানিই কোন পাকে indexএর মান জোড় কোন পাকে সেটা বিজোড়। কাজেই loopএর (ঘূর্ণী) ভিতরে যে শর্ত পরীক্ষণ সেটা আসলে আমাদের অতিরিক্ত হয়েছে বলে মনে হচ্ছে। এই রকমের ক্ষেত্রে আমরা আসলে একটা loopএর (ঘূর্ণী) বদলে দুটো loop লিখে ফেলতে পারি। নীচে দেখো আমরা তাই করেছি। প্রথম loopএ ১ থেকে শুরু করে প্রতিপাকে ২ করে বাড়বে, ফলে কেবল বিজোড় সংখ্যাগুলোই হবে indexএর মান, আর দ্বিতীয় loopএ ২ থেকে শুরু করে প্রতিপাকে দুই করে বাড়বে, ফলে কেবল জোড় সংখ্যাগুলোই হবে indexএর মান। প্রথম loopএ indexএর মান oddSumএর সাথে আর দ্বিতীয় loopএ indexএর মান evenSumএর সাথে যোগ করা হয়েছে। তো এইরূপ বিভাজনের ফলে আমাদের আর কোন loopএই (ঘূর্ণী) শর্ত পরীক্ষা করতে হলো না, অথচ একই ফলাফল পাওয়া গেলো।

```
cout << "positive integer ";
int integer; cin >> integer;

int evenSum = 0, oddSum = 0;
for(int index = 1; index <= integer; index += 2)
  oddSum += index;

for(int index = 2; index <= integer; index += 2)
  evenSum += index;

cout << evenSum << " " << oddSum << endl;
```

এবার একই রকমের আরেকটি ব্যাপার দেখো নীচের programএ (ক্রমলেখ)। এইখানে indexএর মান জোড় নাকি বিজোড় সেটার ওপর ভিত্তি করে যোগ না করে, loopএর (ঘূর্ণী) প্রতি পাকে একটা করে নম্বর input (যোগান) নেয়া হয়েছে। Input নেয়া নম্বরটি যদি জোড় হয় তাহলে evenSumএর সাথে আর বিজোড় হলে oddSumএর সাথে যোগ করা হয়েছে। এই programএ (ক্রমলেখ) loopএর (ঘূর্ণী) ভিতরে থাকা if-else (যদি-নাহলে) চাইলেও দুটো loopএ বিভাজন করা সম্ভব না। কারণ এখানে আমাদের আগে থেকে বুঝার উপায় নেই input নেয়া নম্বরটি জোড় হবে নাকি বিজোড় হবে! কাজেই if-else loopএর ভিতরেই থাকবে।

```
cout << "positive integer";
int integer; cin >> integer;

int evenSum = 0, oddSum = 0;
for(int index = 1; index <= integer; ++index)
{
  int number; cin >> number;   // যোগান
  if (number % 2 != 0)   // বিজোড়
    oddSum += number;
  else     // জোড়
    evenSum += number;
}

cout << evenSum << " " << oddSum << endl;
```

তারপর আরো একটি একইরকম ব্যাপার দেখা যাক। এখানে আমাদেরকে integer (পূর্ণক) variableএর মান ১০এর কম হলে আমরা indexএর মানগুলো smallSumএ যোগ করতে চাই, আর integer variableএর মান ১০ বা বেশী হলে indexএর মানগুলো largeSumএ পেতে চাই। তো নীচের programএ (ক্রমলেখ) আমরা একটা loop (ঘূর্ণী) ব্যবহার করে লিখেছি, আর loopএর ভিতরে রয়েছে শর্ত পরীক্ষা (integer < 10)। শর্ত সত্য হলে smallSumএ যোগ আর শর্ত মিথ্যা হলে largeSumএ যোগ। এখানেও আমাদের একই রকমের সমস্যা, শর্ত পরীক্ষণ কি loopএর (ঘূর্ণী) ভিতরে দরকার আছে? নাকি এটাকে loopএর বাইরে নেয়া সম্ভব?

```
int integer;  cin >> integer; // চলকের মান যোগান
int smallSum = 0, largeSum = 0;

for (int index = 1; index <= integer; ++index)
  if (integer  < 10)
    smallSum += index;
  else
    largeSum += index;

cout << smallSum << " " << largeSum << endl;
```

একটু খেয়াল করলেই একটা বিষয় নজরে আসে সেটা হলো integer < 10 শর্তটি আসলে কোন ভাবেই indexএর মানের সাথে সম্পর্কিত নয়, ফলে এই শর্ত পরীক্ষণ আসলে পাকের ওপর নির্ভর করে না। সুতরাং আমরা চাইলে এই শর্তটিকে loopএর (ঘূর্ণী) বাইরে নিয়ে যেতে পারি। নীচের program (ক্রমলেখ) খেয়াল করো আমরা তাই করেছি। Loopএ যাওয়ার আগেই আমরা শর্ত পরীক্ষা করেছি। শর্ত (integer < 10) সত্য হলে আমরা একটা loopএ smallSum নির্ণয় করেছি, আর শর্ত মিথ্যা হলে আরেকটি loopএ (ঘূর্ণী) largeSum নির্ণয় করেছি।

```
int integer;  cin >> integer; // চলকের মান যোগান
int smallSum = 0, largeSum = 0;

if (integer  < 10)
  for (int index = 1; index <= integer; ++index)
    smallSum += index;
else
  for (int index = 1; index <= integer; ++index)
    largeSum += index;

cout << smallSum << " " << largeSum << endl;
```

উপরের এই programএ (ক্রমলেখ) শর্ত মাত্র একবার পরীক্ষা হলো, প্রতি পাকে একই শর্ত বারবার পরীক্ষা করার ব্যাপার আর রইলো না। কাজেই আমাদের program খানিকটা দক্ষ হয়ে গেলো, এটা চালাতে সময় কম লাগবে, ঠিক যেটা আমরা করতে চেয়েছিলাম।

১৬.১৮ Nested Independent Loops (অন্তান্তি স্বাধীন ঘূর্ণী)

এমন একটি program (ক্রমলেখ) লিখো যেটি একটি দুই-মাত্রার ছকের বর্গগুলোর ক্রমিক নম্বর নীচের মতো করে লিখবে। খেয়াল করো row (আড়ি) ক্রমিক ১ থেকে ৪, কিন্তু column (খাড়ি) ক্রমিক উল্টো দিকে ৪ থেকে ১। এই program তুমি দুটো nested independent loop (অন্তান্তি স্বাধীন ঘূর্ণী) ব্যবহার করে অর্থাৎ loopএর ভিতরে loop ব্যবহার করে লিখবে। চাইলে for loop (জন্য ঘূর্ণী), while loop (ক্ষণ ঘূর্ণী), do loop (করো ঘূর্ণী) ব্যবহার করতে পারো।

```
(1,4) (1,3) (1,2) (1,1)
(2,4) (2,3) (2,2) (2,1)
(3,4) (3,3) (3,2) (3,1)
(4,4) (4,3) (4,2) (4,1)
```

তো চলো আমরা প্রথমে এই programটি (ক্রমলেখ) for loop (জন্য ঘূর্ণী) দিয়ে লিখি। প্রথমে চলো আমরা প্রথম rowএর (আড়ি) দিকে নজর দেই (1,4) (1,3) (1,2) (1,1)। এই rowতে চারটি ক্রমিক দেখানো হয়েছে, প্রতিটি ক্রমিকে দুটি করে সংখ্যা আছে, প্রথমটি row নম্বর, আর দ্বিতীয়টি column (খাড়ি) নম্বর। তো সবগুলো ক্রমিকের row নম্বরই 1, কেবল column নম্বর বদলে গেছে 4 থেকে শুরু করে 1 পর্যন্ত, প্রতিবার 1 করে কমবে। কাজেই আমরা loop (ঘূর্ণী) চালাবো কেবল columnএর জন্য আর rowনম্বরটি প্রত্যেক ক্ষেত্রে সরাসরি outputএ দেখাবো।

```
for(int col = 4; col >= 1; --col)
    cout << "(" << 1 << "," << col << ") ";
```

উপরের এই programএর output যদি দেখো, তাহলে নীচের মতো লাগবে।

```
(1,4) (1,3) (1,2) (1,1)
```

আমাদের যে output (ফলন) দিতে বলা হয়েছে সেখানে যেহেতু চারটি row (আড়ি) আছে, সেহেতু উপরের programএর (ক্রমলেখ) মতো loop (ঘূর্ণী) আমরা চারবার লিখলেই কাঙ্খিত output (ফলন) পেয়ে যাবো। তবে প্রতিটা loopএ কেবল row নম্বরের জায়গায় নীচের মতো করে 1এর বদলে 2, 3, 4 লিখে নিতে হবে। আর প্রতিটি rowএর পরে পরের rowতে output (ফলন) যাওয়ার জন্য আমাদের cout << endl; লিখতে হবে।

```
for(int col = 4; col >= 1; --col)
    cout << "(" << 1 << "," << col << ") ";
cout << endl;
for(int col = 4; col >= 1; --col)
    cout << "(" << 2 << "," << col << ") ";
cout << endl;
for(int col = 4; col >= 1; --col)
    cout << "(" << 3 << "," << col << ") ";
cout << endl;
for(int col = 4; col >= 1; --col)
    cout << "(" << 4 << "," << col << ") ";
cout << endl;
```

১৬.১৮. Nested Independent Loops (অন্তান্তি স্বাধীন ঘূর্ণী)

চারটি rowএর (আড়ি) জন্য না হয় প্রায় একই রকম code (সংকেত) চারবার লিখলাম, কিন্তু আরো বেশী সংখ্যক rowএর জন্য নিশ্চয় অতবার লিখবো না। আমরা একই কাজ বার বার করার জন্যেই তো loop (ঘূর্ণী) ব্যবহার করা শিখেছিলাম, সেটা কি rowএর জন্যেও ব্যবহার করতে পারি না? নিশ্চয় পারি। নীচে দেখো আমরা তাই করলাম। আমরা প্রথমে rowএর জন্য একটি loop নিয়েছি যেটি row ১ থেকে ৪ পর্যন্ত প্রত্যেকবার ১ করে বাড়িয়ে চলবে। আর এই loopএর (ঘূর্ণী) blockএর (মহল্লা) ভিতরে থাকবে আমাদের আগের লেখা loopটা যেটা rowএর জন্য ঘুরে। Block (মহল্লা) ব্যবহার করতে হলো কারণ একটা loop আর একটা cout মোট দুটো statement (বিবৃতি) execute করতে হবে প্রতিটি rowএর জন্য। আর rowএর জন্যে লেখা ভিতরের loopটাতে যেখানে rowএর নম্বর 1, 2, 3, 4 সরাসরি লিখে দিয়েছিলাম, এবার সেখানে row variable লিখে দিলেই হয়ে গেলো।

```
for(int row = 1; row <= 4; ++row)
{
  for (int col = 4; col >= 1; --col)
    cout << "(" << row << "," << col << ")";
  cout << endl;
}
```

এবার চলো এই programটিই (ক্রমলেখটি) for loop (জন্য ঘূর্ণী) ব্যবহার না করে আমরা while loop (ক্ষণ ঘূর্ণী) ব্যবহার করে লিখে ফেলি। তবে row ও col সংখ্যা নির্দিষ্ট করে ৪ ধরে না নিয়ে আমরা এখানে rowCount ও colCount নামে দুটো variable (চলক) ব্যবহার করবো, যার মান তুমি চাইলে input (যোগান) নিতে পারো। For loop (জন্য ঘূর্ণী) থেকে while loop (ক্ষণ ঘূর্ণী) লেখা তো তেমন কঠিন কিছু নয়। For loopএর (জন্য ঘূর্ণী) initialisation (আদ্যায়ন) অংশটাকে while loopএর (ক্ষণ ঘূর্ণী) আগে লিখে ফেলো, আর জন্য loopএর (ঘূর্ণী) update (হালায়ন) অংশটাকে while loopএর statement (বিবৃতি) অংশের শেষে দিয়ে দাও। তুমি চাইলে একটা loopকে for loop রেখে আরেকটাকে while loop করে দিতে পারো। তা ছাড়া তুমি চাইলে for বা while loop বাদ দিয়ে do loopও ব্যবহার করতে পারো।

```
int rowCount = 4, colCount = 4;

int row = 1;
while (row <= rowCount)
{
  int col = colCount;
  while (col >= 1)
  {
    cout << "(" << row << "," << col << ") ";
    --col;
  }
  cout << endl;
  ++row;
}
```

১৬.১৯ Nested Dependent Loop (অন্তান্তি নির্ভরশীল ঘূর্ণী)

Loopএর (ঘূর্ণী) ভিতরে loop অর্থাৎ nested loop (অন্তান্তি ঘূর্ণী) ব্যবহার করে এমন একটি program (ক্রমলেখ) লিখো যেটি নীচের মতো output (ফলন) দিবে। এইক্ষেত্রে ভিতরের loopটি বাইরের loopএর ওপরে নির্ভরশীল হবে: ভিতরের loopএর indexএর (সুচক) মান বাইরেরটির indexএর মানের সাথে সম্পর্কিত হবে। তুমি তিন রকম loopএর (ঘূর্ণী) যে কোনটিই ব্যবহার করতে পারো।

```
(1,1)
(2,2) (2,1)
(3,3) (3,2) (3,1)
(4,4) (4,3) (4,3) (4,1)
```

আমরা প্রথমে নীচের মতো করে দুটো nested independent loop (অন্তান্তি স্বাধীন ঘূর্ণী) লিখে ফেলতে পারি যেটা আমরা ঠিক আগের পাঠেই শিখেছি। আলোচনার ধারাবাহিকতা বুঝার জন্য তুমি চাইলে আগের পাঠটি একটু দেখে নিতে পারো। এখানে বাইরের loop (ঘূর্ণী) ৪ বার চলবে, আর তার প্রতি পাকের জন্য ভিতরের loopটাও ৪ বার চলবে।

```
for(int row = 1; row <= 4; ++row)
{
  for (int col = 4; col >= 1; --col)
    cout << "(" << row << "," << col << ") ";
  cout << endl;
}
```

দুটো nested independent loop (অন্তান্তি স্বাধীন ঘূর্ণী) লেখার ফলে আমরা যে output (ফলন) পাবো তা নীচের বাম পাশের মতো। খেয়াল করে দেখো ছকের প্রতিটি rowতে (আড়ি) প্রতিটি columnএ (খাড়ি) সংশ্লিষ্ট ঘরের ক্রমিক নম্বর লেখা হয়েছে। এখন এটার সাথে আমাদের এই পাঠে যে output (ফলন) চাওয়া হয়েছে (ডান পাশেরটি) তা মিলাও।

```
(1,4) (1,3) (1,2) (1,1)                 (1,1)
(2,4) (2,3) (2,2) (2,1)             (2,2) (2,1)
(3,4) (3,3) (3,2) (3,1)         (3,3) (3,2) (3,1)
(4,4) (4,3) (4,2) (4,1)     (4,4) (4,3) (4,2) (4,1)
```

এবার একটা ব্যাপার খেয়াল করো যে ডান পাশের যে output (ফলন) চাওয়া হয়েছে সেখানে প্রত্যক rowতে (আড়ি) এমন একটা ঘর থেকে লেখা শুরু হয়েছে যেখানে row (আড়ি) আর column (খাড়ি) সমান, যেমন (1,1), (2,2), (3,3), (4,4)। Columnএর ক্রমিক rowএর ক্রমিক থেকে বড় হলে সেই ঘরে কিছু দেখানো হয় নাই। এটার জন্য আমরা আমাদের programএ (ক্রমলেখ) কেবল ভিতরের loopটি (ঘূর্ণী) কিঞ্চিত বদলে নিবো। ভিতরের loopটি আগে ছিলো `for (int col = 4; col >= 1; --col)`, এখন সেখানে নীচের মতো করে আদিমান 1 এর বদলে `row` লিখে দিবো। এর ফলে ভিতরের loopটি আর স্বাধীন থাকলো না, কারণ এটি কতবার ঘুরবে সেটা নির্ভর করবে বাইরের loopএ `row` এর মান কতো তার ওপর।

```
for(int row = 1; row <= 4; ++row)
{
```

```
    for (int col = row; col >= 1; --col)
      cout << "(" << row << "," << col << ") ";
    cout << endl;
  }
```

উপরের এই programটির (ক্রমলেখ) ফলে আমরা যে রকম output পাবো সেটি নীচের বাম পাশের মতো, কিন্তু আমরা যে output (ফলন) পেতে চাই তা ডান পাশের মতো।

```
(1,1)                                                   (1,1)
(2,2) (2,1)                                       (2,2) (2,1)
(3,3) (3,2) (3,1)                           (3,3) (3,2) (3,1)
(4,4) (4,3) (4,2) (4,1)           (4,4) (4,3) (4,2) (4,1)
```

খেয়াল করো এখনও ঠিক হয়ে ওঠে নি। যথাযথ ভাবে ফাঁকা দিলেই হয়ে যাবে। তো ফাঁকা দেওয়ার ক্ষেত্রে খেয়াল করো আমরা ১ম rowতে ফাঁকা দিয়েছি ৩টি, ২য়টিতে ২টি, ৩য়টিতে ১টি, ৪র্থটিতে ০টি। অর্থাৎ row এর মান অনুযায়ী 4−row সংখ্যক ফাঁকা দিয়েছি। অথবা বলতে পারো ৪ থেকে গোনা ও ফাঁকা দিতে শুরু করেছি, কিন্তু rowর চেয়ে বড় সংখ্যা পর্যন্ত ফাঁকা দিয়েছি, আর সমান হলে তো ঘরের স্থানাংক দেখানো শুরু করেছি। তাহলে সব মিলিয়ে আমরা নীচের programএর (ক্রমলেখ) মতো করে আরেকটি loop (ঘূর্ণী) ব্যবহার করতে পারি ফাঁকা দেওয়ার জন্য। এই loopটিও কত বার ঘুরবে, সেটা কিন্তু বাইরের loopএর ওপর নির্ভর করবে।

```
// প্রতি এরrow জন্য একটা করে পাক।
for(int row = 1; row <= 4; ++row)
{
  // প্রতি তেrow প্রথমে ফাঁকা দেওয়ার জন্য
  for (int col = 4; col > row; --col)
    cout << "      "; // মালার ভিতরে ছয়টি ফাঁকা

  // প্রতি তেrow ঘরগুলোর স্থানাঙ্ক লেখার জন্য
  for (int col = row; col >= 1; --col)
    cout << "(" << row << "," << col << ") ";
  cout << endl;
}
```

তুমি চাইলে কেবল এই সমস্যাটির ক্ষেত্রে ভিতরের loop দুটিকে Loop-If interaction (ঘূর্ণী যদি মিথস্ক্রিয়া) বিবেচনা করে একটা loop দিয়েই সারতে পারতে। কারণ ভিতরের দুটি loop মিলিয়ে তো 4 হতে 1 গুনতি চলে, colর মান row হতে বড় হলে ফাঁকা দেখানো হয় আর না হলে ঘরের স্থানাঙ্ক দেখানো হয়। সুতরাং একটা যদি-নাহলে (if else) লাগালেই হবে।

```
for(int row = 1; row <= 4; ++row)
{
  for (int col = 4; col >= 1; --col)
    if (col > row)
      cout << "      ";
    else
      cout << "(" << row << "," << col << ") ";
```

```
    cout << endl;
  }
```

১৬.২০ Deeply Nested Loops (গভীর অন্তান্তি ঘূর্ণী)

Loopএর (ঘূর্ণী) ভিতরে loop তার ভিতরে loop ব্যবহার করে তিনটি সংখ্যা 1, 2, 3 এর বিন্যাস (permutation) output (ফলন) দাও। বিন্যাসগুলোতে একই সংখ্যা বারবার ব্যবহার করা যাবে হলে কী করবে, আর একই সংখ্যা একের অধিকবাব ব্যবহার না করা গেলে কী করবে?

```
for (int x = 1; x <= 3; ++x)
  for (int y = 1; y <= 3; ++y)
    for (int z = 1; z <= 3; ++z)
      cout << x << " " << y << " " << z << endl;
```

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

এই programএর (ক্রমলেখ) জন্য আমরা এভাবে চিন্তা করি: প্রথম স্থানটিতে সংখ্যা তিনটি একে একে বসাতে হবে কাজেই একটা loop (ঘূর্ণী) লাগবে। তারপর দ্বিতীয় স্থানের জন্যেও সংখ্যা তিনটি একে একে বসাতে হবে, সুতরাং আরেকটা loop লাগবে। আর একই ভাবে তৃতীয় স্থানের জন্যেও আরেকটি loop দিয়ে সংখ্যা তিনটি একে একে বসাতে হবে। কাজেই সব মিলিয়ে আমাদের loop লাগবে তিনটি, আর বিন্যাস পাওয়া যাবে সর্বমোট ২৭ টি। তো এরকম একটি program আমরা উপরে দেখালাম, খুবই সহজ program। আর ওই programএর output (ফলন) কেমন হবে সেটাও উপরে দেখানো হয়েছে। তবে স্থানের ব্যবহার বাড়ানোর জন্য ২৭ টি বিন্যাস নীচে নীচে না লেখে তিন স্তম্ভে (column) দেখানো হয়েছে, আসলে ওগুলো একের পর এক নীচে নীচে আসবে।

উপরের programএ (ক্রমলেখ) কিন্তু একই সংখ্যা একের অধিকবার ব্যবহার করা হয়েছে। যদি সেটা করতে না দেয়া হয়, তাহলে আমরা যেটা করতে পারি তা হলো যখনই দুটি সংখ্যা এক হয়ে যাবে তখন আমরা output (ফলন) দিবো না। অর্থাৎ x যদি yএর সমান হয় অথবা x যদি z এর সমান হয়, অথবা y যদি zএর সমান হয় তাহলে output হবে না, আর না হলে output হবে। তার মানে output দেয়া হবে !(x == y || x == y || y == z) শর্ত সত্য হলে, আর দেয়া হবে না শর্ত মিথ্যা হলে। বুলক বীজগণিতের ডি মরগ্যানের সূত্রানুযায়ী আমরা এটাকে সরলীকরণ করতে পারি। তাহলে পাবো !(x == y) && !(x == z) && !(y == z) বা (x != y && x != z && y != z)। সবমিলিয়ে এমন program আর তার output হতে পারে নীচের মতো।

```
for (int x = 1; x <= 3; ++x)
```

```
    for (int y = 1; y <= 3; ++y)
      for (int z = 1; z <= 3; ++z)
        if (x != y && x != z && y != z)
          cout << x << " " << y << " " << z << endl;
```

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

একটা বিষয় খেয়াল করো উপরের programএ (ক্রমলেখ) তিনটি loopই কিন্তু তিনবার করে ঘুরবে, ফলে মোট ২৭ টি পাকই সম্পন্ন হবে, তবে এই ২৭টি পাকের মাত্র ৬টিতে output (ফলন) আসবে, বাঁকীগুলোতে if-elseএর (যদি-নাহলে) শর্ত মিথ্যা হওয়ায় output আসবে না। কথা হচ্ছে ওই যে ২১টি পাক যেগুলোতে কোন output আসবে না, সেগুলো কমানো সম্ভব কিনা। কিছু-টা তো সম্ভব। দ্বিতীয় loopএর কথা বিবেচনা করো, যখন আমরা জানিই যে y এর মান x এর সমান, তখন তো তৃতীয় loopটি ঘুরিয়ে লাভ নেই, আমাদের কোন output আসবে না। কাজেই আমরা if else interaction (যদি নাহলে মিথস্ক্রিয়া) বিবেচনা করে নীচের programএর (ক্রমলেখ) মতো করে if (x != y) কে তৃতীয় loopএর উপরে নিয়ে আসতে পারি। এই programএর ক্ষেত্রে কোন loop কত বার ঘুরবে? তুমি কি নিজে নিজে সেগুলো হিসাব করতে পারবে?

```
for (int x = 1; x <= 3; ++x)                    //৩ বার
  for (int y = 1; y <= 3; ++y)                  // ৩*৩ = ৯বার
    if (x != y)                                 // ৩*১ = ৩ বার মিথ্যা
      for (int z = 1; z <= 3; ++z)              // ৬*৩ = ১৮বার
        if (x != z && y != z)                   // ৬ বার সত্য
          cout << x << " " << y << " " << z << endl;
```

আচ্ছা তোমাকে তিনটি সংখ্যা না দিয়ে বরং চারটি বা পাঁচটি বা আরো বেশী সংখ্যার বিন্যাস (permutation) output (ফলন) দিতে বলা হয় তুমি কী পারবে তার জন্যে program (ক্রম-লেখ) লিখতে? নিশ্চয় পারবে, যতটি সংখ্যা নিয়ে বিন্যাস করতে হবে ততগুলো loop (ঘূর্ণী) নিলেই হয়ে গেলো। এই যে loop inside loop (ঘূর্ণীর ভিতরে ঘূর্ণী), তার ভিতরে loop, তার ভিতরে আরো loop এগুলো হলো deeply nested loop (গভীর অন্তান্তি ঘূর্ণী), যতটা ভিতরে একটা loop ততটা হলো তার গভীরতা। যেমন উপরের programএ সবচেয়ে ভিতরের loopএর গভী-রতা হলো ৩, মাঝখানেরটার গভীরতা হলো ২ আর বাইরেরটার গভীরতা হলো ১। আমরা সাধারণত খুব বেশী গভীরতার nested loop তৈরী করতে চাই না। যেমন আরো বেশী সংখ্যার বিন্যাস কর-তে গেলেই আমরা আর এ রকম deeply nested loop ব্যবহার করবো না, বরং আমরা অন্য কোন পদ্ধতির খোঁজ করবো। তাছাড়া এরকম deep loop আরো একটা ক্ষেত্রেও অসুবিধাজনক। যেমন ধরো তোমাকে input (যোগান) নিতে হবে কয়টা সংখ্যার বিন্যাস করতে চাও। তো সেটা-তো আগে থেকে মানে program লেখার সময় জানা সম্ভব না, কাজেই program লেখার সময় কত গভীরতা পর্যন্ত loop লিখবো সেটাও জানা সম্ভব না, আর তাই এরকম করে program লেখা আসলেই সম্ভব হবে না।

১৬.২১ Deflating Nested Loops (অন্তান্তি ঘূর্ণী হ্রাসকরণ)

ধরো তোমাকে এমন একটা program (ক্রমলেখ) লিখতে হবে যেটি একদিনের ২৪ ঘন্টায় প্রতি সেকেন্ডে সময় output (ফলন) দিবে ১০:৩৯:৪৬ এই ছাঁচে। এই program তোমাকে nested loop (অন্তান্তি ঘূর্ণী) ব্যবহার না করে কেবল একটি loop (ঘূর্ণী) ব্যবহার করেই লিখতে হবে।

ফিরিস্তি ১৬.৯: Displaying Clock Time (ঘড়ির সময় দেখানো)

```
for(int h = 0; h < 24; ++h)
  for(int m = 0; m < 60; ++m)
    for(int s = 0; s < 60; ++s)
      cout << h << ":" << m << ":" << s << endl;
```

প্রথমে আমরা nested loop (অন্তান্তি ঘূর্ণী) ব্যবহার করেই programটি (ক্রমলেখ) লিখি। আমাদের ঘন্টা চলবে ০ হতে ২৩ পর্যন্ত, মিনিট চলবে ০ হতে ৫৯ পর্যন্ত, আর সেকেন্ডও চলবে ০ হতে ৫৯ পর্যন্ত। সুতরাং ৩ depthএর nested loop হলেই আমাদের চলবে। উপরের programএ দেখো তিনটি loop একটার ভিতরে আরেকটা লিখে আমরা তা করেছি।

এবার আমরা nested loop (অন্তান্তি ঘূর্ণী) ব্যবহার না করে একটা loop ব্যবহার করে programটি (ক্রমলেখ) লেখার চেষ্টা করবো। সারাদিনে আমাদের মোট সেকেন্ড আছে কতটি? ২৪ * ৬০ * ৬০ = ৮৬৪০০টি। তাহলে আমাদের একটি loop চালাতে হবে ৮৬৪০০ বার। আর প্রতিবারে সেকেন্ডকে ৬০ দিয়ে ভাগ করে মিনিটে আর মিনিটকে ৬০ দিয়ে ভাগ করে ঘন্টায় প্রকাশ করতে হবে। তারপর অবশিষ্ট সেকেন্ড, অবশিষ্ট মিনিট, ও কত ঘন্টা হলো তা outputএ দেখাতে হবে। ভাগফল / আর ভাগশেষ % ব্যবহার করে আমরা এই program নীচের মতো করে লিখবো।

```
for(int k = 0; k < 86400; ++k)
{
  int h, m, s = k;  // k কে ঘন্টা মিনিট সেকেন্ড নিতে হবে

  m = s / 60;       // মিনিটে রূপান্তর
  s = s % 60;       // অবশিষ্ট সেকেন্ড

  h = m / 60;       // ঘন্টায় রূপান্তর
  m = m % 60;       // অবশিষ্ট ঘন্টা

  cout << h << ":" << m << ":" << s << endl;
}
```

তুমি কিন্তু চাইলে উপরের মতো করে প্রতিবার সেকেন্ডকে ৬০ দিয়ে ভাগ করে মিনিটে, তারপর আবার ৬০ দিয়ে ভাগ করে ঘন্টায় প্রকাশ না করে অন্যভাবেও করতে পারো। ধরো সেকেন্ড loopএর প্রতি পাকে ১ করে বাড়লো। আর যখন ৬০ সেকেন্ড হয়ে গেলো তখন আমরা মিনিটে এক যোগ করে দিলাম, আর সেকেন্ডকে আবার ০ বানিয়ে দিলাম। একই ভাবে মিনিট যদি ৬০ হয়ে যায় তাহলে ঘন্টাকে এক বাড়িয়ে দিলাম, আর মিনিটকে ০ বানিয়ে দিলাম। আর যখন ঘন্টা ২৪ হয়ে গেলো তখন program শেষ করে দিলাম। তো এই রকম program (ক্রমলেখ) আমরা নীচে দেখালাম।

```
int h = 0, m = 0, s = 0;     // আদি মান
```

```
while(h < 24)
{
  cout << h << ":" << m << ":" << s << endl;

  if (++s == 60)     // সেকেন্ড এক বাড়িয়ে ৬০ হলে
  {
    s = 0;           // সেকেন্ড হবে শুন্য
    if (++m == 60)   // মিনিট এক বাড়বে, আর ৬০ হলে
    {
      m = 0;         // মিনিট হবে শুন্য
      ++h;           // ঘন্টা এক বাড়বে
    }
  }
}
```

আসলে যে কোন nested loopকে (অন্তান্তি ঘূর্ণী) এই ভাবে কেবল একটা loop দিয়েই লিখে ফেলা যায়। Nested loopএ indexএর (সুচক) মানগুলো যে ক্রমে বদল হতে থাকে, উপরের এই একটা loopএও variableগুলোর মান সেই একই ক্রমেই বদল হতে থাকে।

১৬.২২ Nested Loop in Disguise (ছদ্মবেশের অন্তান্তি ঘূর্ণী)

Nested loop (অন্তান্তি ঘূর্ণী) ব্যবহার করে এবং না করে (1) + (1 + 2) + (1 + 2 + 3) + ... + (1 + 2 + ... + n) ধারাটির যোগফল নির্ণয়ের program (ক্রমলেখ) রচনা করো। এখানে তুমি 1 + 2 + ... + n = n(n + 1)/2 এ রকম সূত্র ব্যবহার করতে পারবে না।

প্রদত্ত ধারাটিতে বন্ধনীর ভিতরে অংশগুলোকে যদি একটা করে পদ ধরে নাও তাহলে প্রথম পদ (1), দ্বিতীয় পদ (1 + 2), আর এই ভাবে nতম পদ (1 + 2 + ... + n)। কাজেই উপরের ধারাটিতে আমাদের nটি পদ আছে, সুতরাং আমাদের একটি loop (ঘূর্ণী) লাগবে যেটি 1 থেকে n পর্যন্ত ঘুরবে। এবার বন্ধনীর ভিতরের প্রতিটি পদের দিকে তাকাই। ধরা যাক আমরা kতম পদ বিবেচনা করছি, তাহলে বুঝতেই পারছো পদটি হবে (1 + 2 + ... + k)। এখানে এই পদটি নিজেও একটা ধারা। কাজেই আমাদের পুরো ধারাটি আসলে ধারার ভিতরে ধারা, বা nested series (অন্তান্তি ধারা)। যাইহোক, kতম পদ (1 + 2 + ... + k) তো আমরা আরেকটি loop 1 থেকে k পর্যন্ত ঘুরিয়ে সহজেই হিসাব করে ফেলতে পারি। তাহলে সব মিলিয়ে প্রদত্ত ধারার জন্য আমাদের loop inside loop বা nested loop (অন্তান্তি ঘূর্ণী) ব্যবহার করতে হবে।

```
int n = 10;    // input নিতে পারো
int s = 0;     // পুরো ধারার যোগফল
for(int k = 1; k <= 10; ++k)
{
  int t = 0;   // বন্ধনীতে পদের যোগফল
  for(int l = 1; l <= k; ++l)
    t += l;    // বন্ধনীর ভিতরে যোগফল
  s += t;      // পুরো ধারার যোগফল
}
```

```
cout << s << endl;
```

উপরের programএ (ক্রমলেখ) দুটো nested loop (অন্তান্তি ঘূর্ণী) মিলিয়ে ঠিক কতবার ঘুরবে? বাইরের loopএ kএর মান যখন 1 তখন ভিতরের loop ঘুরবে 1 বা, বাইরের loopএ kএর মান যখন 2 তখন ভিতরের loop ঘুরবে 2 বার, এই ভাবে বাইরের loopএ kএর মান যখন n তখন ভিতরের loop ঘুরবে n বার। কাজেই বাইরের loopএর সব পাক মিলিয়ে ভিতরের loop ঘুরবে $1 + 2 + \ldots + n = n(n+1)/2$ বার। তার মানে nএর মান 10 হলে দুই loop মিলে পাক খাবে সর্বমোট $10(10+1)/2 = 55$ বার। কথা হচ্ছে এই ধারাটির যোগফল বের করতে আসলেই কি এত পাকের দরকার আছে? বিশেষ করে বন্ধনীর ভিতরের প্রতিটি পদ কেন আলাদা করে আবার নতুন করে হিসাব করতে হবে? আগের বন্ধনীর ভিতরের পদ জানা থাকলে তো তার সাথে কেবল পরের integer (পূর্ণক) যোগ করেই পরের বন্ধনীর ভিতরের পদ বের করা সম্ভব।

```
int n = 10;    // input নিতে পারো
int s = 0, t = 0; // ধারা ও পদের যোগফল
for(int k = 1; k <= 10; ++k)
{
  t += k;    // বন্ধনীর ভিতরে যোগফল
  s += t;      // পুরো ধারার যোগফল
}
cout << s << endl;
```

উপরের programএ দেখো আমরা শুরুতে পদ tএর initial value শুন্য ধরে নিয়েছি। আর loopএর ভিতরে ঢুকেই tএর সাথে k যোগ করে দিচ্ছি, যাতে বন্ধনীর ভিতরে থাকা আগের পদে t এর মান যত ছিলো, এই পাকে যাতে tএর মান তার চেয়ে যাতে k বেশী হয়, কারণ পরের বন্ধনীর ভিতরে পদে তো kটাই অতিরিক্ত আছে। তারপর t টাকে sএর সাথে যোগ করলেই ধারার যোগফল হয়ে গেলো। তো এই programএ loop কত বার ঘুরবে? বুঝতেই পারছো মাত্র 10 বার।

তাহলে আমরা দেখলাম দেখতে nested loop (অন্তান্তি ঘূর্ণী) মনে হলেও অনেক সময় একটা loop ব্যবহার করেই দক্ষ program (ক্রমলেখ) রচনা করা যায়। Nested loop লিখার সময় সেটা আসলেই nested loop নাকি স্রেফ ছদ্মবেশী এ ব্যাপারে সতর্ক থাকবে কেমন!

১৬.২৩ অনুশীলনী সমস্যা (Exercise Problems)

Conceptual Questions: নীচে কিছু conceptual প্রশ্ন রয়েছে। প্রশ্নগুলোর উত্তর নিজে নিজে বের করবে।

১. Iterative programming (পুনালি পরিগণনা) বলতে কী বুঝো? আলোচনা করো।

২. For loopএ (জন্য ঘূর্ণী) চারটি অংশ আছে। এগুলো হলো initialisation (আদ্যায়ন), condition (শর্ত), update (বৃদ্ধি), statement (বিবৃতি)। কোন অংশ কখন কতবার execute হয়, কার পরে কোনটি execute হয় আলোচনা করো।

৩. সমান্তর ধারার বর্তমান পদটিকে loopএর (ঘূর্ণী) indexএর সাথে সম্পর্কিত করা বনাম আগের পাকের সাথে সম্পর্কিত করার মধ্যে কী তফাৎ ঘটে আলোচনা করো।

৪. Loopএ (ঘূর্ণী) breakএর (ক্ষান্তি) ব্যবহার উদাহরণসহ আলোচনা করো।

৫. Loopএ (ঘূর্ণী) পাক continue (ডিঙানো) উদাহরণসহ আলোচনা করো।

৬. Loopএ (ঘূর্ণী) empty condition (শর্ত ফাঁকা) হলে loop থামবে কী করে?

৭. Infinite loop (অসীম ঘূর্ণী) কী? অসীম ঘূর্ণী কি কাঙ্খিত না অনাকাঙ্খিত?

৮. For loopকে (জন্য ঘূর্ণী) কী ভাবে সাধারণ (general) loop হিসাবে ব্যবহার করা যায়?

৯. পাকের আগে শর্ত পরীক্ষণ ও পাকের পরে শর্ত পরীক্ষণ বিষয়ে আলোচনা করো।

১০. loop ও if interaction (মিথস্ক্রিয়া) কী ভাবে programএর গতিতে প্রভাব ফেলে?

১১. অন্তান্তি (nested) একাধিক loopকে কেমনে একটা loop ব্যবহার করেই সামলানো যায়?

Programming Solutions: এবার আমরা programming সমস্যাগুলোর সমাধান দেখবো। মনে রাখবে সমাধানগুলো দেয়া হয়েছে তুমি যদি একান্তই নিজে করতে পারছোনা বলে মনে হয়, তখন কেবল একটু সাহায্য যাতে পেতে পারো তাই। কাজেই পারতপক্ষে সমাধান দেখবে না।

১. নীচের নকশার মতো নকশা তৈরী করো। এই নকশার কোনার বিন্দুগুলোতে + আছে, একদম বাম আর ডান পাশে আছে |, আর অন্য সবগুলো হলো –, প্রতিটি সারিতে – আছে ২০টি করে। প্রত্যেক সারির –গুলোর জন্য তোমাকে একটি করে loop (ঘূর্ণী) লিখতে হবে।

```
+————————————————————+
|————————————————————|
|————————————————————|
+————————————————————+
```

এই program (ক্রমলেখ) রচনা করা খুবই সহজ। আমাদের চারটি সারির জন্য চারটি for loop (জন্য ঘূর্ণী) লাগবে। প্রত্যেক সারির শুরু ও শেষে সংশ্লিষ্ট বিশেষ চিহ্নগুলো দিতে হবে। আর loop লাগবে মাঝখানের – চিহ্ন বারবার লেখার জন্য।

```
cout << "+";                    // উপরে বাম কোনা
for(int i = 0; i < 20; ++i)
  cout <<"–";                   // প্রথম সারি মাঝ
cout << "+" << endl;            // উপরে ডান কোনা
cout << "|";                    // দ্বিতীয় সারি শুরু
for(int i = 0; i < 20; ++i)
  cout <<"–";                   // দ্বিতীয় সারি মাঝ
cout << "|" << endl;            // দ্বিতীয় সারি শেষ
cout << "|";                    // তৃতীয় সারি শুরু
for(int i = 0; i < 20; ++i)
  cout <<"–";                   // তৃতীয় সারি মাঝ
cout << "|" << endl;            // তৃতীয় সারি শেষ
cout << "+";                    // নীচে বাম কোনা
for(int i = 0; i < 20; ++i)
  cout <<"–";                   // চতুর্থ সারি মাঝ
cout << "+" << endl;            // উপরে ডান কোনা
```

২. এমন একটি program (ক্রমলেখ) রচনা করো যেটি একটি ধনাত্মক (positive) পূর্ণক (integer) input (যোগান) নিয়ে সেটা মৌলিক (prime) সংখ্যা কিনা নির্ণয় করবে।

এই programটি (ক্রমলেখ) নানান ভাবে করা যেতে পারে। আমরা প্রথমে সবচেয়ে সহজ-টি কিন্তু সবচেয়ে ধীর গতির উপায়টি দেখি। একটি সংখ্যা n মৌলিক কিনা সেটার পরীক্ষা হলো একটি loop (ঘূর্ণী) চালিয়ে ২ থেকে শুরু করে $n-1$ পর্যন্ত প্রতিটি দিয়ে n বিভাজ্য কিনা পরীক্ষা করে দেখো। যদি একটি দিয়েও বিভাজ্য হয় তাহলে n মৌলিক নয়, আর সে-ক্ষেত্রে loop আর চালানো দরকার নেই, break (ক্ষান্তি) দিয়ে বের হয়ে আসতে হবে। আর loop যদি শেষ পর্যন্ত চলে, মানে loop indexএর (ঘূর্ণীর সূচক) মান যদি n হয়, তাহলে n মৌলিক। নীচে programটি (ক্রমলেখ) দেখো।

```
int n, k;                // n মুল সংখ্য, k সূচক
cout << "number: ";
cin >> n;

if (n <= 0)              // ঋণাত্মক কিনা পরীক্ষা
{
  cout << "negative" << endl;
  return EXIT_FAILURE;
}

// ২ থেকে n-1 পর্যন্ত কোনটি দিয়ে বিভাজ্য কিনা
for(k = 2; k < n; ++k)
    if (n % k == 0)
        break;           // বিভাজ্য হলে আগেই ক্ষান্তি

if (k == n)              // শেষ পর্যন্ত ঘূর্ণী চলেছে
    cout << "prime yes" << endl;
else                     // আগেই বের হয়ে এসেছে
    cout << "prime no" << endl;
```

একটু খেয়াল করলেই বুঝবে কোন সংখ্যা মৌলিক কিনা তার জন্য আসলে ২ থেকে n পর্যন্ত পরীক্ষা করা দরকার নেই। আসলে n/2 পর্যন্ত অথবা আরো ভালো করে বলতে গেলে n এর বর্গমূল পর্যন্ত পরীক্ষা করলেই চলে। কাজেই উপরের programটি চাইলে আমরা আর একট দক্ষ করে লিখতে পারি। নীচে আমরা কেবল পরিবর্তন সংশ্লিষ্ট অংশ দেখালাম।

ফিরিস্তি ১৬.১০: Whether a Number is Prime (মৌলিক সংখ্যা কিনা নির্ণয়)

```
for(k = 2; k < sqrt(n); ++k)     // sqart(n) পর্যন্ত
    if (n % k == 0)
        break;           // বিভাজ্য হলে আগেই ক্ষান্তি

if (k >= sqrt(n))        // শেষ পর্যন্ত ঘূর্ণী চলেছে
    cout << "prime yes" << endl;
```

```
else                    // আগেই বের হয়ে এসেছে
    cout << "prime no" << endl;
```

৩. এমন একটি program (ক্রমলেখ) লিখো যেটি দুটো ধনাত্মক পূর্ণক (integer) input নিয়ে তাদের গসাগু (HCF) ও লসাগু (LCM) নির্ণয় করে।

দুটি সংখ্যা a ও bএর গসাগু হলো এমন একটি সংখ্যা g যেটি দ্বারা a ও b উভয় সংখ্যা বিভাজ্য হয়। এই রকম একাধিক সংখ্যা থাকলে সবচেয়ে বড়টি হবে গসাগু। গসাগু বের করা হয়ে গেলে আমরা a ও b এর গুণফল কে গসাগু দিয়ে ভাগ করে লসাগু পেতে পারি। তো এই program (ক্রমলেখ) লিখতে আমরা ১ থেকে শুরু করে প্রতিটি সংখ্যা দিয়ে ভাগ করে দেখবো a ও b উভয় সংখ্যা বিভাজ্য কিনা। যদি বিভাজ্য হয় তাহলে ভাজকটি আমাদের গসাগু হতে পারে, আর গুণফলকে গসাগু দিয়ে ভাগ করে লসাগু পেতে পারি। তবে আমাদের এখানেই থেমে গেলে হবে না, কারণ এর চেয়ে বড় কোন সংখ্যা সাধারণ ভাজক হিসাবে পাওয়া যায় কিনা তা দেখতে হবে। তবে একটা বিষয় মনে রাখতে হবে গসাগু g কখনই a বা b কোনটার চেয়েই বড় হবে না, দুটোর চেয়েই ছোট হবে।

```
cout << "two numbers? ";      // যোগান যাচনা
int a, b;                     // চলক দুটি
cin >> a >> b;                // যোগান নেওয়া

if (a < 0 || b < 0)           // ঋণাত্মক কিনা?
{
  cout << "negative" << endl;
  return EXIT_FAILURE;
}

int p = a * b, h, l;     // গুণফল, গসাগু ও লসাগু

// একে একে পরীক্ষা করো উভয়ে বিভাজ্য কিনা
for(int k = 1; k <= a && k <= b; ++k)
  if (a % k == 0 && b % k == 0) // উভয়ে বিভাজ্য
    { h = k; l = p/k; }

cout << "hcf = " << h << endl;
cout << "lcm = " << l << endl;
```

চাইলে উপরের programকে আর একটু দক্ষ করতে পারো। যেহেতু গসাগু সংখ্যা দুটোর কোনটা থেকেই বড় হয়, কাজেই আমরা সংখ্যা দুটোর ছোটটি থেকে loop (ঘূর্ণী) শুরু করতে পারি। আর দুটোকে ভাগ করা যায় এমন সবচেয়ে বড় ভাজকটি যেহেতু আমাদের দরকার, আমরা তাই loopটি ছোট থেকে শুরু করে বড়র দিকে চালাবো, আর প্রথমটি পাওয়া মাত্র loop থেকে বের হয়ে আসবো।

```
// চলক ঘোষণা, input ও negative checking এখানে করো
int p = a * b, h, l;     // গুণফল, গসাগু ও লসাগু
```

```
int m = a > b ? b : a;  // দুটোর মধ্যে ছোটটি
// একে একে পরীক্ষা করো উভয়ে বিভাজ্য কিনা
for(int k = m; k; ++k)  // m হতে যতক্ষণ শুন্য নয়
  if (a % k == 0 && b % k == 0) // উভয়ে বিভাজ্য
    { h = k; l = p/k; break}

cout << "hcf = " << h << endl;
cout << "lcm = " << l << endl;
```

আরো এক ভাবে যেমন ক্রমাগত ভাগের মাধ্যমেও আমরা গসাগু নির্ণয় করতে পারি। প্রথমে একটি সংখ্যাকে ভাজক আর আরেকটিকে ভাজ্য ধরে নিয়ে ভাগশেষ বের করবো। তারপর আগের ভাজকটি হয়ে যাবে নতুন ভাজ্য আর ভাগশেষটি নতুন ভাগশেষ। তারপর আবার ভাগ ও ভাজকটিকে নতুন ভাজ্য, ভাগশেষকে নতুন ভাজক । এই করে চলবে যতক্ষণ ভাগ-শেষ শুন্য না হচ্ছে। আর সেই মুহূর্তের ভাজকটিই হবে গসাগু।

ফিরিস্তি ১৬.১১: Determining HCF and LCM (গসাগু ও লসাগু নির্ণয়)

```
int t, h, l;     // সাময়িক, গসাগু, লসাগু
int p = a * b;   // গুণফল

do
{
  t = a % b;     // ভাগশেষ নির্ণয়
  a = b;         // আগের ভাজক হবে নতুন ভাজ্য
  b = t;         // ভাগশেষটি হবে নতুন ভাজক
}
while(t);        // ভাগশেষ শুন্য হলে শেষ

h = a, l = p/a; // গসাগু ও লসাগু

cout << "hcf = " << h << endl;
cout << "losagu = " << l << endl;
```

৪. নীচের programএর (ক্রমলেখ) output (ফলন) কী হবে, computerএ (গণনি) না চালিয়ে বের করো। তারপর গণনিতে চালিয়ে তোমার হিসাব করা ফলাফল যাচাই করো। যদি কোন infinite loop (অসীম ঘূর্ণী) থেকে থাকে সেটাকে মেরামতো করো।

```
int n = 3;
while (n >= 0)     // প্রথম ঘূর্ণী
{
  cout << n * n << " ";
  --n;
}
cout << n << endl;
```

```
while (n < 4)       // দ্বিতীয় ঘূর্ণী
  cout << ++n << " ";
cout << n << endl;

while (n >= 0)      // তৃতীয় ঘূর্ণী
  cout << (n /= 2) << " ";
cout << endl;
```

উপরের programএর (ক্রমলেখ) output (ফলন) নীচে দেখানো হলো। প্রথম loop 3 থেকে শুরু করে 0 পর্যন্ত সংখ্যাগুলোর বর্গ দেখাবে। কাজেই 9 4 1 0 outputএ আসবে, তারপর প্রথম loopএর (ঘূর্ণী) ঠিক পরের cout এর কারণে আসবে −1। দ্বিতীয় loop 4 হওয়ার আগে পর্যন্ত প্রতিবার এক বাড়িয়ে সংখ্যাটি output দেখাবে। কাজেই আমরা পাবো 0 1 2 3 4, দ্বিতীয় loopএর ঠিক পরের coutএর কারণে 4 আরো একবার আসবে। তারপর তৃতীয় loopএ nএর মান শুন্য বা বেশী হলে আগে 2 দিয়ে ভাগ করবে তারপর output দিবে। তো 4 হতে শুরু করলে আমরা outputএ পাবো 2 1 0 কিন্তু একবার শুন্য হওয়ার পরে তারপর প্রতিবার 2 দিয়ে ভাগ করলেও nএর মান শুন্যই থাকবে। কাজেই loopএর (ঘূর্ণী) শর্ত কখনো মিথ্যা হবে না। কাজেই আমরা একের পর এক অসীম সংখ্যক বার শুন্য পেতে থাকবো। অর্থাৎ এটি একটি infinite loop (অসীম ঘূর্ণী) হয়ে যাবে।

```
9 4 1 0 −1
0 1 2 3 4 4
2 1 0 0 0 0 ....
```

Infinite loop ঠিক করতে চাইলে আমরা তৃতীয় loopএর শর্তটি n >= 0 বদলে n > 0 লিখে দিতে পারি। তাতে তৃতীয় loopএর কারণে output আসবে 2 1 0।

৫. একজন অনভিজ্ঞ programmer নীচের programটি (ক্রমলেখ) লিখেছে। Programটির indentation (ছাড়ন) দেখে যেমন মনে হচ্ছে programটি ঠিক তেমন output (ফলন) দিচ্ছে না। Programmer চেয়েছিলেন ১০ থেকে শুরু করে প্রতিবার ২ দিয়ে ভাগ করবেন আর ভাগফলের বর্গ দেখাবেন। ভাগ করতে গিয়ে শুন্য হয়ে গেলে থেমে যাবেন। সুতরাং তার কাঙ্খিত output হচ্ছে 25 4 1 কিন্তু programটি হতে সেরকম output আসছে না। তো তুমি প্রথমে এই program যেমন আছে তেমন রেখেই এর output নির্ণয় করো। আর সেক্ষেত্রে indentation কেমন হবে সেটাও দেখাও। তারপর কাঙ্খিত ফলাফল পেতে গেলে programএ কী পরিবর্তন করতে হবে সেটাও করে দেখাও।

```
int n = 10;
while (n > 0)
  n /= 2;
  cout << n * n << " ";
cout << endl;
```

উপরের programটিতে indentation দেখে মনে হয়ে loopএর (ঘূর্ণী) পরের দুই সারি loopএর আওতার মধ্যে। কিন্তু গঠনরীতি অনুযায়ী আসলে তা হবে না, কারণ এখানে বক্র বন্ধনী দেয়া নেই। ফলে কেবল n /= 2টাই loopএর আওতায়। কাজেই loop চলবে

ঠিকই, প্রতিবার 2 দিয়ে ভাগ হবে, আর শুন্য হলে loop থেমে যাবে। তারপর loopএর বাইরে থাকা coutএর কারণে আমরা 0এর বর্গ 0ই outputএ পাবো। ফলে output হবে কেবল 0। আর এই ক্ষেত্রে indentationএর বিষয়টি ঠিকঠাক করলে program দেখতে হবে নীচের মতো।

```
int n = 10;
while (n > 0)
  n /= 2;                 // কেবল এটি ঘূর্নীর ভিতরে
cout << n * n << " "; // ছাড়ন ঠিক করা হলো
cout << endl;
```

এবার কাঙ্খিত output (ফলন) পেতে গেলে আমাদের আসলে programএ বক্র বন্ধনী (curly brackets) { } ব্যবহার করে coutটাকেও loopএর একটি block (মহল্লা) তৈরী করে তার ভিতরে আনতে হবে। সুতরাং সেইক্ষেত্রে programটি হবে নীচের মতো।

```
int n = 10;
while (n > 0)
{                                 // মহল্লা শুরু
  n /= 2;                         // ঘূর্নীর ভিতরে ছিলোই
  cout << n * n << " ";     // ঘূর্ণীর ভিতরে এখন
}                                 // মহল্লা শেষ
cout << endl;
```

৬. নীচের programটি (ক্রমলেখ) কী করবে বর্ণনা করো। তারপর এটিকে এমন ভাবে আবার লিখো যাতে এতে while loopএর (ক্ষণ ঘূর্ণী) বদলে do loop (করো ঘূর্ণী) ব্যবহৃত হয়, কিন্তু সব মিলিয়ে programএর বৈশিষ্ট্য একই থাকে।

```
int n;
cout << "positive number: ";
cin >> n;

while (n <= 0)
{
  cout << "Not positive." << endl;
  cout << "positive number: ";
  cin >> n;
}
```

উপরের programটি (ক্রমলেখ) ধনাত্মক সংখ্যা দরকার এরকম input prompt (যোগান যাচনা) করে nএর মান input (যোগান) নিবে। তারপর n যদি ধনাত্মক না হয় তাহলে loopএর (ঘূর্ণী) ভিতরে ঢুকবে আর message (বার্তা) দেখাবে ধনাত্মক নয়, আর আবার input prompt করে nএর মান input নিবে। তারপর loopএর ভিতরে আবার পরীক্ষা করবে অধনাত্মক কিনা, এবং এই ভাবে চলতে থাকবে যতক্ষণ না nএর মান ধনাত্মক হচ্ছে। সব মিলিয়ে বলা যায়, কমপক্ষে একবার input prompt দিয়ে n এর মান input নেওয়া

হবে: loopএর বাইরের input prompt (যোগান যাচনা) ও input (যোগান) নেওয়াটা হলো সেটি। সুতরাং আমরা খুব সহজেই do loop (করো ঘূর্ণী) ব্যবহার করতে পারি এখানে।

```
int n;
do
{
  cout << "positive number: ";
  cin >> n;
  if (n <= 0)
    cout << "not positive." << endl;
}
while (n <= 0);
```

তুমি চাইলে নীচের মতো করেও লিখতে পারো, যেখানে আমরা ধনাত্মক হলে বরং loop (ঘূর্ণী) থেকে break (ক্ষান্তি) নিবো। আর সেক্ষেত্রে অবশ্য while(n <= 0) না লিখে আমরা কেবল while(true) ও লিখতে পারি। আবার চাইলে do loop (করো ঘূর্ণী) থেকে while loopএ (ক্ষণ ঘূর্ণী) ফেরতও যেতে পারি, যেখানে নীচের while(true) টাকে সরিয়ে নিয়ে গিয়ে do এর বদলে বসিয়ে দিবো।

```
int n;
do
{
  cout << "positive number: ";
  cin >> n;
  if (n > 0) break;
  cout << "not positive." << endl;
}
while (n <= 0);
```

৭. নীচের programটির (ক্রমলেখ) output কী? এটিকে এমন ভাবে বদলে লেখো যাতে block (মহল্লা) ব্যবহার না করেই একই ফলাফল পাওয়া যায়। তারপর programটিকে while loop (ক্ষণ ঘূর্ণী) ব্যবহার না করে for loop (জন্য ঘূর্ণী) ব্যবহার করে লিখো।

```
int i = 5;
while (i > 0)
{
  i = i + 1;
  cout << i << endl;
}
```

উপরের programএ (ক্রমলেখ) loopএর ভিতরে iএর মান আগে কমানো হচ্ছে তারপর সেটা outputএ (ফলন) দেখানো হচ্ছে। কাজটি আমরা outputএ দেখানোর সময়েই করতে পারি pre-increment (পূর্ব বৃদ্ধি) ব্যবহার করে, যা নীচে দেখানো হলো।

```
int i = 5;
while (i > 0)
  cout << —i << endl;
```

তুমি চাইলে for loop (জন্য ঘূর্ণী) ব্যবহার করে নীচের মতো করেও লিখতে পারো।

```
for (int i = 5; —i; )
  cout << —i << endl;
```

৮. এমন একটি program রচনা করো যেটি একটি loopএর (ঘূর্ণী) ভিতরে ব্যবহারকারীর কাছে থেকে একের পর এক একটি করে পূর্ণক (integer) input নিবে। Input নেওয়া সংখ্যাটি ধনাত্মক না হলে program থেকে বের হয়ে যাবে, আর ধনাত্মক হলে মানের ক্রমানুসারে সংখ্যাটির উৎপাদকগুলোকে পরপর এক সারিতে output (ফলন) দিবে, আর পরের সংখ্যা input নিতে চাইবে। Sample input-output (যোগান-ফলন) নিম্নরূপ:

```
> 0 factors <= 0 ends
number is? 36
factor list: 36 18 12 9 4 3 2 1
> 0 factors <= 0 ends
number is? —1
program finished!
```

আমরা এখানে একটা infinite loop (অসীম ঘূর্ণী) নিবো শুরুতে while(true) লিখে, তার মানে loopএর ভিতরে আমাদের অবশ্যই একটা break (ক্ষান্তি) দিতে হবে। তো নীচের programএ (ক্রমলেখ) দেখো আমরা loopএর ভিতরে যথাযথ input prompt (যোগান যাচনা) দিয়ে সংখ্যাটি input নিয়েছি। তারপর সংখ্যাটি ধনাত্মক না হলে break দিয়েছি, আর সেক্ষেত্রে loopএর বাইরে "program finished!" message (বার্তা) দেখিয়েছি। আর সংখ্যাটি ধনাত্মক হলে আমরা বড় থেকে ছোটর দিকে প্রতিটি সংখ্যা দিয়ে প্রদত্ত সংখ্যাটিকে ভাগ করেছি। ভাগশেষ শুন্য হওয়া মানে ভাজকটি একটি উৎপাদক, সেটি outputএ (ফলন) দেখাতে হবে। তুমি চাইলে এখানে do while (করো ঘূর্ণী) ব্যবহার করতে পারতে, আমরা সেটি তোমার নিজের চেষ্টার ওপরে ছেড়ে দিলাম, চেষ্টা করে দেখো।

ফিরিস্তি ১৬.১২: Display List of Factors (উৎপাদক তালিকা দেখাও)

```
while(true)
{
  cout << "> 0 factors <= 0 ends" << endl;
  int;  cin >> n;          // যোগান

  if (n <= 0) break;       // ঋনাত্মকে ক্ষান্তি

  cout << "factor list: ";
  for(int k = n; k > 0; —k)
    if (n % k == 0)
      cout << " " << k;
```

```
    cout << endl;
  }
  cout << "program finished!" << endl;
```

৯. এমন একটি program (ক্রমলেখ) রচনা করো যেটি একটি ধনাত্মক পূর্ণ সংখ্যা যেমন 23154 input (যোগান) নিয়ে output দিবে 45132।

```
int n, r, t; // নম্বর, উল্টা, সাময়িক
cout << "number? ";
cin >> n;
if (n <= 0)
{
  cout << "not positive" << endl;
  return EXIT_FAILURE;
}

r = 0;  // শুরুতে উল্টা নম্বর শুন্য
while(n > 0)
{
  t = n % 10;    // এককের অঙ্ক
  r = r * 10 + t; // উল্টার পিছে
  n = n/10;      // অবশিষ্ট অংশ
}

cout << "reverse = " << r << endl;
```

উপরের programএ প্রথমে উল্টা নম্বর r শুন্য ধরে নিয়েছি। তারপর n যতক্ষণ শুন্য না হচ্ছে ততক্ষণ loop (ঘূর্ণী) চালানো হবে। প্রতিবার nএর যে এককের অঙ্ক আছে সেটি নিয়ে r পিছে লাগিয়ে দিতে হবে। n এর এককের অংক পাওয়া যায় ১০ দিয়ে ভাগ করে ভাগশেষ নিলে। এখন rএ যা আছে তার পিছনে ওই অঙ্কটি লাগাতে হলে rএর মানকে আগে ১০ দিয়ে গুণ করে নিতে হবে কারণ এগুলো তে বামের দিকে এক ঘর সরে যাবে, আর তাতে ডানের যে স্থানটি ফাঁকা হলো সেখানে ওই অঙ্কটি বসিয়ে দিতে হবে, অর্থাৎ যোগ করতে হবে। Loopএর পরের পাকের জন্যে n হবে আগের পাকের এককের অঙ্ক ছাড়া বাঁকী অংশ, আর সেটি পাওয়া যাবে ১০ দিয়ে n কে ভাগ করে।

১০. এমন একটি program (ক্রমলেখ) রচনা করো যেটি একে একে সংখ্যা input (যোগান) নিবে যতক্ষণ ধনাত্মক সংখ্যা দেওয়া হচ্ছে, অধনাত্মক সংখ্যা হলে program শেষ হবে। Programটির output (ফলন) হবে input নেয়া সংখ্যাগুলোর মধ্যে সবচেয়ে বড়টি আর সেটি কত নম্বরে input দেওয়া হয়েছিলো সেই ক্রমিক নম্বরটি।

```
int large = 0, index = 0; // শুরুতে দুটোই শুন্য

int k = 0;       // ক্রমিক গোনার জন্য
while(true)
{
```

```
    cout << "number? ";     // যোগান যাচনা
    int n; cin >> n;        // যোগান নেওয়া

    if (n <= 0) break;      // অধনাত্মক সংখ্যা

    k = k + 1;          // আর একটি যোগান হলো

    if (n > large)      // এটি আগের বড়র চেয়েও বড়
    {
      large = n;        // এটি তাই নতুন বড়
      index = k;     // আর নতুন বড়টির সূচক
    }
  }

  if (k > 0)  // যদি কোন নম্বর যোগান হয়ে থাকে
  {
    cout << "large = " << large << endl;
    cout << "index = " << index << endl;
  }
```

উপরের programটির (ক্রমলেখ) শুরুতে আমরা বড় সংখ্যা হিসাবে আদিতে ধরে নিয়েছি শুন্য, যেটি input দেওয়া যে কোন ধনাত্মক সংখ্যার চেয়ে ছোট হবে। অনেকগুলো সংখ্যার মধ্যে সবচেয়ে বড় সংখ্যাটি বের করতে চাইলে আমরা সাধারণত শুরুতে ছোট একটা সংখ্যাকে ফলাফল হিসাবে ধরে নেই। যাতে সেটার চেয়ে তুলনা করে করে আরো বড় আরো বড় সংখ্যা পাওয়া যায়। তুমি যদি অনেকগুলো সংখ্যার মধ্যে সবচেয়ে ছোট সংখ্যাটি বের করতে চাও তাহলে তোমাকে শুরুতে বড় একটা সংখ্যাকে ফলাফল হিসাবে ধরে নিতে হবে। যাইহোক এরপর উপরের programটি দেখো input prompt করে input নিয়ে প্রথমে পরীক্ষা করেছে ধনাত্মক কিনা। ধনাত্মক না হলে loopএ (ঘূর্ণী) break (ক্ষান্তি) দিতে হবে আর না হলে যেহেতু আরেকটি ধনাত্মক সংখ্যা পাওয়া গেলো তাই ক্রমিক নম্বর এক বাড়বে। এরপর বর্তমানের বড়টি সাথে তুলনা করে যদি দেখা যায় নতুন নম্বরটি বড়, তাহলে নতুন নম্বরটিই হবে বড় আর তার indexটি আরেকটি variableএ নিতে হবে। Loopএর বাইরে কেবল বড় সংখ্যাটি আর তার index outputএ (ফলনে) যাবে।

১১. ফিবোনাস্যি (Fibonacci) প্রগমন হলো ০, ১, ১, ২, ৩, ৫, ৮, ...। লক্ষ্য করো এই প্রগমনের প্রথম দুটি পদ হলো ০ আর ১। আর এর পর থেকে প্রতিটি পদ তার আগের দুটো পদের যোগফল। এমন একটি program (ক্রমলেখ) রচনা করো যেটি একটি ধনাত্মক পূর্ণক n input (যোগান) নিয়ে nতম পদ নির্ণয় করবে।

ফিরিস্তি ১৬.১৩: Fibonacci Progression (ফিবোনাস্যি প্রগমন নির্ণয়)

```
int n; // মান তুমি যোগান নিয়ে ধনাত্মক কিনা পরীক্ষা করবে।

// প্রথম দুটো সংখ্যা আর তারপরেরটির চলক
int first = 0, second = 1, next;
```

```
if (n == 1) // প্রথমটি সরাসরি ফলন
  cout << "first = " << first << endl;
else if (n == 2)  // দ্বিতীয়টি সরাসরি ফলন
  cout << "second = " << second << endl;
else  // অন্য যে কোনটা ক্রমাগত হিসাব করতে হবে
{
  for(int k = 3; k <= n; ++k)
  {
    // k–তম পদ হিসাব করা হবে
    next = first + second;  // যোগ
    first = second; // পরেরটার জন্য ২য়টাই ১ম
    second = next;  // পরেরটার জন্য নতুনটা ২য়
  }
  // নতুন যেটি সেটি তমk পদ, সুতরাং ফলন
  cout << n << "th = " << next << endl;
}
```

প্রথমে n variable declare (চলক ঘোষণা) করে, তুমি input prompt (যোগান যাচনা) দিয়ে nএর মান input নিবে। তারপর n ধনাত্মক কিনা পরীক্ষা করে দেখবে। n ধনাত্মক না হলে একটি loopএর (ঘূর্ণী) ভিতরে আবার input নিতে পারো যতক্ষণ না ধনাত্মক মান দেওয়া হচ্ছে এই অংশটুকু করে দেওয়া হলো না, নিজের করো। আমরা কেবল ফিবোনাস্যি পদগুলো নির্ণয়ের অংশটুকু দেখি। প্রথম দুটি পদ ধারণ করার জন্য আমাদের first আর second নামে দুটি আর পরের পদের জন্য next নামে আরেকটি variable (চলক) আছে। যদি nএর মান ১ বা ২ হয় তাহলে আমরা তো প্রথম পদ দুটি থেকে কোনরূপ হিসাব করা ছাড়া সরাসরিই output দিতে পারি। আর যদি nএর মান 3 বা বেশী হয় তাহলে একটি for loopএ (জন্য ঘূর্ণী) আমরা প্রথমে next পদটি হিসাব করবো first ও second পদ দুটি যোগ করে। এবার তারওপরের পদটির জন্য আমাদের যা করতে হবে এই পাকে সেটা ঠিক করে রাখতে হবে, যাতে পরের পাকের শুরুতেই আমরা ওই পদটি হিসাব করতে পারি। তো next হিসাব করার পরে এখন আমাদের প্রথম পদটি হবে আগে যেটি দ্বিতীয় পদ ছিলো সেটি, আর দ্বিতীয় পদটি হবে next পদটি। এবং তারফলেই পরের পাকে first ও second যোগ করলে আমরা তারওপরের পদটি পাবো। Loop (ঘূর্ণী) চলবে এখানে ৩ থেকে n হওয়া পর্যন্ত। খেয়াল করো kএর মান যত আমরা loopএর (ঘূর্ণী) blockএর ভিতরে ততম পদটি হিসাব করছি, আর তারপরের পদটির জন্য প্রস্তুতি নিচ্ছি।

১২. একটি ধনাত্মক পূর্ণক n input নিয়ে প্রথম n স্বাভাবিক সংখ্যার (১, ২, ৩, ...) যোগফল ও গুণফল নির্ণয়ের program (ক্রমলেখ) রচনা করো। তুমি হয়তো জানো প্রথম n স্বাভাবিক সংখ্যার গুণফলকে factorial (উৎপাদকীয়) বলা হয়। Factorial খুবই বড় সংখ্যা হয় যা intএ নাও ধরতে পারে, কাজেই n বড় হলে আমরা উল্টাপাল্টা ফল পেতে পারি।

```
int n; // মান তুমি যোগান নিয়ে ধনাত্মক কিনা পরীক্ষা করবে।

// নীচের সারিতে যোগফল কেন ০ আর গুণফল ১?
int sum = 0, product = 1;
for(int k = 1; k <= n; ++k)
```

```
{
  sum += k;    // যোগ করো
  product *= k;    // গুণ করো
}
cout << "sum = " << sum << endl;
cout << "product = " << product << endl;
```

এই programটি (ক্রমলেখ) খুবই সহজ। আমরা n variable (চলক) ঘোষণা দেখিয়েছি, কিন্তু এর input prompt (যোগান যাচনা) ও input (যোগান) নেওয়া তুমি করবে। তারপর সেটি ধনাত্মক কিনা সেটিও পরীক্ষা করবে। এর পরে খেয়াল করো আমরা দুটি variable (চলক) নিয়েছি যোগফল ও গুণফলের জন্যে। মজার ব্যাপার হচ্ছে sumএর initial value (আদি মান) দিয়েছি ০ কিন্তু productএর initial value দিয়েছি ১। কেন বলতে পারবে? যোগের মানে আগে থেকে ১ থাকলে তো যোগফল সঠিক আসবে না, ১ বেশী আসবে, তাই initial value শুন্য। আর গুণফলের ক্ষেত্রে initial value ০ হলে এরপরের সকল গুণফলই তো ০ হয়ে যাবে, ১ দিলে সেটি হবে না। যোগ ও গুণের জন্য initial valueএর এই তফাৎ সবসময় মনে রাখবে। এই programএর বাঁকী অংশটুকুতো আর ব্যাখ্যা করছি না।

১৩. নীচের ধারাগুলোর প্রথম n পদের সমষ্টি নির্নয় করো। তোমার programএ (ক্রমলেখ) তুমি n input (যোগান) হিসাবে নিবে, আর ধারাটির সমষ্টি output (ফলন) দিবে।

ক) Sine series (লম্বানুপাত ধারা): $\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$

খ) Cosine series (লগ্নানুপাত ধারা): $\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$

গ) Harmonic series (ভাজিত ধারা): $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$

ঘ) Euler number (অয়লার সংখ্যা): $e_n = 1 + \frac{1}{1!} + \frac{1}{2!} + \ldots + \frac{1}{(n-1)!}$

ঙ) Geometric series (গুণোত্তর ধারা): $\frac{1}{1-x} = 1 + x + x^2 + \ldots + x^{n-1}$

এখানে আমরা শুধু $\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$ এর program (ক্রমলেখ) দেখাবো। বাঁকীগুলো তুমি একই রকম করে নিজে করো। আমাদের nএর পাশাপাশি xও input (যোগান) নিতে হবে। তবে n যেখানে ধনাত্মক পূর্ণক (positive integer) x সেখানে ভগ্নক (fractioner)। তারপর আমাদের একটি loop (ঘূর্ণী) নিতে হবে যেটি 1 থেকে n পর্যন্ত ঘুরবে, আর প্রতিপাকে kতম পদটি হিসাব করে যোগফলের সাথে যোগ করে দিবে। একটু খেয়াল করলে দেখবে এখানে kতম পদটি আসলে $-\frac{(-x)^k}{k!}$। সুতরাং আমরা পদ নির্ণয়ের জন্য একটি loop (ঘূর্ণী) চালিয়ে power (শক্তি) $(-x)^k$ নির্ণয় করবো, আরেকটি loop (ঘূর্ণী) চালিয়ে factorial (উৎপাদকীয়) $k!$ নির্ণয় করবো, আর তারপর পদটি যোগ করে দেবো যোগফলের সাথে।

```
int n;    // যোগান নাও
float x;  // যোগান নাও

float sum = 0;  // যোগফল
for(int k = 1; k <= n; ++k)
{
```

```
    float power = 1;     // (-x)^k
    float factorial = 1;  // k!

    for(int p = 1; p <= k; ++p)
    {
      power *= (-x);  // শক্তি নির্ণয়
      factorial *= p; // উৎপাদকীয়
    }
    sum += - power / factorial;
  }

  cout << sum << endl;
```

উপরের programটি আসলে খুব একটা দক্ষ হয় নি। কারণ এতে loopএর ভিতরে loop বা nested loop (অন্তান্তি ঘূর্ণী) ব্যবহৃত হয়েছে। আসলে ভিতরের loopটা ব্যবহার না করেই এই program (ক্রমলেখ) লেখা সম্ভব। একটা ব্যাপার খেয়াল করো আগের পদের সাথে আমরা কেবল $frac-x^2(2k-1)(2k-2)$ গুণ করলেই পরের পদ পাবো। কাজেই আগের পদ আরেকটা variableএ (চলক) ব্যবহার করে মনে রাখলে পরের পদ সহজে বের করা যাবে, এবং কোন loop (ঘূর্ণী) না চালিয়েই তা বের করা সম্ভব হবে।

```
  int n;    // যোগান নাও
  float x;  // যোগান নাও

  float sum = 0;  // যোগফল
  float term = x;
  for(int k = 1; k <= n; ++k)
  {
    sum += x;

    term *= -x*x/((2*k-1)*(2*k - 2));
  }

  cout << sum << endl;
```

১৪. দশটা সংখ্যা input (যোগান) নিয়ে তাদের গড় (mean) ও প্রমিত বিচ্যুতি (standard deviation) নির্ণয় করো। প্রমিত বিচ্যুতি হলো সংখ্যাগুলোর বর্গের গড় থেকে গড়ের বর্গ বিয়োগ করে বিয়োগফলের বর্গমূল অর্থাৎ $\sqrt{\frac{\sum x^2}{n} - (\frac{\sum x}{n})^2}$।

```
  int n; // কয়টি সংখ্যা যোগান নাও

  // নীচের চলক দুটি যোগফল ও বর্গের যোগফল
  float sumx = 0, sumx2 = 0
```

```
for(int k = 0; k < n; ++k)
{
  cout << k << "tom pod? "; // যাচনা
  float x; cin >> x;        // যোগান

  sumx += x;          // যোগফল নির্ণয়
  sumx2 += x*x;       // বর্গের যোগফল
}

float mean = sumx / n;     // গড়, নীচে প্রমিত বিচ্যুতি
float std = sqrt(sumx2 / n - mean * mean);
```

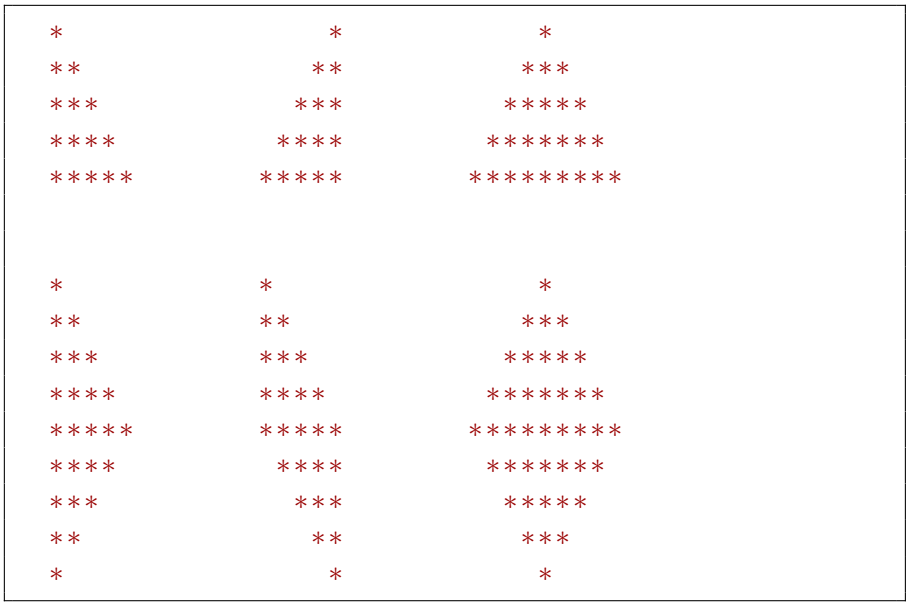
উপরের programটি (ক্রমলেখ) খুবই সাধারণ। একটি loopএর ভিতরে যোগফল ও বর্গের যোগফল বের করে নাও। তারপর loopএর বাইরে প্রথমে গড় আর তারপরে প্রমিত বিচ্যুতি নির্ণয় করো। এখানে sqrt() function (বিপাতক) ব্যবহার করে বর্গমূল নির্ণয় করা হয়েছে। কাজেই programএর শুরুতে #include <cmath> লিখে cmath header file (শির নথি) include (অন্তর্ভুক্ত) করে নিতে হবে।

১৫. একটি শ্রেণীতে n সংখ্যক শিক্ষার্থী আছে আর তাদের প্রত্যেকে m সংখ্যক বিষয়ে পরীক্ষা দিয়েছে। প্রত্যেক ছাত্রের প্রত্যেক বিষয়ের নম্বর input (যোগান) নিয়ে প্রত্যেক ছাত্রের মোট নম্বর output (ফলন) দাও।

```
int m, int n; // যোগান নাও

// প্রত্যেক ছাত্রের জন্য ঘূর্ণী
for(int i = 1; i <= n; ++i)
{
  float sum = 0;

  // প্রত্যেক বিষয়ের জন্য ঘূর্ণী
  for(int j = 1; j <= m; ++j)
  {
    float mark;
    cout << j << "th subject? ";
    cout << mark;

    sum += mark;
  }
  cout << "total marks = " << sum;
  cout << endl;
}
```

এই programএ (ক্রমলেখ) দুটো nested loop (অন্তান্তি ঘূর্ণী) লাগবে। প্রথমটি প্রতি শিক্ষার্থীর জন্য, আর দ্বিতীয়টি তাদের প্রতিটি বিষয়ের জন্য। বাঁকী অংশ সহজ, দেখে নাও।

১৬. তারকা ব্যবহার করে নীচের বিভিন্ন রকম আকৃতিগুলো output দাও। প্রতিটি ক্ষেত্রে তুমি দরকার মতো তুমি parameter (পরামিতি) n input (যোগান) নিবে। Parameter মানে হচ্ছে সারির সংখ্যা বা সারিতে সর্বোচ্চ কয়টি তারা থাকবে বা থাকবে না এগুলো n ওপর নির্ভরশীল। নীচের প্রতিটি ক্ষেত্রে nএর মান 5।



এই সব আকৃতির প্রতিটির জন্য আমাদের nested loop (অন্তান্তি ঘূর্ণী) লাগবে। আমরা এখানে কেবল উপরের ডানপাশেরটি করে দিবো। বাঁকীগুলো তুমি নিজে করবে।

```
int n = 5; // যোগান নিতে পারো

// প্রতিটি আড়ির (row) জন্য ঘূর্ণী
for(int row = 1; row <= n; ++row)
{
  // শুরুতে ফাঁকা আছে n – row সংখ্যক
  for(int col = 1; col <= n – row; ++col)
    cout << "–";
  // মাঝখানে তারা আছে 2 * row – 1 সংখ্যক
  for(int col = 1; col <= 2*row – 1; ++col)
    cout << "*";
  // শেষে ফাঁকা আছে n – row সংখ্যক
  for(int col = 1; col <= n – row; ++col)
    cout << "–";
  cout << endl;
}
```

এই program (ক্রমলেখ) লেখার সুবিধার্থে আমরা প্রথমে outputটাকে (ফলন) একটু বদলে লিখি। মূলত ফাঁকাগুলোকে (space) নীচের ডানপাশের মতো করে – বসিয়ে চিন্তা করি। তাহলে ফাঁকা গুনতে সুবিধা হওয়ায় নকশাটির ধাঁচ (pattern) পাওয়া সহজ

হবে, outputএর সময় আমরা ফাঁকাই লিখবো – নয়। মনে রাখবে এখানে আমাদের parameter ৫। এখানে ৫ টি সারি আছে। প্রতিটি সারিতে তারা অক্ষর আছে ৯টি মানে 2∗5 – 1। আরো খেয়াল করো ১ম সারিতে শুরু ও শেষে ফাঁকা আছে ৪টি করে, ২য় সারিতে ৩টি করে, ৩টি সারিতে ২টি করে। এসবের প্রতিটি ক্ষেত্রে যোগফল পাঁচ মানে ১ + ৪ = ২ + ৩ = ৩ + ২ = ৪ + ১। অর্থাৎ যততম সারি, শুরু ও শেষে ফাঁকার সংখ্যা ৫ থেকে তত বিয়োগ করলে পাওয়া যাবে। এরপর দেখো তারকার সংখ্যা হলো পরপর সারিতে ১, ৩, ৫, ৭, ৯, অর্থাৎ যততম সারি তার দ্বিগুণের চেয়ে এক কম সংখ্যক তারা আছে।

```
    *                ————*————
   ***               ———***———
  *****              ——*****——
 *******             —*******—
*********            *********
```

১৭. এক পরীক্ষায় মোট নম্বর ৯। কাজেই একজন ছাত্র ০ হতে ৯ পর্যন্ত যে কোন নম্বর পেতে পারে। এমন একটি program (ক্রমলেখ) রচনা করো যেটি এইরূপ ২০ জন ছাত্রের প্রত্যেকের নম্বর input (যোগান) নিয়ে তাদের নম্বর তারকা চিহ্ন দিয়ে আনুভূমিক চিত্রে দেখাবে। তিনজন ছাত্রের জন্য নমুনা input ও output (যোগান ও ফলন) নিম্নরূপ:

```
1th mark? 9
1: ********* (9)
2th mark? 5
2: ***** (5)
3th mark? 6
3: ****** (6)
```

এখানেও আমদের nested loop (অন্তান্তি ঘূর্ণী) লাগবে। প্রথম loopটি প্রতিটি ছাত্রের জন্য আর ভিতের একটা loop লাগবে দরকার মতো তারকা দেখানোর জন্য।

```
// প্রতিটি ছাত্রের ঘূর্ণী
for(int k = 1; k <= 20; ++k)
{
  // যাচনা করে নম্বর যোগান
  cout << k << "th mark? ";
  int mark; cin >> mark;

  // শুরুতে ক্রমিক নম্বর
  cout << k << ": ";

  // তারকা দেখানোর ঘূর্ণী
  for(i = 1; i <= mark; ++i)
    cout << "*";

  // শেষে বন্ধনীতে নম্বর দেখানো
  cout << " (" << mark << ")";
```

```
    cout << endl;
  }
```

১৮. ০ থেকে ৯ পর্যন্ত নামতার সারণী (table) লিখার জন্য একটি program রচনা করো। সারণীকে সুন্দর করে সাজানোর জন্য তোমাকে এক অঙ্কের সংখ্যা ও দুই অঙ্কের সংখ্যা মাথায় রাখতে হবে। এক অঙ্কের সংখ্যার জন্য তুমি প্রথমে একটি অতিরিক্ত ফাঁকা (space) দিয়ে নিবে। তাতে সারণীতে row and column (আড়ি ও খাড়ি) ঠিক মতো থাকবে। আর নামতার সারণীতে সবচেয়ে বামের columnএ (খাড়ি) আর উপরের rowতে (আড়ি) অবশ্যই ০ থেকে ৯ পর্যন্ত সংখ্যাগুলো থাকবে heading হিসাবে।

এই programটি একটু খুঁটিনাটি খেয়াল করে লিখতে হবে, যাতে সারণীটি আসলেই সুন্দর লাগে দেখতে। সর্বপ্রথমে সারণীর উপরের বাম কোনা খেয়াল করো সেটি অবশ্যই ফাঁকা হবে। আর প্রথম সারিতে আসলে সংখ্যাগুলো থাকবে ০ থেকে ৯ পর্যন্ত heading হিসাবে। দ্বিতীয় সারি থেকে মূলত গুণফলগুলো থাকবে। তবে প্রথমেই থাকবে heading হিসাবে সংখ্যা, আর তারপর গুণফলগুলো। গুণফল দেখানোর আগে অবশ্যই খেয়াল করবে ৯ বা ছোট কিনা, সেক্ষেত্রে একটা অতিরিক্ত ফাঁকা (space) দেখাতে হবে।

```
  cout << " ";        // উপরে বাম কোনা

  // প্রথম সারি হলো শির নাম
  for(int i = 0; i <= 9; ++i)
    cout << "  " << i;  // দুইটা ফাঁকা
  cout << endl;

  // প্রতিটি সংখ্যার জন্য সারি
  for(int i = 0; i <= 9; ++i)
  {
    cout << i;  // প্রথম খাড়ি

    // প্রতিটি সংখ্যার জন্য খাড়ি
    for(int j = 0; j <= 9; ++j)
    {
      int p = i * j;  // গুণফল

      cout << " ";      // পৃথকী
      if (p <= 9)       // এক অঙ্কের
        cout << " ";    // অতিরিক্ত ফাঁকা
      cout << p;        // গুণফল
    }
    cout << endl;       // সারি শেষ
  }
```

১৯. এমন একটি program (ক্রমলেখ) রচনা করো যেটি ৫০ জন ছাত্রের নম্বর input (যোগান) নিবে, আর output দিবে কোন পাল্লায় কতজন ছাত্রের নম্বর পড়েছে তা। প্রতি ১০ নম্বরের

জন্য একটি করে পাল্লা চিন্তা করতে পারো, আর মোট নম্বর হলে ৩০ যেখানে এক জন ছাত্র ১ থেকে ৩০ পর্যন্ত যে কোন নম্বর পেতে পারে। কাজেই মোটা পাল্লা রয়েছে ৩টি।

```
int first = 0, second = 0, third = 0;

for(int k = 1; k <= 50; ++k)
{
  cout << k << "th mark? ";
  int mark; cin >> mark;

  if (mark <= 0)
    cout << "pallar baire" << endl;
  else if (mark <= 10)
    ++first;
  else if (mark <= 20)
    ++second;
  else if (mark <= 30)
    ++third;
  else
    cout << "pallar baire" << endl;
}

cout << "first = " << first << endl;
cout << "second = " << second << endl;
cout << "third = " << third << endl;
```

২০. দ্বিপদী সহগ (binomial coefficient) হলো ${}^nC_r = \frac{n}{1} * \frac{n-1}{2} * \ldots * \frac{n-r+1}{r}$। দ্বিপদী সহগগুলো নিয়ে একটা ত্রিভুজ তৈরী করা যায় যাকে বলা হয় প্যাসক্যাল ত্রিভুজ। প্যাসক্যাল ত্রিভুজ নীচে দেখানো হলো। প্রথম সারিতে মানটি হলো ${}^0C_0$, দ্বিতীয় সারিতে মানগুলো হলো ${}^0C_1$ ও ${}^1C_1$, তৃতীয় সারিতে ${}^0C_2$, ${}^1C_2$ ও ${}^2C_2$। Parameter n input (যোগান) নিয়ে তার জন্য প্যাসক্যাল ত্রিভুজ output (ফলন) দাও। নীচের প্যাসক্যাল ত্রিভুজের parameter ৪। তুমি চাইলে উল্টিয়ে পাল্টিয়ে নানান রকমের প্যাসক্যাল ত্রিভুজ তৈরী করতে পারো।

```
    1                 ———1———
   1 1                ——1–1——
  1 2 1               —1–2–1—
 1 3 3 1              –1–3–3–1–
1 4 6 4 1             1–4–6–4–1
```

আমরা আপাতত ধরে নেই প্রতিটি সংখ্যাতে একটাই অঙ্ক থাকবে। বেশী অঙ্কের সংখ্যার জন্য তোমাকে আর একটু কষ্ট করে অতিরিক্তি ফাঁকা (space) ব্যবহার করতে হবে। যা-ইহোক নকশার ধাঁচ বুঝার জন্য আমরা ফাঁকাগুলোতে সাময়িক ভাবে – বসিয়ে নিয়েছি। আমাদের এখানে nএর মান ৪। তবে হিসাবের সুবিধার্থে আমরা এখানে গণনা শুরু করবো

০ থেকে। কাজেই ০তম সারিতে শুরু ও শেষে ফাঁকা রয়েছে ৪টি। ১ম সারিতে ফাঁকা রয়েছে ৩টি। এইভাবে ৪র্থ সারিতে ফাঁকা শুন্যটি। অর্থাৎ যত নম্বর সারি ৪ বিয়োগ তত হলো শুরু ও শেষের ফাঁকার সংখ্যা। আর প্রতি দুটো সংখ্যার মাঝখানে একটা ফাঁকা আছে অথবা বলতে পারে প্রতিটি সারির প্রথম সংখ্যাটি ছাড়া পরের সংখ্যাগুলোর সামনে ফাঁকা আছে।

```
int n = 4;  // যোগান নিতে পারো, শুন্যও নেয়া যেতে পারে

// প্রতিটি সারির জন্য
for(int k = 0; k <= n; ++k)
{
  // সারির শুরুতে ফাঁকা
  for(int i = 0; i < n - k; ++i)
    cout << " ";

  // মাঝখানে সংখ্যাগুলো
  for(int i = 0; i <= k; ++i)
  {
    // দুটি সংখ্যার মধ্যবর্তী ফাঁকা
    if (i > 0)
      cout << " ";

    // এবার সংখ্যাটি C(k, i) নির্নয়
    int product = 1;
    for(int j = 1; j <= i; ++j)
      product = product * (k - j + 1) / j;
      // উপরের সারিতে product *= (k - j + 1) / j;
      // লিখলে ঠিক মতো ফলন আসবে না,
      // কারণ পূর্ণকের ভাগ সমস্যা করবে

    cout << product;
  }

  // সারির শেষে ফাঁকা
  for(int i = 0; i < n - k; ++i)
    cout << " ";
  cout << endl;
}
```

২১. এমন একটি program (ক্রমলেখ) রচনা করো যেটি তোমার সাথে একটি খেলা খেলবে। Programটি তোমাকে মনে মনে ০ থেকে ১০২৩ এর মধ্যে একটি নম্বর মনে মনে ধরতে বলবে। আর তারপর programটি তোমাকে বেশ কিছু প্রশ্ন করবে, এই যেমন তোমার ধরেন নেওয়া সংখ্যাটি অমুক সংখ্যার চেয়ে বড় বা সমান নাকি ছোট। তুমি মূলত সত্য না মিথ্যা উত্তর দিবে। উত্তরগুলোর ভিত্তিতে programটি বলে দিবে তোমার ধরে নেওয়া সংখ্যাটি কতো? এই খেলায় আসলে প্রতিবার অর্ধেক করে পাল্লা কমাতে হয়। তাই এটাকে

bisection method (দ্বিখন্ডন পদ্ধতি) বলা হয়। এই পদ্ধতিটিকে অনেক সময় binary searchও (দুয়িক অনুসন্ধান) বলা হয় অবশ্য।

```
int low = 0, high = 1024;

cout << "assume a number in your mind" << endl;
cout << "the number should be 0—1023" << endl;
cout << "now answer the questions below" << endl
 ;
cout << "reply 1 if true, 0 if false" << endl;

do
{
  float mid = (low + high) / 2.0;
  cout << "number >= " << mid << "? ";
  int answer; cin >> answer;

  if (answer)
    low = mid;
  else
    high = mid - 1;
      cout << low << " " << high << endl;
}
while (low < high);
cout << "number = " << low << endl;
```

চলো আমরা উপরের programটি (ক্রমলেখ) বিশ্লেষণ করি। আমরা দুটি সংখ্যা নিয়েছি low আর high যেখানে আমরা আমাদের পাল্লা রাখতে চাই। আমরা পাল্লা হিসাবে ধরে নিলাম ০ আর ১০২৪। তো প্রথমবারে আমরা তাহলে প্রশ্ন করবো ধরে নেওয়া সংখ্যাটি অর্ধেক অর্থাৎ ৫১২ এর চেয়ে বড় বা সমান কিনা। উত্তর যদি হ্যাঁ হয় তার মানে ধরে নেওয়া সংখ্যাটি সর্বনিম্ন ৫১২, কাজেই lowকে আমরা বদলে করে ফেলতে পারি ৫১২। এতে আমাদের পাল্লা অর্ধেক হয়ে গেলো। এবার ৫১২ ও ১০২৪ এর মাঝখানেরটি ৭৬৮ দিয়ে একই রকম প্রশ্ন করবো। উত্তর যদি না হয় তার মানে ধরে নেওয়া সংখ্যাটি সর্বোচ্চ ৭৬৭, কাজেই high কে আমরা বদলে করে ফেলতে পারি ৭৬৭। এবারেও পাল্লা অর্ধেক হয়ে গেলো। এইভাবে চলতে থাকবে। উপরের program খেয়াল করো, দেখো এত ক্ষণ যা বললাম, তাই করেছি কি না? তাহলে হয়ে গেলো আমাদের মজার খেলা!

১৬.২৪ Computing Terminologies (গণনা পরিভাষা)

- Iterative (পুনালি)
- Loop (ঘূর্ণী)
- Index (সূচক)
- For loop (জন্য ঘূর্ণী)
- While loop (ক্ষণ ঘূর্ণী)
- Do loop (করো ঘূর্ণী)

১৬.২৪. Computing Terminologies (গণনা পরিভাষা)

- Repetition (পুনরাবৃত্তি)
- Initialisation (আদ্যায়ন)
- Update (হালায়ন)
- Continue (ডিঙানো)
- Flag (পতাকা)
- Infinite (অসীম)
- Interaction (মিথস্ক্রিয়া)