শর্টকাট প্রোগ্রামিং

মুহম্মদ জাফর ইকবাল

শর্টকাট প্রোগ্রামিং

মুহম্মদ জাফর ইকবাল

ভূমিকা

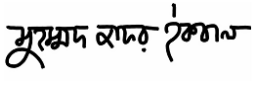
(কিংবা, সংবিধিবদ্ধ সতর্কীকরণ)

আমি এই বইটা কেন লিখেছি নিজেই জানি না। মাঝে মাঝেই বাচ্চাকাচ্চারা আমার সাথে যোগাযোগ করে, যারা প্রোগ্রামিং শিখতে চায় কিন্তু কীভাবে শুরু করবে বুঝতে পারে না। অনেক সুন্দর সুন্দর বই আছে, আমি তাদের সেই বইগুলোর নাম লিখে দেই, কেউ কেউ সেই বই পড়ে চমৎকার প্রোগ্রামিং করা শিখে যায়। আমার নিজেরও মাঝে মাঝে প্রোগ্রামিং করতে হয়, কিন্তু আমি খুব কম জানি, তাতেই আমার কাজ চলে যায়। তাই মাঝে মাঝেই আমার মনে হয়েছে, কত কম জেনে প্রোগ্রামিং করা যায় সেটা অন্যদের বলে দেই না কেন? শুরুতে ভেবেছিলাম আট-দশ পৃষ্ঠায় একটা তালিকা করে আমার ওয়েবসাইটে দিয়ে রাখব, যার শখ হবে দেখে নেবে।

লিখতে গিয়ে আবিষ্কার করলাম, একটা জিনিস না বুঝিয়ে লিখতে কেমন জানি অপরাধী মনে হয়। তাই দুই-এক কথায় ব্যাখ্যা দিতে গিয়ে লেখাটা একটু বড় হয়ে গেল। তখন আমি এক দুইজনকে এটা দেখিয়ে জিজ্ঞেস করলাম এটা ওয়েবসাইটে দিয়ে রাখলে বাচ্চাকাচ্চাদের ভুল জিনিস শেখানোর জন্য সবাই কি আমাকে গালমন্দ করবে?

যাদেরকে দেখিয়েছি তারা দেশের অনেক গুরুত্বপূর্ণ মানুষ, তারা বলল, করলে করুক, কিন্তু এটাকে একটা বই হিসেবেই বের করা উচিত। অনেকেরই উপকার হবে। প্রোগ্রামিং যে শুধু অল্প কিছু এলিটদের বিষয় নয়, চাইলে সবাই করতে পারে, সেটা বুঝতে পারবে। ভয় না পেয়ে শুরু করে দিলে অনেকেই হয়তো পরে সত্যিকারের প্রোগ্রামার হয়ে যাবে।

সেই জন্যে এটা শেষ পর্যন্ত বই হিসেবে প্রকাশিত হয়েছে!



মুহম্মদ জাফর ইকবাল
বনানী, ঢাকা
15 সেপ্টেম্বর, 2020

শর্টকাট প্রোগ্রামিং

শুরুর কথা

আমি আমার নিজের ব্যাপারে একটা বিচিত্র বিষয় লক্ষ করেছি। মাঝে মাঝে এমন হয় যে, আমি হয়তো সারাদিন পরিশ্রম করে ক্লান্ত, কিছুই করার ইচ্ছা করছে না। আমার সবসময়ই লেখালেখি করতে হয়, সেটাও করতে মন চাইছে না, বইও পড়ার ইচ্ছা করছে না। মাঝে মাঝে আমি আঁকা-আঁকি করি সেটাও ভালো লাগছে না। যেহেতু টেলিভিশন দেখি না, ফেসবুক করি না, সেগুলো করারও ঝামেলাই নেই। আমি লক্ষ করেছি, তখনও কিন্তু আমি খুব উৎসাহ নিয়ে প্রোগ্রামিং করতে পারি! কোনো গণিতের সমস্যা, পদার্থবিজ্ঞানের সমস্যা, ভাষার সমস্যা কিংবা শুধু মজা করার জন্য কোনো কিছু একটা করা।

কেন এটা হয় আমি জানি না, আমার ধারণা মানুষের মস্তিষ্ক কিছু একটা তৈরি করতে ভালোবাসে। কম্পিউটার প্রোগ্রামিং কিছু একটা তৈরি করার মতো, এটা শুধু যে কিছু একটা তৈরি করে তা নয়, যেটা তৈরি করা হয়েছে সাথে সাথে সেটা কিছু একটা করে দেখিয়ে দেয়!

আমি সারা জীবন শুধু যে মজা করার জন্য প্রোগ্রামিং করেছি তা নয়, কাজের কাজও করেছি। আমি যেহেতু পদার্থবিজ্ঞানের এক্সপেরিমেন্ট করেছি, তাই আমার প্রিয় কাজ ছিল কম্পিউটার দিয়ে যন্ত্রকে চালানো, ডেটা নিয়ে সেগুলো প্রক্রিয়া করা। প্রথম যখন

পার্সোনাল কম্পিউটার এসেছে, আমি বাংলা লেখার জন্য ‘ওয়ার্ড প্রসেসর’ তৈরি করেছি, ‘প্রিন্টার ড্রাইভার’ তৈরি করেছি। যখন দেশে ফিরে এসেছি, আমার তখন নিজের প্রোগ্রামিং করার প্রয়োজন নেই তখনও শখের জন্য প্রোগ্রামিং করেছি। বাংলা এবং ইংরেজি লেখার জন্য ‘ব্রেইল কনভার্টার’ এবং ‘ব্রেইল প্রিন্টার’ তৈরি করেছি। দাবা খেলার সফটওয়ার তৈরি করেছি (একটা বাচ্চা ছেলেও সেটাকে হারিয়ে দিতে পারবে—কিন্তু সেটা চেষ্টা করত)। আমার প্রোগ্রাম সুডোকু সমাধান করে দিত, রুবিকস কিউব খেলত ইত্যাদি ইত্যাদি।

আমি জানি সবাই ভাবছে, এই মানুষটার কি মাথায় গোলমাল হয়ে গেছে, নিজেকে নিয়ে এরকম বড় বড় গালগল্প করছে কেন? আসলে আমি এই গালগল্প করছি সবাইকে একটা জিনিস জানানোর জন্য। সেটা হচ্ছে, আমি কিন্তু প্রোগ্রামিং জানি না! প্রোগ্রামিংয়ের মাত্র ডজন খানেক স্টেটমেন্ট লিখতে পারি, সেগুলোই কোনোটা আগে কোনোটা পরে লিখে কাজ চালিয়ে ফেলি। কোড লিখতে লিখতে আমি মাঝখানে কোনো এক জায়গায় যখন আটকে যাই তখন আমি আমার কোনো ছাত্র কিংবা ছাত্রীকে ফোন করে জিজ্ঞেস করি, “অমুক স্টেটমেন্টটা জানি কীভাবে লিখতে হয় বলবে, প্লিজ।” তারা শুনে হি হি করে হাসে তারপর দয়া করে আমাকে বলে দেয়! আমি কখনো আমার লেখা কোড কাউকে দেখাই না, কারণ যারা প্রোগ্রামিং জানে তারা আমার কোড দেখলে হাসতে হাসতে মারা যাবে! এর কোনো নিয়ম নাই, ছিরি-ছাঁদ নেই, কিন্তু কাজ করে!

আমি তাই কিছুদিন থেকে ভাবছিলাম আমি যদি সত্যিকারের প্রোগ্রামিং না জেনে অল্প কয়েকটা জিনিস জেনেই এত মজা করতে পারি, সেগুলো নিয়ে এত কিছু করে ফেলতে পারি তাহলে সেটা অন্যদের বলে দেই না কেন? যারা ভাবে প্রোগ্রামিং করা বুঝি খুব কঠিন, যাদের মাথার ভেতরে শুধু মগজ আর মগজ, শুধু তারাই প্রোগ্রামিং করতে পারে, অন্যরা পারে না—তাদেরকে কেন বলে দিই না যে আসলে মজা করার জন্য কিংবা নিজের কোনো একটা কাজ করার জন্য প্রোগ্রামিং করা খুবই সোজা। সে জন্য আমি এই বইটা লিখেছি। বইটি পড়ার একটি মাত্র শর্ত—যারা আসলেই প্রোগ্রামিং জানে, তাদেরকে কখনও এই বইটা দেখানো যাবে না, তাহলে তারা হাসতে হাসতে মারা যাবে, আর সারা জীবন তোমাকে নিয়ে ঠাট্টা করবে!

প্রস্তুতি

কম্পিউটার প্রোগ্রাম আসলে কিছু ইংরেজি লেখা। অনেকভাবেই ইংরেজি কিছু একটা লেখা ফেলা সম্ভব, তাই প্রোগ্রামটা লিখে ফেলা এমন কিছু কঠিন নয়। কিন্তু সেটাকে কাজে লাগানোর জন্যে, সেটাকে প্রথমে কম্পিউটারের ভাষায় রূপান্তর করতে হয়, যেটাকে কায়দা করে বলা হয় কম্পাইলিং (compiling)। যদি নিয়ম মেনে সঠিকভাবে প্রোগ্রামটা লেখা হয়, তাহলে কম্পাইল শেষে সেটাকে ব্যবহার করা যায়, কম্পিউটারের ভাষায় সেটাকে বলে এক্সেকিউট (execute) করা বা রান (run) করা। কাজেই কেউ যদি কম্পিউটার

প্রোগ্রামিং করতে চায়, তার একটা কম্পিউটার দরকার হবে, টুকটাক কম্পিউটার ব্যবহার করায় অভ্যস্ত থাকতে হবে এবং সেই কম্পিউটারে একটা কম্পাইলার থাকতে হবে। (আমি যত দূর জানি আজকাল স্মার্ট ফোনেও কম্পাইলার পাওয়া যায়, কাজেই যার কাছে কম্পিউটার নেই তারাও স্মার্ট ফোনে প্রোগ্রামিং করতে পারবে।)

এই বইটা লেখা হয়েছে সি (C) কিংবা সি প্লাস প্লাস (C++) ভাষায় প্রোগ্রামিং করার জন্য, কাজেই এই ভাষার একটা কম্পাইলার দরকার। এ রকম অনেক কম্পাইলার আছে। এ দেশে জনপ্রিয় একটা কম্পাইলার হচ্ছে কোডব্লকস (CodeBlocks)। কাজেই কম্পিউটার প্রোগ্রামিং শুরু করার আগে তার কম্পিউটারে কোডব্লকস কিংবা কোডব্লকসের মতো কোনো একটা কম্পাইলার বসাতে হবে—কম্পিউটারের ভাষায় ইনস্টল (install) করতে হবে। সেটা করা হলে এবং সেটা ব্যবহার করা শিখে গেলে বলা যায় আর কোনো চিন্তা নেই, প্রোগ্রামিং করার অর্ধেক কাজ হয়ে গেছে!

এখন বাকি অর্ধেক কাজ শুরু করা যায়।

কী কী শিখতে হবে

যারা এখনো নার্ভাস হয়ে আছে তাদেরকে কী কী শিখতে হবে, সেটা আগে থেকে বলে দিয়ে একটু সাহস দেওয়া যাক। এই বইয়ে নিচের তালিকায় যা যা লেখা আছে তার বাইরে আর কিছু শিখতে হবে না। এগুলো ব্যবহার করে কী ধরনের প্রোগ্রাম করা যাবে, সেটা জানার

জন্য বইটা একটু উল্টেপাল্টে দেখ! যদি পছন্দ হয় বইটা পড়া শুরু করো, যদি পছন্দ না হয়, বইটা ছুড়ে ফেলে দাও।

```
#include <stdio.h>  #include <math.h>
int main() void SomeName()

{...}   //     /* ... */

int float char
VariableName    ArrayName[...]

printf(...); \n %d  %f  %c
scanf(...); &

+ - * / sin(...) cos(...)
for(...)  if(...)  while(...)

FILE ...; fprintf(...); fscanf(...);
```

উপরের তালিকায় যা দেয়া আছে তার অর্ধেকের মাঝে জানার বা শেখার কিছু নেই, প্রোগ্রামটা কাজ করার জন্য চোখ বন্ধ করে ব্যবহার করতে হবে। বাকিগুলো জানতে হবে এবং কিছু কিছু বুঝতেও হবে। যেখানেই এ রকম তিনটা ডট (...) লেখা হয়েছে সেখানে কিছু একটা নিয়ম মেনে লিখতে হবে, যখন ব্যবহার করা হবে তখন নিয়মটা বলে দেওয়া হবে।

প্রথম প্রোগ্রাম

সি (C) কিংবা সি প্লাস প্লাস (C++) ভাষায় সবার প্রথম কোন প্রোগ্রামটি লিখতে হবে সেটি অলিখিতভাবে ঠিক করা আছে। কেউ তার বাইরে যায় না—যেন তার বাইরে গেলে তার ওপর

কম্পিউটারের অভিশাপ নেমে আসবে! আমরাও সেই নিয়ম মেনে তিনটি লাইন এবং দুটি ব্র্যাকেট চিহ্ন দিয়ে সেই প্রোগ্রামটি লিখে ফেলব:

```
#include<stdio.h>
int main()
{
printf("Hello World");
}
```

আমরা এই প্রোগ্রামটা কম্পাইল করার পর রান করার আগে এক নজর দেখে নেই। প্রথম লাইনটাকে বলে 'হেডার'। প্রোগ্রামের ভেতর যেসব ফাংশন ব্যবহার করা হবে সেগুলো হেডার ফাইল সরবরাহ করে, তাই শুরুতে এটা লিখে সেটা নিশ্চিত করে নিতে হবে। প্রোগ্রামে অনেক ধরনের ফাংশন ব্যবহার করা হয়, তাই প্রয়োজনবোধে অনেক ধরনের হেডার ব্যবহার করতে হয়। আমরা যেহেতু শর্টকাট প্রোগ্রামিং করছি, তাই শুধু দুইটা হেডার লাগবে, তার একটা এখনই শিখে ফেললাম!

দ্বিতীয় লাইনটি হচ্ছে, মূল প্রোগ্রামের নাম, যেহেতু এটাই আমাদের মূল প্রোগ্রাম, `main ()` এই নামটাই এভাবেই দিতে হবে এবং এভাবেই লিখতে হবে। সবসময় সব প্রোগ্রামে এটা থাকতে হয় এবং প্রোগ্রাম শুরু হয় এই নাম থেকে।

মূল প্রোগ্রামটা শুরু হয়েছে দ্বিতীয় ব্র্যাকেটের শুরু ({) দিয়ে এবং শেষ হয়েছে দ্বিতীয় ব্র্যাকেটের শেষ (}) দিয়ে—এর মাঝখানে মূল প্রোগ্রাম। আমরা যখন প্রোগ্রামের আরো একটু ভেতরে যাব তখন

সেখানেও সবসময় প্রোগ্রামের বিভিন্ন সুনির্দিষ্ট অংশ দ্বিতীয় ব্র্যাকেট দিয়ে বিভিন্নভাবে নির্দিষ্ট করে দেবো।

এবারে এই প্রোগ্রামের একমাত্র কোড—এটি মনিটরে প্রিন্ট করার ফাংশনটি (`printf`) ব্যবহার করে লেখা একটি স্টেটমেন্ট। শেষ হয়েছে সেমিকোলন দিয়ে। দেখবে প্রোগ্রামিং করার বেলায় সবসময় স্টেটমেন্ট শেষ করতে হয় সেমিকোলন (;) দিয়ে। যদি দিতে ভুলে যাও কম্পাইলার তোমাকে চোখে আঙুল দিয়ে মনে করিয়ে দেবে! দ্বিতীয় ব্র্যাকেট দিয়ে আবদ্ধ জায়গায় ডাবল কোটেশনের (`"..."`) ভেতর যা লেখা হয় `printf` ফাংশনটি সেটাই মনিটরে প্রিন্ট করে দেয়।

এবারে আমরা আমাদের প্রথম প্রোগ্রামটাকে কম্পাইল করব। যদি সবকিছু ঠিকঠাকভাবে লেখা হয়ে থাকে তাহলে চোখের পলকে কম্পাইল হয়ে যাবে। যদি লিখতে গিয়ে ভুল-ত্রুটি হয়ে থাকে তাহলে কম্পাইলার সেটা দেখিয়ে দেবে। কম্পাইল করার পর একটা জুতসই নাম দিয়ে প্রোগ্রামটি সেভ করব তারপর রান করব। রান করা হলে মনিটরে লেখা হবে:

```
Hello World
```

নিচে আরো কিছু লেখা থাকতে পারে, আমরা সেটা নিয়ে মাথা ঘামাব না।

তুমি যদি সত্যি মনিটরে `Hello World` লিখে থাকতে পারো তাহলে তোমাকে অভিনন্দন, কারণ তুমি প্রোগ্রামিংয়ের নব্বই ভাগ করে ফেলেছো। এখন শুধু সেটাকে আরো একটু বিস্তৃত করা!

আরো একটু প্রথম প্রোগ্রাম

আমরা যেহেতু একটা প্রোগ্রাম লিখেই ফেলেছি, এটাকে আরো একটুখানি নাড়াচাড়া করে দেখি। যেমন: `printf` ফাংশনের ভেতর `Hello World` শব্দটার মাঝখানে `Beautiful` কথাটা ঢুকিয়ে দিই। তাহলে প্রোগ্রামটা রান করা হলে মনিটরে লেখা হবে— `Hello Beautiful World.`

এবারে `printf` ব্যবহার করে স্টেটমেন্টের নিচে আরো দুটো একই ধরনের স্টেটমেন্ট লিখি, একটাতে ডাবল কোটেশনের ভেতর লেখা থাকবে, `Hello Universe` অন্যটার ভেতর লেখা থাকবে `Hello Everyone`, অর্থাৎ পুরো প্রোগ্রামটা দেখাবে এরকম:

```
#include<stdio.h>
int main()
{
printf("Hello Beautiful World");
printf("Hello Universe");
printf("Hello Everyone");
}
```

এবারে এই প্রোগ্রামটা রান করে দেখি, আমরা নিশ্চয়ই ভাবছি মনিটরে লেখা হবে:

```
Hello Beautiful World
Hello Universe
Hello Everyone
```

কিন্তু মনিটরে তাকিয়ে দেখো, সেখানে লেখা আছে—

Hello Beautiful WorldHello universeHello Everyone

বোকা প্রোগ্রামটি একটার পর একটি লিখে রেখেছে! দুটো বাক্যের মাঝখানে একটু space পর্যন্ত দেয়নি! কী জ্বালা!

আসলে প্রোগ্রামটিকে বোকা বলে লাভ নেই, তুমি যেটা করতে বলেছো সে সেটাই করেছে। ভিন্ন ভিন্ন লাইনে লিখতে চাইলে সেভাবে স্টেটমেন্টটা লিখতে হবে। একটা লাইন লেখা শেষ হলে পরের লাইনে যেতে হলে `\n` লিখতে হয়। কাজেই তোমার প্রোগ্রামটা হতে হবে এ রকম:

```
#include<stdio.h>
int main()
{
printf("Hello Beautiful World\n");
printf("Hello Universe\n");
printf("Hello Everyone\n");
}
```

এবারে এটা রান করো, দেখবে তোমার প্রোগ্রামটি একটার নিচের লাইনে আরেকটা লিখেছে। ডাবল কোটেশন যেখানে শেষ হয়েছে তার ঠিক আগে `\n` লিখলে প্রোগ্রামটি পরের লাইনে চলে যায়। দুইবার লিখলে দুই লাইন নিচে, তিনবার লিখলে তিন লাইন। তাহলে তোমাদের আরো একটি জিনিস শেখা হলো!

নিজে করো: আমরা বলেছি `printf` ফাংশনে ডাবল কোটেশনের ভেতরে যেটাই লেখা হয় সেটাই মনিটরে দেখা যাবে। কথাটা সত্যি কিনা নানা কিছু লিখে সেটা পরীক্ষা করে দেখো। পরীক্ষা করা শেষ হলে তোমার প্রোগ্রামে লেখাগুলোর মাঝখানে %d %f কিংবা %c লিখে সেগুলো মনিটরে দেখানোর চেষ্টা করে দেখো তো কী হয়!

সবকিছুই যদি লেখা সম্ভব হয় তাহলে এগুলো লেখা যাচ্ছে না কেন? সমস্যা কোথায়?

ধৈর্য ধরো। একটু পরেই বিষয়টা তোমাদের বলছি।

হিসাব-নিকাশ

কম্পিউটারের নাম দেখেই বোঝা যাচ্ছে এটা আসলে তৈরি হয়েছিল কম্পিউট (compute) বা হিসাব-নিকাশ করার জন্য, গণিত করার জন্য। অথচ এখন মানুষ এটা ভুলেই গেছে—এটা দিয়ে সিনেমা দেখে, গান শুনে, ফেসবুক করে, কেউ আর হিসাব-নিকাশ বা গণিত করে না। কাজেই নিজেদের মান-সম্মান রাখার জন্য আমরা একটু গণিত করব। কঠিন কোনো গণিত নয়, যোগ-বিয়োগ এবং গুণ-ভাগ।

কেমন করে প্রোগ্রাম লিখতে হয় এখন আমরা জানি। প্রথমে হেডার ফাইল, তারপর মূল প্রোগ্রামের নাম, তারপর দুটো দ্বিতীয় ব্র্যাকেটের মাঝে প্রোগ্রাম বা কোড। কাজেই একটা প্রোগ্রাম লিখে ফেলি!

```
#include <stdio.h>
int main()
{
Jog=10+4;
Biyog=10-4;
Goon=10*4;
Bhag=10/4;
}
```

এখন পর্যন্ত যেটুকু জানি, তাতে মনে হচ্ছে প্রোগ্রামটা খারাপ হয়নি! শুরুতে হেডার ফাইল, তারপর 10 এবং 4-কে যোগ, বিয়োগ, গুণ ও ভাগ করার পর ফলাফলটা লেখার জন্য চারটা নাম দিয়েছি, নাম দেখেই বোঝা যাচ্ছে কোনটা কী! এবারে প্রোগ্রামটা কম্পাইল করো, সেভ করো এবং তারপর রান করো।

কম্পাইল করতে গিয়ে নিশ্চয়ই ধাক্কা খেয়েছো, কম্পাইলারটি রেগেমেগে বলছে সে Jog, Biyog, Goon এবং Bhag—এই ভেরিয়েবলগুলো সে চিনে না! বাংলা শব্দ ইংরেজিতে লিখেছি বলে সে বিরক্ত হয়নি, বিরক্ত হয়েছে অন্য কারণে! প্রোগ্রাম করার সময় ভেরিয়েবলগুলোর নাম আগে "ঘোষণা" করে দিতে হয়! শুধু ঘোষণা করলেই হয় না, ঘোষণা করার সময় বলে দিতে হয় যে এই ভেরিয়েবলগুলোর ভেতরে কী ধরনের তথ্য রাখা হবে।

আমরা যেহেতু সংখ্যা নিয়ে কাজ করছি, তাই আপাতত ভেরিয়েবলগুলো হবে সংখ্যা। শুধু সংখ্যা বলে দিলেই হবে না, সংখ্যারও রকমফের আছে, সেটাও বলে দিতে হবে। যেমন: তোমাকে যদি জিজ্ঞেস করি তোমরা কয় ভাই-বোন। তুমি সবসময় পূর্ণসংখ্যায় উত্তর দেবে, কখনোই বলবে না 3.25 জন (তিনজন বড় এবং একজন খুব ছোট ভাই কিংবা বোন থাকলেও!)। এ বছর তোমার স্কুল কতদিন খোলা ছিল এই প্রশ্নের উত্তরও তোমাকে পূর্ণসংখ্যায় উত্তর দিতে হবে। এই ধরনের সংখ্যাকে বলে পূর্ণসংখ্যা (integer)।

আবার অনেক সংখ্যা আছে যেগুলো পূর্ণসংখ্যা হতেই হবে এমন কোনো বাধ্যবাধকতা নেই। কেউ যদি তোমাকে জিজ্ঞেস করে, আজ সারাদিনে তুমি কয় গ্লাস পানি খেয়েছো? উত্তরে তুমি বলতেই পারো, সাড়ে সাত গ্লাস (7.5), কেউ অবাক হবে না। বাজার করতে গিয়ে একটা ইলিশ মাছ ওজন করে দেখা গেল সেটার ওজন 1.45 কেজি—এটাও খুবই স্বাভাবিক ব্যাপার। এই ধরনের সংখ্যাকে বলে বাস্তব সংখ্যা (real)।

কাজেই প্রোগ্রামিং করার সময় যদি সংখ্যা নিয়ে হিসেব করার জন্য ভেরিয়েবল ঘোষণা দিতে হয় তাহলে বলে দিতে হবে এটা কি পূর্ণসংখ্যা (integer) নাকি বাস্তব সংখ্যা (real)। পদ্ধতিটা খুবই সহজ—পূর্ণসংখ্যা হলে int শব্দটি লিখে তারপর ভেরিয়েবলগুলোর নাম লিখে দিতে হবে। বাস্তব সংখ্যা হলে float শব্দটি লিখে তারপর ভেরিয়েবলগুলোর নাম লিখে দিতে হবে। ইচ্ছে করলে আলাদাভাবে, ইচ্ছে করলে একসাথে। ভেরিয়েবলগুলোর নাম ব্যবহার করার আগে প্রোগ্রামের যেকোনো জায়গায় সেটাকে ঘোষণা দিতে হবে।

ধরা যাক আমরা পূর্ণসংখ্যা হিসেবে যোগ, বিয়োগ, গুণ ও ভাগ করতে চাই। তাহলে এবারে প্রোগ্রামটা আবার লিখে ফেলি:

```
#include <stdio.h>
int main()
{
int Jog, Biyog, Goon, Bhag;
Jog=10+4;
Biyog=10-4;
Goon=10*4;
```

```
Bhag=10/4;
}
```

কীভাবে ভেরিয়েবলগুলো ঘোষণা করা হয়েছে ভালোভাবে দেখে নাও। তারপর কম্পাইল করো। টাইপ করতে ভুল করলে কম্পাইলার তোমাকে বলে দেবে। ঠিক করে নাও। ঠিকভাবে কম্পাইল হয়ে থাকলে, সেভ করো এবং রান করো। বিশ্বাস করো আর নাই করো, প্রোগ্রামটি যোগ, বিয়োগ, গুণ এবং ভাগ করে ফেলেছে!

মনিটরে কী দেখাচ্ছে? কিছুই না! অবাক হওয়ার কিছু নাই, তুমি তাকে কিছু দেখাতে বলোনি। কাজেই প্রোগ্রামটা ঠিকভাবে যোগ, বিয়োগ, গুণ এবং ভাগ করেছে কিনা দেখার জন্য আমাদের ভেরিয়েবলগুলোর মানগুলো দেখাতে হবে। আমরা নিশ্চয়ই অনুমান করতে পারছি `printf` ব্যবহার করে মানগুলোর প্রিন্ট করতে হবে। সেটা করার জন্য সবশেষে আমাদের এই স্টেটমেন্টটি যোগ করতে হবে:

```
printf("%d %d %d %d\n",Jog, Biyog, Goon, Bhag);
```

এবারে কিছুক্ষণ এই `printf` স্টেটমেন্টটার দিকে তাকিয়ে থেকে নিজেই বোঝার চেষ্টা করো এখানে কী করার চেষ্টা করা হয়েছে। প্রথম প্রোগ্রামে আমরা যখন `printf` ফাংশনটি ব্যবহার করেছি তখন শুধু দুটো কোটেশন মার্কের মাঝে কিছু লেখা ছিল। এবারে কোটেশন মার্কের শেষে একটি কমা দিয়ে `Jog, Biyog, Goon, Bhag` আমাদের এই চারটে ভেরিয়েবল পরপর লেখা হয়েছে। বোঝাই যাচ্ছে এটা লেখা হয়েছে এদের মান লেখার জন্য

অর্থাৎ এখানে যে ভেরিয়েবল লেখা হবে তার মান মনিটরে লেখা হবে। মানগুলো কীভাবে লেখা হবে সেটা কোটেশন মার্কের ভেতর নির্দিষ্ট করে দেয়া হয়েছে। যেহেতু চারটি ভেরিয়েবল পূর্ণসংখ্যা বা `integer` তাই দুই কোটেশন মার্কের ভেতর চারবার `%d` লেখা হয়েছে। একেকটা ভেরিয়েবলের জন্য একেকবার।

এবারে পুরো প্রোগ্রামটা হবে এ রকম:

```
#include <stdio.h>
int main()
{
int Jog, Biyog, Goon, Bhag;
Jog=10+4;
Biyog=10-4;
Goon=10*4;
Bhag=10/4;
printf("%d %d %d %d\n",Jog, Biyog, Goon, Bhag);
}
```

প্রোগ্রাম লেখার পর কম্পাইল করো। ভুল-ত্রুটি হয়ে থাকলে ঠিক করো। সেভ করো তারপর রান করো। মনিটরে উত্তর লিখে দেবে উত্তরটা হবে এ রকম:

```
14 6 40 2
```

যোগ, বিয়োগ, গুণ ঠিক আছে—ভাগফল ঠিক নেই, ঠিক হবে সেটা নিশ্চয়ই আশা করোনি। ভাগফল পূর্ণসংখ্যা নয়

```
10/4 = 2.5
```

কাজেই তোমার প্রোগ্রাম ভাগফলের শুধু পূর্ণসংখ্যা অংশটুকু লিখে দিয়েছে। তার বুদ্ধিতে যেটুকু কুলিয়েছে সে তাই করেছে, তাকে

দোষ দিয়ে লাভ নেই! দোষ আমাদের। যদি ফলাফল পূর্ণ সংখ্যা হবে না তাহলে ভেরিয়েবলগুলো পূর্ণসংখ্যা হিসেবে ঘোষণা দেওয়া ঠিক হয়নি। এটাকে `float` হিসেবে ঘোষণা দেওয়া উচিত ছিল!

প্রোগ্রামের যেটুকু লিখেছি সেটুকু থাকুক, আমরা তার নিচে নূতন চারটি ভেরিয়েবল ঘোষণা দিই এবং নূতন চারটি স্টেটমেন্ট লিখি, তারপর নূতন আরেকটা `printf` স্টেটমেন্ট লিখি। তাহলে প্রোগ্রামটা হবে এ রকম:

```
#include <stdio.h>
int main()
{
int Jog, Biyog, Goon, Bhag;
Jog=10+4;
Biyog=10-4;
Goon=10*4;
Bhag=10/4;
float jog, biyog, goon, bhag;
jog=10.+4.;
biyog=10.-4.;
goon=10.*4.;
bhag=10./4.;
printf("%d %d %d %d\n",Jog, Biyog, Goon, Bhag);
printf("%f %f %f %f\n",jog, biyog, goon, bhag);
}
```

যদি তুমি সবকিছু ঠিকভাবে করে থাকো, তাহলে এই প্রোগ্রামটা রান করলে মনিটরে লেখা হবে:

14 6 40 2

14.000000 6.000000 40.000000 2.500000

দশমিকের পর একেবারে 6 ঘর পর্যন্ত দেখতে একটু বাড়াবাড়ি মনে হচ্ছে কিন্তু উত্তরটি যে সঠিক তাতে সন্দেহ নেই! (এখন যেহেতু

আমরা প্রোগ্রামিং শেখার পর্যায়ে আছি, আপাতত বাড়াবাড়িটুকু নিয়ে মাথা না ঘামালাম, একটু সহ্য করে যাই!)

`printf` ফাংশনটি দিয়ে আরো অনেক কিছু করা যায়, সবকিছু একবারে শেখানোর চেষ্টা করছি না। যখন যেটুকু দরকার হবে তখন সেটা বলা যাবে।

নিজে করো: এর আগেরবার নিজে করো নিয়ে কী সমস্যা হয়েছিল, তখন সেটা ব্যাখ্যা করা হয়নি। এতক্ষণে নিজেই সেটা বুঝে ফেলার কথা। তোমার `printf` ফাংশনের ভেতর নানা কথা লিখে একটু পরীক্ষা-নিরীক্ষা করো। এটা হচ্ছে প্রোগ্রামিং শেখার আসল টেকনিক, যখনই কিছু একটা নিয়ে প্রশ্ন থাকে তখনই ছোট একটা প্রোগ্রাম লিখে সেটা পরীক্ষা করে দেখো।

অকার্যকর লাইন

আগের প্রোগ্রামে আমরা ভুল এবং শুদ্ধ দুটিই রেখে দিয়েছিলাম। এত কষ্ট করে টাইপ করেছি তাই মুছতে মন চাইছিল না! সেটা নিয়ে বিচলিত হবার কিছু নেই, আমরা চাইলে না মুছেও লাইনগুলোকে অকার্যকর করে রেখে দিতে পারি! একটা লাইনের সামনে `//` লিখলে সেই লাইনটা অকার্যকর হয়ে যায়। লাইনের মাঝখানে `//` লিখলে লাইনে এর পরের অংশটুকু অকার্যকর হয়ে যায়। তাই যদি কোনো মন্তব্য লিখতে হয় তাহলে লাইনের শেষে `//` লিখে তারপরে যা ইচ্ছা লিখে রাখা যায়!

একটি-দুইটি লাইন অকার্যকর করে ফেলার জন্য // লেখা ঠিক আছে কিন্তু যদি একটা বড় অংশ অকার্যকর করতে চাই তাহলে ধরে ধরে সব লাইনের আগে // লেখা একটু ঝামেলার ব্যাপার। তখন শুরুতে /* লিখতে হয় এবং যেটুকু অকার্যকর করতে চাও তার শেষে */ লিখতে হয়। // কিংবা /*...*/ সরিয়ে নিলে লাইনটি কিংবা লাইনগুলো আবার কার্যকর হয়ে যায়।

এই নিয়ম ব্যবহার করে আমরা আগের প্রোগ্রামটির শুধু শুদ্ধ অংশটুকু কার্যকর করে ফেলি, তারপর কম্পাইল এবং রান করে দেখি!

```
#include <stdio.h>
int main()
{
/*
int Jog, Biyog, Goon, Bhag;
float jog, biyog, goon, bhag;
Jog=10+4;
Biyog=10-4;
Goon=10*4;
Bhag=10/4;
*/
float jog, biyog, goon, bhag;
jog=10.+4.;
biyog=10.-4.;
goon=10.*4.;
bhag=10./4.; //Now it will divide properly
//printf("%d %d %d %d\n",Jog, Biyog, Goon, Bhag);
printf("%f %f %f %f\n",jog, biyog, goon, bhag);
}
```

নিজে করো: আমরা কী কী শিখব, তার একটা তালিকা দিয়েছিলাম। তার ভেতর কতগুলো শেখা হয়েছে? কতগুলো বাকি আছে?

প্রথম সত্যিকারের প্রোগ্রাম

আমরা এখন পর্যন্ত যেটুকু শিখেছি সেটুকু দিয়েই সত্যিকারের একটা গাণিতিক সমাধানের প্রোগ্রাম লিখতে পারি। সেটাই করা যাক— আমরা দ্বিঘাত সমীকরণের (quadratic equation) সমাধান বের করব। যারা দ্বিঘাত সমীকরণের বিষয়টা জানো না, তাদের জন্য আগে একটুখানি বলে নিই। নিচের সমীকরণটি দেখো:

$$2x^2 + x - 6 = 0$$

x-এর সব মানের জন্য এটি সত্যি নয়, দুটি নির্দিষ্ট মানের জন্য এটি সত্যি। (যেহেতু দ্বিঘাত সমীকরণ, তাই x-এর দুটি মানের জন্য এর মান শূন্য। তিন মাত্রার সমীকরণ হলে তিনটি মানের জন্য এর মান শূন্য হতো, চার মাত্রার হলে চারটি মানের জন্য এর মান শূন্য হতো ইত্যাদি।)

সমীকরণটি x-এর কোন কোন মানের জন্য এর মান শূন্য হবে, সেটা সমীকরণ একটু অন্যভাবে লিখলেই বোঝা যাবে। একই সমীকরণটি একটু নাড়াচাড়া করে আমরা এভাবে লিখতে পারি:

$$(2x - 3)(x + 2) = 0$$

এটার দিকে তাকালেই আমরা বলতে পারব যদি $2x = 3$ (বা $x = 1.5$) হয় কিংবা $x = -2$ হয় তাহলে সমীকরণটির মান শূন্য

হবে। যারা দ্বিঘাত সমীকরণের সমাধান কীভাবে করতে হয় সেটা জানো তারা বলবে, সমীকরণটি যদি হয়:

$$ax^2 + bx + c = 0$$

(আমাদের উদাহরণটির জন্য $a = 2,\ b = 1,\ c = -6$) তাহলে তার সমাধান হচ্ছে:

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

কিংবা

$$x = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

তোমরা ইচ্ছা করলে a, b এবং c-এর মান বসিয়ে পরীক্ষা করে দেখতে পারো—প্রথমটি দেবে $x = 1.5$ হয় এবং দ্বিতীয়টি দেবে $x = -2$, ঠিক যে রকম হওয়ার কথা। আমরা এবারে এই সমীকরণটি সমাধান করার জন্য একটা প্রোগ্রাম লিখে ফেলব। সব মিলিয়ে এখানে 5টি ভেরিয়েবল লাগবে, `a`, `b`, `c` এবং দুটি সমাধান `root1` এবং `root2`, সবই বাস্তব। কাজেই প্রোগ্রামটা হবে এ রকম:

```
#include<stdio.h>
#include<math.h>
int main()
{
float a,b,c,root1,root2;
a=2;
b=1;
```

```
c=-6;
root1=(-b+sqrt(b*b-4*a*c))/(2*a);
root2=(-b-sqrt(b*b-4*a*c))/(2*a);
printf("root1= %7.2f  root2= %7.2f\n",
root1,root2);
}
```

প্রোগ্রামটি মনোযোগ দিয়ে দেখো, এখানে তোমাদের না বোঝার মতো কিছু নেই। `root1` এবং `root2`-এর জন্য গাণিতিক স্টেটমেন্টটি দেখতে প্রায় এলজেবরার লাইনের মতো। শুধু `sqrt` ফাংশনটি নূতন, এটা দিয়ে বর্গমূল বের করা হয়। আমরা যেহেতু বর্গমূল বের করছি, তাই গাণিতিক কাজকর্ম করার জন্য নূতন একটা হেডার (`#include <math.h>`) দিতে হয়েছে। `printf` ফাংশনে `%f` না লিখে % এবং `f`-এর মাঝখানে 7.2 বসিয়ে `%7.2f` লেখা হয়েছে। এর কারণে সংখ্যাগুলো লিখতে সাত ঘর নেবে এবং দশমিকের পর দুই ঘর লিখবে। আগে প্রয়োজন ছাড়াই বাড়াবাড়ি করে দশমিকের পর ছয়ঘর লিখত, এবারে লিখবে না।

এবারে প্রোগ্রামটা কম্পাইল করো, সেভ করো তারপর রান করো। দেখবে মনিটরে লেখা হবে:

```
root1=  1.50                    root2=  -2.00
```

ঠিক যে রকম হওয়ার কথা। অভিনন্দন, তুমি প্রথম বার সত্যিকার কাজের জন্য সত্যিকার একটা প্রোগ্রাম লিখেছো।

যেহেতু প্রোগ্রামটা লেখা হয়েছে তাই এটাকে নিয়ে একটু পরীক্ষা-নিরীক্ষা করা দরকার। `a, b` এবং `c`-এর নানা ধরনের

মান দিয়ে দেখো কী সমাধান পাওয়া যায়। b=0 বসালে একটু বিশেষ সমাধান পাওয়া যায়। ঠিক সে রকম a=1, b=2, c=1 আবার a=1, b=-2, c=1 বসালেও কিছু বিশেষ সমাধান আসে। কারণটা চিন্তা করে বের করো । প্রোগ্রামিং করতে বসেছি তাই গণিত নিয়ে চিন্তা-ভাবনা করা যাবে না—কে বলেছে?

সরাসরি

আমি তোমাদের বলেছি দ্বিঘাত সমীকরণের এই প্রোগ্রামটা বিভিন্ন a, b এবং c-এর মান দিয়ে পরীক্ষা করতে। কিন্তু কাজটা তো খুব সহজ না, কারণ সেটা করার জন্য প্রত্যেকবার প্রোগ্রামের ভেতর a, b এবং c-এর জন্য নূতন সংখ্যা টাইপ করতে হয়, তারপর সেটা আবার কম্পাইল করতে হয়, তারপর রান করতে হয়। কাজটা অনেক সহজ হতো, যদি প্রোগ্রাম রান করার সময় প্রোগ্রাম আমাদের কাছে a, b এবং c-এর মান চাইত এবং আমরা সেটা টাইপ করে দিয়ে দিতে পারতাম।

কাজটা খুবই সহজ। আমরা printf দিয়ে কীভাবে সংখ্যা লিখতে হয় সেটা শিখেছি। ঠিক সে রকম একটা ফাংশন ব্যবহার করতে হবে, যেটা সংখ্যাটা গ্রহণ করবে। এই ফাংশনটাকে বলে scanf, এটার ব্যবহারের সাথে printf-এর ব্যবহারের মিল আছে। ধরা যাক আমাদের প্রোগ্রামের একটা পূর্ণসংখ্যার ভেরিয়েবল আছে x, আমরা সেটার জন্য একটা সংখ্যা “নেব”, তাহলে আমরা লিখব:

```
scanf ("%d", &x);
```

ডাবল কোটেশনের ভেতর %d অংশটুকু বোঝা সহজ, printf-এর বেলায় সেটা আমরা দেখেছি। কিন্তু পরের অংশে শুধু x না লিখে &x লিখতে হচ্ছে, সেটা যথেষ্ট বিচিত্র! আমি নিজে সবসময় & লিখতে ভুলে যাই, তারপর বহু সময় নষ্ট হয় কেন আমার প্রোগ্রাম কাজ করছে না সেটা বের করার জন্য। কাজেই আমরা ঝগড়াঝাঁটি না করে এটা মেনে নিই এবং কষ্ট করে এভাবেই লিখি। যদি দুটো ভেরিয়েবল x এবং y-এর মান নিতে হয় তাহলে লিখব:

```
scanf ("%d %d", &x, &y);
```

ঠিক সে রকম যদি একটা ভেরিয়েবল a বাস্তব সংখ্যার হয় তাহলে লিখব:

```
scanf ("%f", &a);
```

বাস্তব সংখ্যার দুটো ভেরিয়েবল a এবং b হলে

```
scanf ("%f %f", &a, &b);
```

এবং তিনটি হলে

```
scanf ("%f %f %f", &a, &b &c);
```

আমরা আমাদের প্রোগ্রামে a, b এবং c-এর জন্য তিনটি মান নিতে চাই, তাই আমাদের এই স্টেটমেন্টটা লিখতে হবে। এবার তাহলে প্রোগ্রামটা আমরা লিখে ফেলি।

```
#include<stdio.h>
#include<math.h>
int main()
```

```
{
float a,b,c,root1,root2;
printf("Type values of a, b and c:");
scanf("%f %f %f", &a, &b, &c);
root1=(-b+sqrt(b*b-4*a*c))/(2*a);
root2=(-b-sqrt(b*b-4*a*c))/(2*a);
printf("root1= %7.2f root2= %7.2f\n",
root1,root2);
}
```

প্রোগ্রামটার দিকে তাকিয়ে দেখো। `scanf` ব্যবহার করে `a`, `b` এবং `c`-এর মান নেয়ার ঠিক আগে আগে একটা `printf` ব্যবহার করে মানগুলো টাইপ করার কথা বলে দিয়েছি। তাহলে প্রোগ্রাম রান করার সময় তুমি বুঝতে পারবে ঠিক কখন `a`, `b` এবং `c`-এর জন্য মান টাইপ করতে হবে। এবার প্রোগ্রামটা কম্পাইল করো, সেভ করো তারপর রান করো, দেখবে মনিটরে লেখা হবে।

type values of a b c:

তুমি তিনটি সংখ্যা টাইপ করবে। ধরে নিই তুমি আমাদের আগের সংখ্যা তিনটিই টাইপ করতে চাও। তাহলে মাঝে space দিয়ে দিয়ে তুমি লিখবে:

type values of a b c: 2 1 -6

এবারে প্রোগ্রামটা রান করলে সেটা তোমাকে ফলাফল দিয়ে দেবে:

root1= 1.5 root2= -2

তুমি এখন যতবার খুশি প্রোগ্রামটা রান করে যত ইচ্ছা তত ভিন্ন ভিন্ন a, b এবং c-এর মান দিতে পারবে। আগের বারের মতো প্রত্যেকবার প্রোগ্রাম পরিবর্তন করে সেটা আর নূতন করে কম্পাইল করতে হবে না।

যদি এবং তাহলে

দ্বিঘাত সমীকরণের সমাধান বের করার সময় a, b এবং c-এর বিভিন্ন মান দিয়ে পরীক্ষা করার সময় কখনো কি হাবিজাবি উত্তর পেয়েছো? যদি পেয়ে না থাকো তাহলে a=2, b=1 এবং c=6 বসিয়ে দেখো কী হয়, তুমি হাবিজাবি উত্তর পাবে। তার কারণ তাহলে b^2-4ac-এর মান হবে -47 এবং এই -47-এর বর্গমূল নিতে গিয়ে তোমার প্রোগ্রামের দফা রফা হয়ে যাবে। নেগেটিভ সংখ্যার বর্গমূল ইমাজিনারি (imaginary) সংখ্যা হয় কিন্তু আমাদের কম্পাইলার ইমাজিনারি বা কমপ্লেক্স (complex) সংখ্যায় হিসাব-নিকাশ করতে পারে না।

একটা প্রোগ্রাম মাঝে মাঝে হাবিজাবি সংখ্যা উত্তর দেবে, সেটা ভালো কথা নয়। আমরা কি এই সমস্যাটার একটা ভদ্র সমাধান করতে পারি?

অবশ্যই পারি, কম্পিউটার প্রোগ্রামে এ রকম সমস্যার সমাধান সবসময় হাতের কাছে থাকে। সেটা হচ্ছে b^2-4ac সংখ্যাটি প্রত্যেকবার পরীক্ষা করে দেখা, যদি এটা নেগেটিভ হয় তাহলে আমরা হিসাব-নিকাশ বন্ধ করে দেবো, কিছু প্রিন্টও করব না। যদি

b^2-4ac পজিটিভ কিংবা শূন্য, হয় শুধু তাহলে আমরা হিসাব-নিকাশ করব।

এটা করার জন্য আমরা এই প্রথম বার `if` স্টেটমেন্ট ব্যবহার করব। এটা খুবই গুরুত্বপূর্ণ একটা স্টেটমেন্ট—নানাভাবে এটা ব্যবহার করা যায়। আমরা ব্যবহার করব সবচেয়ে সহজ উপায়ে। আমরা লিখব:

```
If ((b*b-4*a*c)>0)
```

{ b^2-4ac শূন্যের সমান কিংবা পজিটিভ হলে প্রোগ্রামের যে অংশটুকু করতে চাই সেটা এখানে লেখা হবে }

ঠিক প্রয়োজন নেই, কিন্তু ইচ্ছা করলে নিচের লাইনটাও যুক্ত করে দিতে পারি:

```
If ((b*b-4*a*c)<0)
```

{ b^2-4ac নেগেটিভ হলে আমরা যে হিসেব চালিয়ে যেতে পারছি না সে জন্য দুঃখ প্রকাশ করে কোনো কথা এখানে লিখে দেবো!}

বুঝতেই পারছো `If` স্টেটমেন্টের ভেতর একটা শর্ত বা লজিক্যাল condition থাকে, সেই শর্ত বা condition সত্যি হলে একটা কিছু করে, সত্যি না হলে করে না। তাহলে আমরা আমাদের প্রোগ্রামটা এবারে নূতন করে লিখে ফেলি:

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

int main()
{
float a,b,c,root1,root2;
```

```
printf("a b c : ");
scanf("%f %f %f", &a, &b, &c);

if((b*b-4*a*c)>=0){
root1=(-b+sqrt(b*b-4*a*c))/(2*a);
root2=(-b-sqrt(b*b-4*a*c))/(2*a);
printf("root1= %7.2f  root2= %7.2f\n",
root1,root2);
                    }
if((b*b-4*a*c)<0)printf("sorry! cannot
calculate :(\n");
}
```

এবারে প্রোগ্রামটা কম্পাইল করো, সেভ করো এবং রান করো। যদি সবকিছু ঠিকভাবে হয়ে থাকে তাহলে যে exe ফাইলটা তৈরি হবে সেটি হবে একটা স্বয়ংসম্পূর্ণ প্রোগ্রাম। যে নামে প্রোগ্রামিং ফাইল লিখেছিলে সেই নামে এটা সেভ হয়েছে। তুমি কি exe ফাইলটা কপি করে তোমার বন্ধুদের পাঠাতে চাও, যেন তারাও চাইলে দ্বিঘাত সমীকরণের সমাধান বের করতে পারে? তাহলে তোমাদের ছোট একটা কাজ করতে হবে, সবার শেষে লিখতে হবে:

```
scanf("%f", &a);
```

অন্য কাউকে exe ফাইলটা পাঠালে কেন এই সম্পূর্ণ অপ্রয়োজনীয় কথাটা নিচে লিখে কম্পাইল করে দিতে হবে, সেটা আমি বলে দিচ্ছি না, তোমরা নিজেরা বের করে নাও। একবার বাক্যটা লিখে exe ফাইল তৈরি করো, আরেকবার এটা ছাড়া তৈরি করো । দুটোই আলাদা আলাদ ভাবে রান করো । তাহলে নিজেরাই বুঝে ফেলবে।

নিজে করো: তোমার প্রোগ্রামটা রান করার সময় প্রথমে কিছু একটা লিখে দেওয়া যায়। একেবারে শুরুতে printf ব্যবহার করে যা ইচ্ছা তা-ই লিখতে পারবে। নিচে যেটা লেখা আছে সেটা লিখতে পারবে? কয়টা printf স্টেটমেন্ট লাগবে?

```
**************************************************
To find roots of ax^2 + bx + c = 0
Just type the values of a, b and c
Developed by: ... ...  ...
**************************************************
```

নিচে পাশের ফাঁকা জায়গায় তোমার নাম লিখে দেবে!

নিজে করো: আমাদের তালিকা দ্রুত শেষ হয়ে আসছে। তোমরা যদি এই বইটা প্রোগ্রামিং না করে, প্রোগ্রামগুলো ঘাঁটাঘাঁটি না করে শুধু গল্প বইয়ের মতো পড়তে আসছো তাহলে আসলে পড়ার দরকার নেই! তোমার সময় নষ্ট হচ্ছে।

চূড়ান্ত শর্টকাট

আগেই বলে রাখি, এই ছোট অধ্যায়টাতে তোমাদের যেটা শেখানো হবে, সেটা তোমাকে ছেলেমানুষী প্রোগ্রামার থেকে আসল প্রোগ্রামারে পাল্টে দেবে।

তোমরা এরই মাঝে জেনে গেছো কম্পিউটারে হিসাব-নিকাশ করতে হলে ভেরিয়েবলের দরকার হয় এবং সেটা শুরুতে ঘোষণা

দিয়ে নিতে হয়। (আপাতত আমরা পূর্ণসংখ্যা আর বাস্তব সংখ্যা ধরনের ভেরিয়েবল শিখেছি, পরে আরও একটা শিখব।)

ধরা যাক, তুমি একটা কম্পিউটার প্রোগ্রাম লিখতে চাও, যেখানে 347টা ভেরিয়েবল দরকার, তাহলে তুমি নিশ্চয়ই হতাশ হয়ে যাবে। 347টা ভিন্ন ভিন্ন ভেরিয়েবলের নাম চিন্তা করে বের করে সেগুলো একটা একটা করে লিখতে লিখতে তোমার দিন পার হয়ে যাবে। শুধু তাই না, কোন ভেরিয়েবলটা কিসের জন্য সেটা লিখে রাখার জন্য বাজার থেকে একটা নোট বই কিনে আনতে হবে।

এবারে আমি তোমাকে চোখের পলকে 347টা ভিন্ন ভিন্ন ভেরিয়েবল ঘোষণা দেওয়ার টেকনিক শিখিয়ে দেবো। ধরা যাক তুমি পূর্ণসংখ্যা ধরনের ভেরিয়েবল চাও। তুমি লিখবে:

```
int x[347];
```

তুমি বিশ্বাস করো আর না-ই করো, তুমি কিন্তু 347টা পূর্ণসংখ্যা ধরনের ভেরিয়েবল ঘোষণা দিয়ে দিয়েছো। সেগুলো হচ্ছে:

```
x[0], x[1], x[2], x[3]...x[344],
x[345], x[346]
```

কম্পিউটার যেহেতু 0 থেকে গোনা শুরু করে, তাই `x[0]` থেকে শুরু হয়েছে এবং `x[347]`-তে শেষ না হয়ে `x[346]`-এ শেষ হয়েছে। সব মিলিয়ে 347টি। যদি 0 দিয়ে লেবেল করতে অস্বস্তি হয়, তুমি `348`টির ঘোষণা দিতে পারতে এবং `x[0]`-টি ব্যবহার

না করে `x[1]` থেকে শুরু করে `x[347]` পর্যন্ত যেতে পারতে। কিংবা আরো উদারভাবে লিখবে:

```
int x[350];
```

কিছু বেশি থাকুক, সমস্যা কী? (আসলে একটু বেশি লিখে রাখা সবসময়ই বুদ্ধিমানের কাজ!)

তুমি যদি বাস্তব সংখ্যা ধরনের 987টা ভেরিয়েবল চাও, তাহলে লিখতে পারো:

```
float y[1000];
```

আবার `y[0]` থেকে `y[999]` পর্যন্ত 1000টি ভেরিয়েবলের ঘোষণা দেওয়া হয়েছে, দরকার থেকে একটু বেশি, সমস্যা নেই!

কম্পিউটারের একটা বদ অভ্যাস হচ্ছে, কোনো ভেরিয়েবলের ঘোষণা দিয়ে যদি সেখানে একটা মান রাখা না হয় তাহলে সেই ভেরিয়েবলের ভেতর যেকোনো উল্টাপাল্টা সংখ্যা ঢুকে যেতে পারে। ধরা যাক সেজন্য আমরা আমাদের `x[1]` থেকে `x[347]` পর্যন্ত `347` টা ভেরিয়েবলের ভেতর সুনির্দিষ্টভাবে শূন্য (0) সংখ্যাটি ঢুকিয়ে পরিষ্কার করে রাখতে চাই। তাহলে আমরা কী করব? ভোরবেলা ঘুম থেকে উঠে

```
x[1]=0;
x[2]=0;
x[3]=0;
```

এভাবে একেবারে সর্বশেষ `x[347]=0;` পর্যন্ত লিখব? তার প্রয়োজন নেই। এ রকম করার জন্য আমাদের আরও মজার মজার

শর্টকাট টেকনিক আছে। সবচেয়ে সোজাটা হচ্ছে `for` লুপ (loop), আমাদের লিখতে হবে:

```
int x[350], i;
for (i=1; i<=347;i++){x[i]=0;}
```

প্রথম লাইনটাতে `i` ভেরিয়েবল ঘোষণা দিয়েছি। দ্বিতীয় লাইনে `for` লুপের ভেতর সেটা ব্যবহার করেছি। কী লেখা আছে দেখলেই অনুমান করা যায়। বলা হয়েছে `i=1` থেকে শুরু করে `i=347` (এবং তার কম প্রত্যেকটা সংখ্যা) পর্যন্ত যাও। `i++` এর অর্থ হচ্ছে `i`-কে এক এক করে বাড়াও। স্টেটমেন্টের পর দুটি সেকেন্ড ব্র্যাকেটের ভেতর যা কিছু প্রোগ্রাম লেখা হবে তার পুরোটা `for` লুপ একবার একবার করে 347 বার করবে। কম্পিউটারের ক্লান্তি নেই, অভিযোগ নেই। আমাদের এই উদাহরণে একটা মাত্র লাইন `x[i]=0;` করতে বলা হয়েছে, সেখানে `x[1]` থেকে শুরু করে `x[347]` পর্যন্ত সব ক'টিকে এক এক করে `0` করে দেওয়া হয়েছে!

`x[350]` কিংবা `y[1000]` এভাবে একসাথে অনেক ভেরিয়েবল ঘোষণা দেওয়ার এই পদ্ধতির একটা নাম আছে। সেটা হচ্ছে এরে (Array)। এখানে `x[350]` কিংবা `y[1000]` হচ্ছে দুটো Array.

সত্যিকারের প্রোগ্রামিং

এর আগে যা শিখেছি তার সাথে যোগ হয়েছে `array` এবং `for` লুপ। এখন আমরা সত্যিকারের প্রোগ্রামিং করতে পারব। তার আগে

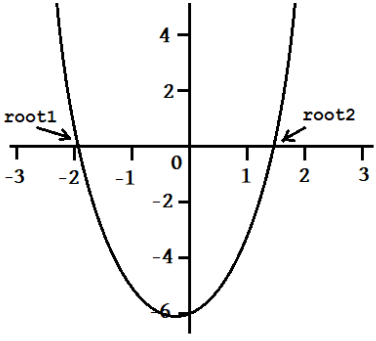
সত্যিকারের প্রোগ্রামে এরে (array) আর `for` লুপ ব্যবহার করা শিখতে হবে।

আমরা এর আগে দ্বিঘাত সমীকরণের সমাধান বের করেছি। এবারে সেই দ্বিঘাত সমীকরণের সমাধানই আবার বের করা হবে কিন্তু একটু অন্যভাবে। এবারে দ্বিঘাত সমীকরণটাকে আমরা লিখব এভাবে:

$$y = ax^2 + bx + c$$

`a, b` এবং `c` জানা থাকলে `x`-এর একটা মান দেওয়া হলে `y`-এর এর মান পাওয়া যাবে। আমরা `x`-এর বিভিন্ন মান দিয়ে দেখব কোন মানের জন্য `y`-এর মান শূন্য, সেটাই হবে আমাদের সমাধান। যদি আমরা `x-y`-এর গ্রাফ আঁকতে পারতাম গ্রাফটা ছবি 1-এর মতো দেখাতো।

দেখতেই পাচ্ছ `x`-এর মান যখন `root1` কিংবা `root2`-এর সমান তখন `y`-এর মান হচ্ছে শূন্য। আমরা এখন গ্রাফ আঁকতে পারছি না কিন্তু বিভিন্ন `x`-এর মানের জন্য `y` নিশ্চয়ই বের করতে পারব। যদি আমরা এমন `a, b` এবং `c` ব্যবহার করি, যার জন্য `y`-এর মান এই গ্রাফের মতো, তাহলে আমরা দেখব `x` যখন কম `y` এর মান পজিটিভ, `x` বাড়তে বাড়তে যখন `root1`-এর সমান হবে



ছবি 1: $2x^2 + x - 6$ এর মান

তখন y-এর মান শূন্য। x-এর মান আরো বাড়ানো হলে y-এর মান নেগেটিভ হতে থাকবে। আরো যদি বাড়াই তখন y-এর মান একসময় উপর দিকে উঠতে থাকবে, যখন x-এর মান root2 হবে তখন y-এর মান আবার শূন্য হবে। যদি আরো বাড়াই তাহলে y-এর মান আবার পজিটিভ হয়ে বাড়তে থাকবে। অর্থাৎ আমরা যদি কম x থেকে বেশি x-এর জন্য y-এর মান বের করি তাহলে তাকে পজিটিভ থেকে নিগেটিভ এবং নেগেটিভ থেকে আবার পজিটিভ হতে থাকব। x-এর যে দুটো মানের জন্য y-এর মান সাইন পরিবর্তন করবে সেটাই হচ্ছে আমাদের সমীকরণের সমাধান।

এবারে আমরা প্রোগ্রামটা লিখি:

```
#include<stdio.h>
int main()
{
float a,b,c,xmin,xmax,delx,x[101],y[101];
int i;
```

```
printf("type a b c xmin xmax: ");
scanf("%f %f %f %f %f", &a,&b,&c,&xmin,
&xmax);
delx=(xmax-xmin)/100;
for(i=1;i<=100;i++)
{
x[i]=xmin+delx*i;
y[i]=a*x[i]*x[i]+b*x[i]+c;
printf("%7.4f  %7.4f\n", x[i], y[i]);
}
}
```

যেহেতু একটি নয়, দুই-দুইটি নূতন ধারণা প্রথমবার ব্যবহার করা হচ্ছে তাই প্রোগ্রামটার প্রায় প্রত্যেকটা লাইন নিচে যথেষ্ট বিস্তৃতভাবে ব্যাখ্যা করা হলো:

```
#include<stdio.h>
int main()
```

সব প্রোগ্রামেই হেডার ফাইল তারপর মূল প্রোগ্রামের নাম থাকে, ব্যাখ্যা করার কিছু নেই।

```
{
```

দ্বিতীয় ব্র্যাকেট দিয়ে প্রোগ্রামের শুরু।

```
float a,b,c,xmin,xmax,delx,x[101],y[101];
```

a, b এবং c হচ্ছে ax^2+bx+c এর তিনটি সহগ। xmin হচ্ছে x-এর সর্বনিম্ন যে মান থেকে y-এর মান বের করা শুরু হবে, xmax হচ্ছে সর্বোচ্চ যে মানের জন্য যাওয়া হবে। আগেই বলছি, a=2, b=1 এবং c=-6 এর জন্য xmin হবে -2.5 এর কাছাকাছি, xmax হবে 2 এর কাছাকাছি। delx হচ্ছে x এর ছোট ছোট ধাপ (step) যে ধাপ দিয়ে xmin থেকে xmax পর্যন্ত যাওয়া হবে। x[101] হচ্ছে xmin থেকে xmax পর্যন্ত x-এর যে

100টি মানের জন্য y বের করা হবে। y[101] হচ্ছে y-এর সেই মানগুলো। সব ক'টি বাস্তব সংখ্যা।

```
int i;
```

পূর্ণসংখ্যা i যেটা ব্যবহার করে for লুপটি চালানো হবে।

```
printf("type a b c xmin xmax: ");
```

এই লাইনটি জানিয়ে দেবে যে এখন a, b, c, xmin এবং xmax-এর মান দিতে হবে। লক্ষ করো \n লেখা হয়নি, কারণ পরের লাইনে যাওয়ার প্রয়োজন নেই।

```
scanf("%f %f %f %f %f", &a,&b,&c,&xmin,
&xmax);
```

a, b, c, xmin এবং xmax এই পাঁচটি ভেরিয়েবলের মান গ্রহণ করব। ভেরিয়েবলগুলোর মান গ্রহণ করার জন্য %f লেখা হয়েছে।

```
delx=(xmax-xmin)/100;
```

xmax থেকে xmin পর্যন্ত x-এর ব্যাপ্তিটুকু (range) 100 দিয়ে ভাগ করা হলো। delx হচ্ছে ছোট ধাপটি (step), যে ধাপ দিয়ে x বাড়ানো হবে।

```
for(i=1;i<=100;i++)
```

i=0 থেকে শুরু করে 1,2,3—এভাবে 100 পর্যন্ত বাড়ানো হবে।

```
{
```

for লুপ প্রোগ্রামের যে অংশটুকু নিয়ন্ত্রণ করবে তার শুরু।

```
x[i]=xmin+delx*i;
```

x-এর মান বের করা শুরু হলে, প্রতিবার i-এর মান 1 করে বাড়বে, x-এর মান delx পরিমাণ বাড়বে। xmin থেকে বেড়ে বেড়ে xmax-এ গিয়ে শেষ হবে।

```
y[i]=a*x[i]*x[i]+b*x[i]+c;
```

x-এর মানের জন্য y-এর মান।

```
printf("%7.4f  %7.4f\n", x[i], y[i]);
```

প্রতি x-এর মানের জন্য বের করা y-এর মান প্রিন্ট করা হলো। সব মিলিয়ে 101 বার।

```
}
```

for লুপ এর সমাপ্তি এখানে।

```
}
```

মূল প্রোগ্রামের সমাপ্তি এখানে।

এটা ঠিকভাবে টাইপ করে সেভ করো, কম্পাইল করো তারপর রান করো। a b c-এর জন্য 2 1 -6 এবং xmin ও xmax এর জন্য -2.5 2.0 টাইপ করো। দেখবে x-এর মান -2.5000 থেকে শুরু করে একটু একটু করে বেড়ে 2.0000 পর্যন্ত যাবে। প্রতিটি x-এর মানের জন্য একটি করে y-এর মান রয়েছে। যেটি পজিটিভ দিয়ে শুরু করে জিরো ক্রসিং করে নিগেটিভ হয়েছে আবার নিগেটিভ থেকে জিরো ক্রসিং করে পজিটিভ হয়েছে। যেখানে জিরো অতিক্রম করেছে সেখানেই রয়েছে সমাধান। একশটি মানের ভেতর দশটি দশটি করে টেবিল 1-এ দেখানো হলো, যেখানে y-এর মান পজিটিভ থেকে নিগেটিভ হয়েছে এবং নেগেটিভ থেকে পজিটিভ

টেবিল 1

...	
-2.1850	1.3634
-2.1400	1.0192
-2.0950	0.6831
-2.0500	0.3550
-2.0050	**0.0350**
-1.9600	**-0.2768**
-1.9150	-0.5806
-1.8700	-0.8762
-1.8250	-1.1638
-1.7800	-1.4432
...	
...	
1.2800	-1.4432
1.3250	-1.1637
1.3700	-0.8762
1.4150	-0.5805
1.4600	**-0.2768**
1.5050	**0.0351**
1.5500	0.3550
1.5950	0.6831
1.6400	1.0192
1.6850	1.3635
...	

হয়েছে। দুই জায়গাতেই বোঝার সুবিধার জন্য বোল্ড (bold) করে দেয়া হয়েছে।

এবারে জিরো ক্রসিংটা আরো খুটিয়ে দেখা সম্ভব। a, b এবং c-এর মান ঠিক রেখে xmin এবং xmax-এর মানের জন্য যথাক্রমে -2.0050 এবং -1.9150 দেওয়া যায় (root1 আরেকটু নিখুঁতভাবে বের হবে)। একইভাবে root2 আরেকটু নিখুঁতভাবে বের করার জন্য a, b এবং c-এর মান ঠিক রেখে xmin এবং xmax-এর মানের জন্য 1.4600 এবং 1.5050 দেওয়া যায়। চেষ্টা করে দেখো, সবসময় হুবহু ঠিক সমাধান পাবে না কিন্তু xmin এবং xmax-এর ব্যাপ্তি (range) কমিয়ে কমিয়ে যত ইচ্ছা আসল সমাধানের কাছাকাছি যেতে পারবে।

মনে আছে যখন আমরা a, b এবং c-এর মান 2, 1, +6 দিয়েছিলাম, তখন আমরা হাবি-জাবি উত্তর পেয়েছিলাম? এবারে এই মান দিয়ে দেখ, দেখবে y এর মান জিরো ক্রসিং করছে না—অর্থাৎ x-এর বাস্তব মানের জন্য কোনো সমাধান নেই।

দ্বিঘাত সমীকরণের সমাধান আছে, স্কুলের ছেলেমেয়েরাও সেই সমাধানটি জানে। তৃতীয় মাত্রার সমীকরণেরও সমাধান আছে, সেটা সমাধান করতে কমপ্লেক্স নম্বর দরকার হয়—কিন্তু সমাধানটা অপূর্ব! কেন এটা শেখানো হয় না আমি জানি না। চার মাত্রার সমীকরণেরও সমাধান আছে, সেটাও শেখানো হয় না, এডভান্স এলজেবরার বইয়ে পাওয়া যায়—পঞ্চম মাত্রা কিংবা তার বেশি মাত্রার সমাধান নেই, শুধু যে সমাধান নেই তা নয়, কেউ সমাধান করতেও পারবে না গণিতবিদেরা সেটাও প্রমাণ করে রেখেছেন!

কিন্তু মজার ব্যাপার কি জানো? তুমি এই মাত্র যে প্রোগ্রামটা লিখেছো সেখানে $y=ax^2+bx+c$-এর জায়গায়:

$y=ax^3+bx^2+cx+d$ কিংবা
$y=ax^4+bx^3+cx^2+dx+e$

লিখে এই সমীকরণগুলোর সমাধান বের করতে পারবে। সেটা করার জন্য a, b, c, d, e—এইগুলোর মান দিতে হবে, তারপর দেখতে হবে x-এর কোন কোন মানের জন্য এটা শূন্যকে অতিক্রম (zero crossing) করছে, যেখানে যেখানে করেছে সেখানেই সমীকরণের সমাধান।

মজার ব্যাপার হচ্ছে, শুধু চার মাত্রা নয়, চাইলে পাঁচ কিংবা তার চাইতে বেশি মাত্রার সমীকরণের সমাধানও বের করে ফেলতে পারবে।

নিজে করো: ত্রিঘাত একটা সমীকরণ সমাধান করার জন্য একটি প্রোগ্রাম লিখো।

নিজে করো: আমরা যদি আমাদের প্রোগ্রামের `printf` ফাংশনটাকে তার লুপ থেকে বের করে এনে আলাদা একটা লুপে ঢুকিয়ে এভাবে প্রিন্ট করি, তাহলে সেটা দেখতে কেমন দেখাবে?

```
#include<stdio.h>
int main()
{
float a,b,c,xmin,xmax,delx,x[101],y[101];
int i;
printf("type a b c xmin xmax: ");
scanf("%f %f %f %f %f", &a,&b,&c,&xmin,
&xmax);
delx=(xmax-xmin)/100;
for(i=1;i<=100;i++)
{
x[i]=xmin+delx*i;
y[i]=a*x[i]*x[i]+b*x[i]+c;
}
for(i=1;i<=50;i++)printf("%7.4f %7.4f    %7.4f
%7.4f\n", x[i], y[i], x[50+i], y[50+i]);
}
```

সংখ্যা নয়, বর্ণ

এতক্ষণ আমরা সংখ্যা দিয়ে হিসাব-নিকাশ করেছি। এবারে একটু অন্য কিছু নিয়ে কাজ করা যাক। সংখ্যার বদলে এবারে আমরা বর্ণ বা character নিয়ে কাজ করব।

তোমাদের নিশ্চয়ই মনে আছে আমরা পূর্ণসংখ্যা আর বাস্তব সংখ্যা—এই ধরনের ডেটা টাইপ নিয়ে কাজ করেছি। এখন নূতন একটা ডেটা টাইপ নিয়ে কথা বলা যাক—সেটা হচ্ছে বর্ণ বা character.

পূর্ণসংখ্যা এবং বাস্তব সংখ্যার ভেরিয়েবলের আগে আমরা `int` আর `float` লিখে ঘোষণা দিয়েছি। বর্ণের জন্য আমরা ভেরিয়েবলের ঘোষণা দিব `char` লিখে। পূর্ণসংখ্যা এবং বাস্তব সংখ্যা গ্রহণ করার জন্য `scanf`-এর ভেতর এবং প্রিন্ট করার জন্য `printf`-এর ভেতর `%d` এবং `%f` ব্যবহার করেছি। `character` টাইপের ভেরিয়েবলের জন্য লিখব `%c`, ব্যস আমাদের সবকিছু জানা হয়ে গেল। এখন শুধু একটুখানি প্র্যাকটিস। প্রথমে একটা ছোট প্রোগ্রাম লিখে পরীক্ষা করে নিই আমরা যেটা বলেছি সেটা সত্যি:

```
#include <stdio.h>
int main()
{
char a,b,c,d,e;
a='H';
b='i';
c=' ';
d=':';
e=')';
printf("%c%c%c%c%c\n",a,b,c,d,e);
}
```

এই প্রোগ্রামটার মাঝে নূতন বিষয়গুলো হচ্ছে `char` লিখে ভেরিয়েবলগুলোর ঘোষণা দেওয়া। তারপর `a=H; b=i;` না লিখে

a=`H'; b=`i'; এভাবে সিংগেল কোটেশনের ভেতর অক্ষরগুলো লেখা এবং printf ফাংশনে পাঁচটা অক্ষরের জন্য পাঁচটা %c লেখার সময় তাদের মাঝে কোনো ফাঁকা জায়গা না রাখা। (মনে আছে, %d এবং %f-এর বেলায় ফাঁকা জায়গা না রাখলে এক সংখ্যার সাথে পরের সংখ্যা মিলেমিশে একাকার হয়ে যেত?) আরেকটা বিষয় লক্ষ্য করা যায়, তোমাদের যদি মনে হয় c ভেরিয়েবলে কোনো মান দেওয়া হয়নি, সেটা কিন্তু সত্যি নয়, c ভেরিয়েবলে space ঢোকানো হয়েছে, কম্পিউটারে লেখার সময় সেটাও একটা বর্ণের মতো ব্যবহার করতে হয়।

এবারে সেভ করো, কম্পাইল করো এবং প্রিন্ট করো । মনিটরে লেখা হবে:

Hi :)

এই প্রোগ্রামটাতে প্রোগ্রামের ভেতর লিখে ভেরিয়েবলে বর্ণগুলি ঢুকিয়ে রাখা হয়েছে। আমরা ইচ্ছা করলে scanf ব্যবহার করে প্রোগ্রাম রান করার সময় একেকবার একেকটা বর্ণ ঢোকাতে পারি। তাহলে প্রোগ্রামটা হবে এ রকম:

```
#include <stdio.h>
int main()
{
char a,b,c,d,e;
scanf("%c%c%c%c%c",&a,&b,&c,&d,&e);
printf("%c%c%c%c%c\n",a,b,c,d,e);
}
```

তুমি যে পাঁচটা বর্ণ লিখবে সেই পাঁচটি বর্ণকে মনিটরে দেখতে পাবে। তুমি ভেরিয়েবলের সংখ্যা বাড়াও তাহলে বর্ণের সংখ্যা বাড়াতে পারবে। চেষ্টা করো নিজের নামটি লিখতে। ভেরিয়েবলের সংখ্যা কতো বাড়াতে হবে? দশ? বিশ? ভেরিয়েবলের সংখ্যা কত পর্যন্ত বাড়ানো বাস্তবসম্মত? এক হাজার? দশ হাজার?

ভেরিয়েবলের সংখ্যা বাড়ানোর আগে চট করে এক লাইনের একটা প্রোগ্রাম লিখে রান করে ফেলি।

এক লাইনের প্রোগ্রাম

নিচের প্রোগ্রামটির দিকে তাকাও, দেখে নিশ্চয়ই তোমার মনে হচ্ছে ভুল কিছু লেখা হয়েছে।

```
#include<stdio.h>
main ()
{
int i;
for(i=32;i<=255;i++)printf("%d %c\n\n", i, i);
}
```

আমরা নিশ্চিতভাবে জানি, এখানে `i` হচ্ছে পূর্ণসংখ্যা, রীতিমতো `int` হিসেবে ঘোষণা দেয়া হয়েছে, কিন্তু `printf`-এর ভেতর আমি সেটাকে `%c (char)` হিসেবে প্রিন্ট করতে বলেছি। এটা কি সম্ভব? পূর্ণসংখ্যাকে কীভাবে বর্ণ হিসেবে প্রিন্ট করবে? কম্পাইলার নিশ্চয়ই আটকে দেবে। কিন্তু টাইপ করতে গিয়ে ভুল না করে থাকলে কম্পাইলার আটকাবে না, তুমি রান করতে পারবে। আমরা `i`-কে দুবার প্রিন্ট করতে বলেছি, একবার `int` হিসেবে

আরেকবার `char` হিসেবে। আমার ধারণা, তুমি অবাক হয়ে দেখবে `%d`-এর জন্য `i`-কে ঠিক ঠিক পূর্ণ সংখ্যা হিসাবে প্রিন্ট করেছে এবং `%c`-এর জন্য `32` থেকে `127` পর্যন্ত ইংরেজি বর্ণগুলো প্রিন্ট শেষ করে নানা ধরনের চিহ্ন প্রিন্ট করতে শুরু করেছে।

তার কারণ কম্পিউটারে বর্ণগুলোর প্রত্যেকটার জন্য আসলে একটা সংখ্যা নির্দিষ্ট করে দেওয়া থাকে যেটাকে বলে অ্যাসকি (ascii)। লিখিত বর্ণ `32` থেকে শুরু হয়ে `127`-এ শেষ হয়েছে। এর পরেরগুলোকে বলে (higher ascii) যেখানে কিছু চিহ্ন আছে। চিহ্নগুলো চাইলে তুমি খুঁটিয়ে খুঁটিয়ে দেখতে পার। `220`-এর চিহ্নটা একটু ভালো করে দেখে রাখো, আমরা এটা পরে ব্যবহার করব।

নিজে করো: সংখ্যাকে `%c` হিসেবে প্রিন্ট করা হলে সেটা অ্যাসকি (ascii) এবং হায়ার অ্যাসকি (higher ascii)-এর বর্ণগুলো প্রিন্ট করে। উল্টোটাও কি সত্যি? বর্ণকে পূর্ণসংখ্যা হিসেবে প্রিন্ট করতে চাইলে কি হবে? আগের প্রোগ্রামের প্রিন্ট স্টেটমেন্টটা `//` দিয়ে অকেজো করে (কিংবা অকেজো না করেই) নিচের স্টেটমেন্টটা লিখে রান করিয়ে দেখো। খুশি?

```
printf("%d %d %d %d %d\n\n", 'A', 'B', 'C', 'D',
'E');
```

নিজে করো: হায়ার অ্যাসকির (higher ascii) এই চিহ্নগুলো ব্যবহার করে কিছু একটা আঁকতে পারবে?

দশ হাজারের বেশি ভেরিয়েবল

হ্যাঁ, আমরা দশ হাজারের বেশি character ধরনের ভেরিয়েবল দিয়ে একটা মজার বিষয় করার চেষ্টা করতে পারি।

প্রথমে c[101][101] নাম দিয়ে দশ হাজারের বেশি (101x101=10201) ভেরিয়েবল ঘোষণা দেবো। তারপর for লুপ দিয়ে সেখানে কিছু তথ্য ঢোকাব। বুঝতেই পারছো এই এরেটি (array) দ্বিমাত্রিক, কাজেই ভেরিয়েবলগুলো

```
c[0][0], c[0][1]  ... c[0][100]
c[1][0], c[1][1]  ... c[1][100]
c[2][0], c[2][1]   ... c[2][100]
...
...
c[99][0], c[99][1]...c[99][100]
c[100][0], c[100][1]...c[100][100]
```

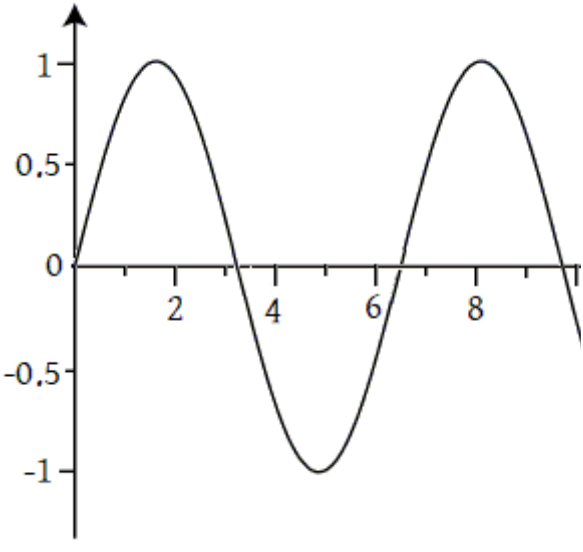
এভাবে বিন্যস্ত থাকবে।

প্রথমে c[101][101] এই এরেটির প্রত্যেকটিতে '.' ফুল স্টপ বা দশমিক বিন্দু ঢুকিয়ে দেই। কাজটি পানির মতো সোজা, এক লাইনে করে ফেলা যাবে। শুধু লিখতে হবে:

```
for(i=0;i<=100;i++)for(j=0;j<=100;j++)
c[i][j]='.';
```

কীভাবে এটা কাজ করে নিশ্চয়ই বুঝতে পারছো। প্রথম লুপে যখন i=0 হয় তখন দ্বিতীয় লুপে j=0 থেকে শুরু করে 100 পর্যন্ত যাবে। এভাবে i বাড়তে বাড়তে 100 পর্যন্ত যাবে।

এরপর আমরা y=sinx বের করব। x-এর মান যদি 0 থেকে 10 হয় (রেডিয়ানে, ডিগ্রিতে নয়) তাহলে এটা ছবি 2-এর মতো দেখাবে।



ছবি 2: রেডিয়ানে 1 থকে 10 -এর জন্য sin -এর মান।

আমরা i-কে 0.1 দিয়ে গুণ দিয়ে একটা for লুপ (loop) চালাব, i-এর মান 0 থেকে 100 পর্যন্ত যাবে। যে মানটা পাব সেটা হবে x । বুঝতেই পারছো x-এর মান 0 থেকে 10 পর্যন্ত যাবে।

এই প্রত্যেকটি x-এর জন্য আমরা একটি করে y বা sinx বের করব। অর্থাৎ x-এর মান বের করার জন্য আমরা লিখতে পারি:

```
for(i=0;i<=100;i++)x[i]=0.1*i;
```

আমরা সবাই জানি sin-এর মান সর্বনিম্ন -1 সর্বোচ্চ +1, কোনো একটা বিশেষ কারণে (কারণটা একটু পরেই বুঝতে পারবে) এই মানটিকে আমি রূপান্তর করতে চাই 0 থেকে 100 পর্যন্ত। এটা এভাবে করা সম্ভব:

```
for(i=0;i<=100;i++)y[i]=50+50.*sin(x[i]);
```

আমাদের প্রোগ্রামে y[i] বাস্তব সংখ্যা। যদি আমি সেটাকে লিখি (int) y[i]তাহলে সেটা হবে পূর্ণ সংখ্যা, যার মান হবে 1 থেকে 100-এর ভেতরে। (আমরা আরো একটা নূতন জিনিষ শিখলাম, কীভাবে float ধরনের সংখ্যাকে int ধরনের সংখ্যায় রূপান্তর করা যায়!) কেন এটা করতে চাই এক্ষুনি দেখবে। এখন আমি লিখব:

```
for(i=0;i<=100;i++)c[i][(int)y[i]]='X';
```

যার অর্থ c[i][j] প্রত্যেকটা ছিল '.'। এখন তার কোনো কোনোটি হয়ে গেছে 'X'। কোন কোনটি 'X' হয়েছে সেটা দেখার সবচেয়ে সহজ উপায় হচ্ছে সব ক'টি প্রিন্ট করে ফেলা। সেটা করতে হলে i এবং j দিয়ে দুটো for লুপ চালিয়ে c[i][j] প্রিন্ট করতে হবে—তাহলে আমরা দেখব অসংখ্য দশমিক বিন্দুর '.' মাঝে কোনটি কোনটি 'X'!

এবারে তাহলে গুছিয়ে পুরো প্রোগ্রামটা লিখে ফেলি:

```
#include<stdio.h>
#include<math.h>
int main()
{
char c[101][101];
float x[101],y[101];
int i,j;

for(i=0;i<=100;i++)for(j=0;j<=100;j++)c[i][j]=
'.';
for(i=0;i<=100;i++)x[i]=0.1*i;
for(i=0;i<=100;i++)y[i]=50+50.*sin(x[i]);
for(i=0;i<=100;i++)c[i][(int)y[i]]='X';

for(i=0;i<=100;i++)
    {
{for(j=0;j<=100;j++)printf("%c",c[i][j]);}prin
tf("\n");
    }
}
```

যদি প্রোগ্রামটি তুমি ঠিকভাবে টাইপ করে ঠিকভাবে কম্পাইল করে রান করে থাকো তাহলে তুমি একটা চমকের জন্য প্রস্তুত হও। কারণ তুমি দেখবে `sin`-এর একটি প্লট! `c[i][j]`-এর সবক'টি দশমিক বিন্দুর মাঝে '`X`' হয়েছে শুধু `sin`-এর মানগুলো। (স্বীকার করছি এটা মোটেও সত্যিকারের প্লট নয়, আসল প্রোগ্রামারেরা বলবে প্লটের নামে এটা একটা তামাশা। কিন্তু শর্টকাট প্রোগ্রামিংয়ে এর চাইতে বেশি আশা করে লাভ নেই। তোমাদের যাদের আগ্রহ আছে তারা সত্যিকারের প্রোগ্রামিংয়ের বই পড়ে সত্যিকারের প্লটিং প্যাকেজ ব্যবহার করে সত্যিকারের প্লট করা শিখে নিতে পারবে।)

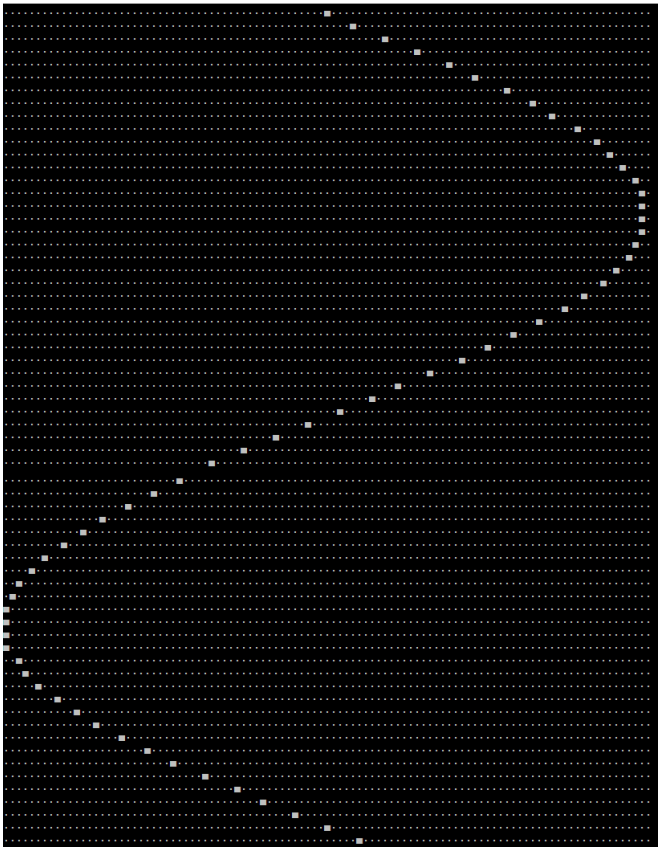
ছোটো একটা কাজ বাকি আছে। মনে আছে অ্যাসকি এবং হায়ার অ্যাসকি নিয়ে কথা বলার সময়ে আমি তোমাদের হায়ার অ্যাসকির 220 নম্বর চিহ্নটা আলাদা ভাবে দেখে রাখতে বলেছিলাম? এখন সেটা ব্যবহার করব। c[i][j]-এর ভেতর sin-এর মানকে 'x' হিসেবে রূপান্তর না করে আমরা সেখানে লিখব 220, অর্থাৎ লাইনটা হবে এ রকম:

```
for(i=0;i<=100;i++)c[i][(int)y[i]]=220;
```

তাহলে গ্রাফটা আরো অনেক ভালো দেখাবে। বিশ্বাস না হলে সেটা লিখে কম্পাইল করে রান করে দেখো (ছবি 3)!

তোমাদের ছোট আরো একটা কাজ বাকি আছে, সেটা শেষ করার জন্য প্রোগ্রামটির দিক আরেকবার তাকাতে হবে। তোমরা নিশ্চয়ই দেখছো এখানে পরপর পাঁচটা for লুপ রয়েছে। শেষ লুপটিতে যেহেতু একটা লুপের ভেতর আরেকটা লুপ আছে সেটাকে বিবেচনা না করলাম, তাহলে থাকল চারটা লুপ। এই চারটা for লুপই একই ধরনের কাজ করছে, তাহলে চারবার চারটা for লুপ না লিখে একটা লুপ লিখে সেটাকে দিয়ে কাজ চালিয়ে নিই না কেন? সত্যিকারের প্রোগ্রামাররা তাই করত, কিন্তু যেহেতু আমরা শর্টকাট প্রোগ্রামিং করছি তাই বোঝানোর সুবিধার জন্য আলাদা আলাদা করে এক জিনিস চার বার লেখা হয়েছে! কম্পিউটার এত তাড়াতাড়ি সব কিছু করে ফেলে যে যেটুকু সময়ের অপচয় হচ্ছে সেটা জন্মেও তুমি টের পাবে না!

নিজে করো: চারটা লুপকে একটা লুপের ভেতর এনে প্রোগ্রামটা লিখে দেখতে চাও? সত্যিকারের প্রোগ্রামারের মতো?



ছবি 3: আমাদের প্রোগ্রামে sin-এর প্লট। বাম থেকে ডানে না গিয়ে উপর থেকে নিচে প্লট করা হয়েছে। এখানে পুরোটা না দেখিয়ে আংশিক দেখানো হয়েছে।

যদি এটা তোমার পছন্দ হয়ে থাকে তাহলে sin এর জায়গায় cos লিখো তাহলে cos-এর প্লট পেয়ে যাবে।

sin -এর জন্য হায়ার অ্যাসকিতে 220 এবং cos-এর জন্য 240 ব্যবহার করে তুমি একই প্লটে একসাথে sin এবং cos প্লট করতে পারবে।

নিজে কর: একই গ্রাফে sin এবং cos প্লট করো।

নিজে কর: তোমরা কি y=0 লাইনটি আঁকতে পারবে? (c[i][50]কে '|' মান করে দিয়ে দেখো কী হয়!)

কাট-পেস্ট প্রোগ্রামিং

বেশ খানিকক্ষণ খাঁটি প্রোগ্রামিং করা হয়েছে, এবারে একটু কাট-পেস্ট প্রোগ্রামিং করা যাক:

(1) প্রথমে এতক্ষণ যতগুলো প্রোগ্রাম লেখা হয়েছে (এবং বিভিন্ন নামে সেভ করা হয়েছে) তার কয়েকটি কপি করে একটা ফাইলে নূতন একটা নাম দিয়ে সেভ করো।

(2) প্রত্যেকটা ফাইলের শুরুতে হেডার ছিল, সব মিলিয়ে দুই রকম হেডার। একেবারে প্রথমটার ওপর এই দুই রকম হেডারগুলো এনে অন্যগুলো ডিলিট করে দাও।

(3) এখানে প্রত্যেকটার নাম main এবং main-এর আগে লেখা int, প্রত্যেকটার বেলায় int শব্দটির বদলে লেখো void

এবং main-এর বদলে নূতন একটা নাম দাও। ধরা যাক প্রথম প্রোগ্রামটার নাম দিয়েছে HelloWorld, যে প্রোগ্রামটি যেকোনো দ্বিঘাত সমীকরণ সমাধান করে তার নাম দাও Soln2ndOrder, যে প্রোগ্রামটা জিরো ক্রসিং দিয়ে দ্বিঘাত সমীকরণের সমাধান করে সেটার নাম দিয়ে ZeroCrossing আর যে প্রোগামটা sin-কে প্লট করে দিয়েছে সেটার নাম PlotSin দেওয়া হলো।

(4) বুঝতেই পারছো আমরা এই চারটা প্রোগ্রাম রান করতে চাই কিন্তু একটা main প্রোগ্রাম না থাকলে সেটা কখনোই হবে না, কারণ তোমার কম্পাইলার সবসময় main থেকে শুরু করে। তাই সব ক'টির নিচে লিখো:

```
int main()
{
HelloWorld();
Soln2ndOrder();
ZeroCrossing();
PlotSin();
}
```

সেভ করো, কম্পাইল করো, রান করো । তুমি যদি সবগুলো কাজ ঠিকভাবে করে থাকো তাহলে দেখবে তোমার main প্রোগ্রামটি তার সমস্ত ক্ষমতা ব্যবহার করে প্রথমে HelloWorld তারপর Soln2ndOrder, তারপর ZeroCrossing এবং সবশেষে PlotSin রান করবে।

তার মানে হচ্ছে আমরা আগে যখন prinf, scanf ব্যবহার করেছি সেগুলো ছিল ফাংশন। তোমার main প্রোগ্রামে এখন এই চারটা প্রোগ্রাম হচ্ছে চারটা ফাংশন। তোমরা দেখেছো printf-এর পর দুটো প্রথম ব্র্যাকেটের ভেতর ভেরিয়েবল নাম, কীভাবে প্রকাশ করে এসব তথ্য দিতে হয়। তোমার ফাংশনগুলোতে এখন কিছু দিতে হচ্ছে না, তাই দুটো ফাস্ট ব্র্যাকেটের মাঝে কিছু নেই, পুরোপুরি খালি, এভাবে লিখেছো ( )। তুমি চাইলেই main প্রোগ্রাম থেকে সেখানে কিছু একটা দিতে পারতে, প্রয়োজন হয়নি বলে দাওনি। ইচ্ছে করলে একটা ফাংশন লেখা যায়, যেখানে কোনো সংখ্যা বা কোনো বর্ণ বা অন্য কিছু দিতে পারবে।

আমাদের main প্রোগ্রামটি একেবারে বেশি সাদামাটা এবং খুব বেশি কাজের না। আমরা যখন প্রোগ্রাম শিখতে শুরু করেছি তখন HelloWorld লিখেছি। যখন বেশ কিছু শিখে ফেলেছি তখন PlotSin লিখেছি। একই সাথে খুব সোজা HelloWorld এবং মোটামুটি ভদ্র PlotSin রান করানোর সম্ভাবনা কম। কখনো একটি, কখনো অন্যটি রান করতে চাইতে পারি। সেটা করা এমন কিছু কঠিন নয়, এখন তোমরা নিজেই করতে পারবে। কিন্তু এই বেলা আমি করে দিই। তুমি main প্রোগ্রামটি এভাবে লিখো:

```
int main()
{
int tag;
printf("Type HelloWorld =1 Soln2ndOrder = 2
ZeroCrossing = 3 PlotSin = 4: ");
scanf("%d", &tag);
if(tag==1)HelloWorld();
```

```
if(tag==2)Soln2ndOrder();
if(tag==3)ZeroCrossing();
if(tag==4)PlotSin();
}
```

লক্ষ করো `if(tag=1)` লিখলে হবে না কারণ `tag=1` এর অর্থ `tag`-এর মান `1`, আমরা সেটা বলতে চাই না। আমরা বলতে চাই "যদি `tag` এর মান হয় `1`", কাজেই লিখতে হবে `if(tag==1)`।

এবারে প্রোগ্রামটা রান করা হলে তোমার কাছে একটা সংখ্যা চাইবে, তুমি যে সংখ্যাটি লিখবে সেই অনুযায়ী একটা প্রোগ্রাম রান করবে। অর্থাৎ আমরা আমাদের যে প্রোগ্রামটা রান করার ইচ্ছা শুধু সেটা রান করতে পারব—আমাদের ইচ্ছা থাকুক বা না থাকুক জোর করে সব ক'টি দেখতে হবে না।

কিন্তু তারপরেও একটুখানি অতৃপ্তি রয়ে গেছে। হয়তো একটা নির্দিষ্ট প্রোগ্রাম আমি একবার নয়, বারবার রান করাতে চাই, সেটা কি করা সম্ভব?

যেহেতু অন্য কোথাও অতৃপ্তি রাখিনি, এখানে কেনই বা অতৃপ্তি থাকবে? `while(...)` নামে একটা ফাংশন দিয়ে এটাও সমাধান করে ফেলব। `while` অনেকটা `if`-এর মতন, `while`-এর ভেতর যে শর্ত থাকে সেটা সত্যি হলে সে এর পরে সেকেন্ড ব্র্যাকেট দিয়ে আবদ্ধ অংশটা সম্পন্ন করে, সত্যি না হলে সেই জায়গাটায় হাত দেয় না। অর্থাৎ:

while (কোনো একটা শর্ত) {শর্ত বা condition সত্যি হলে এই অংশ সম্পন্ন করবে।}

শর্ত সত্যি না হলে উপরের অংশটুকু না করেই এখানে চলে আসবে।

এবারে এটা কেমন করে ব্যবহার করা হয় সেটা দেখি। উপরের চারটা function-এর পর main প্রোগ্রামটা এখন এ রকম:

```
int main()
{
int tag;
tag=1;
while(tag!=0){
printf("Type  HelloWorld  =1  Soln2ndOrder  =  2
ZeroCrossing = 3 PlotSin = 4 Quit = 0 : ");
scanf("%d", &tag);
if(tag==1)HelloWorld();
if(tag==2)Soln2ndOrder();
if(tag==3)ZeroCrossing();
if(tag==4)PlotSin();
             }
}
```

চারটা if statement এখন while-এর আওতায় চলে এসেছে। tag!=0 কথাটির অর্থ tag-এর মান যখন শূন্য নয়।

শুরুতে tag-এর একটা মান (tag=1;) দেয়া হয়েছে, এটা শূন্য ছাড়া যেকোনো সংখ্যা হতে পারত। সে জন্য while-এর আওতায় থাকা অংশ কাজ করতে শুরু করবে। প্রতিবার পুরো অংশ শেষ করে আবার তার জায়গায় ফিরে আসবে। শুধু tag-এর মান শূন্য হলে সে বের হয়ে আসবে। প্রোগ্রামটা সেভ করো, কম্পাইল

করো তারপর রান করে দেখো, ঠিক যেভাবে কাজ করার কথা সেভাবে কাজ করছে কি না।

খাঁটি ফাংশন (প্রায়)

এতক্ষণ আমরা যে ফাংশনগুলো তৈরি করেছি, ব্যবহার করেছি, সেগুলো পুরোপুরি ফাংশনের মতো হয়নি, সবগুলো আলাদা আলাদা প্রোগ্রাম আমরা শুধু `main`-এর ভেতর থেকে সেগুলো রান করেছি, তার বেশি কিছু নয়। যদি সত্যি সত্যি function তৈরি করতে চাই তাহলে `main` প্রোগ্রাম থেকে ফাংশনের ভেতর কিছু ডেটা পাঠানো দরকার আবার দরকার হলে ফাংশন থেকে `main` প্রোগ্রামে কিছু ডেটা পাঠানো দরকার। দেখি সেটা করা যায় কি না।

প্রথমে আমরা একটা ফাংশনের ভেতর `main` প্রোগ্রাম থেকে ডাটা পাঠাব, ফাংশন সেই ডাটাগুলোকে কোনো কাজে লাগাবে। এটা করার সবচেয়ে কাজের কাজ হবে যদি আমরা একটা গ্রাফ আঁকার প্রোগ্রাম তৈরি করতে পারি। `main` প্রোগ্রাম থেকে যেটা প্লট ফাংশনে পাঠাব, `plot` সেটাই প্লট করে দেবে।

আসলে এটা প্রায় করাই আছে, আমাদের শুধু sin আঁকার প্রোগ্রামটাকে ঠান্ডা মাথায় দুই ভাগ করতে হবে। যে অংশটা গ্রাফ আঁকবে সেটা থাকবে `Plot` ফাংশনে। যে অংশটা `y(x)` বের করবে সেটা থাকবে `main` প্রোগ্রামে। তাহলে সেটা করা শুরু করি। প্রোগ্রামে যে তিনটি লাইন `main` প্রোগ্রামে যাবে, সেটা আলাদাভাবে bold করে দেখানো হলো। বাকি সবকিছু ফাংশনের প্রোগ্রামে যাবে।

```
#include<stdio.h>
#include<math.h>
int main()
{
char c[101][101];
float x[101],y[101];
int i,j;

for(i=0;i<=100;i++)for(j=0;j<=100;j++)c[i][j]='
 .';
for(i=0;i<=100;i++)x[i]=0.1*i;
for(i=0;i<=100;i++)y[i]=50+50.*sin(x[i]);
for(i=0;i<=100;i++)c[i][(int)y[i]]=220;

for(i=0;i<=100;i++)
    {
{for(j=0;j<=100;j++)printf("%c",c[i][j]);}print
 f("\n");
    }
}
```

কাজেই এবারে আমরা ফাংশন এবং main প্রোগ্রামকে ভাগ করে দিই, ফাংশনটির নূতন একটি নাম (plot) দিই:

```
#include<stdio.h>
#include<math.h>
float plot(float y[])
{
char c[101][101];
int i,j;

for(i=0;i<=100;i++)for(j=0;j<=100;j++)c[i][j]=
'.';
for(i=0;i<=100;i++)c[i][(int)y[i]]=220;
for(i=0;i<=100;i++)
    {
```

```
{for(j=0;j<=100;j++)printf("%c",c[i][j]);}prin
tf("\n");
    }
}
int main()
{
    int i;
    float x[101],y[101];
    for(i=0;i<=100;i++)x[i]=0.1*i;
    for(i=0;i<=100;i++)y[i]=50+50.*sin(x[i]);
    plot(y);
}
```

মূল প্রোগ্রামের পুরোটা দুই ভাগে ভাগ করার কারণে `main` প্রোগ্রামে আলাদা করে লিখতে হয়েছে। `plot` ফাংশনটিতে `float y[ ]` লিখে `y` এরেটির মানগুলো আনা হয়েছে। `main` প্রোগ্রামে ফাংশনটি চালানোর জন্য `plot(y);` লেখা হয়েছে। লক্ষ কর `y` একটি এরে হলেও সেখানে `y[101]` কিংবা `y[ ]` লিখতে হয়নি, শুধু `y` লিখেই কাজ চালানো হয়েছে।

এবারে প্রোগ্রামটা রান করো। দেখবে এটি sin এঁকে দেবে।

ঘষা-মাজা করা plot ফাংশন

আমরা `plot` ফাংশনটিকে আরো একটু ঘষা-মাজা করে নিতে পারি। তাহলে শুধু sin নয়, ইচ্ছা করলে অন্য যেকোনো কিছুও প্লট করতে পারব। এখানে যে যে ঘষা-মাজা করতে চাই সেগুলো হচ্ছে:

(1) y এরেটিকে (array) 50 দিয়ে গুণ করে, ইনডেক্সে 50 যোগ করে পাঠানোর প্রয়োজন নেই। হিসেবে যা আসে তা-ই পাঠানো হবে। plot ফাংশনটি বাকি কাজ করে নেবে।

(2) কোনো x-এর জন্য y-এর মান কত সেটা হিসাব করে বলে দেবে।

(3) plot ফাংশন y-এর সর্বোচ্চ এবং সর্বনিম্ন মান তোমার কাছ থেকে জেনে নেবে, তাহলে গ্রাফটা আমাদের সীমার ভেতর থাকবে।

(4) x-এর প্রতি ঘরের পাশে একটি পুর্ণসংখ্যায় সেটি কত নম্বর x সেটা বলে দেবে।

তোমরা যদি আরও ঘষা-মাজা করতে চাও করতে পার। আমার মাথায় আর কিছু আসছে না!

ফাংশনটি যে ঘষা-মাজা করা হয়েছে সেটা বুঝতে হলে নানা ধরনের ডেটা প্লট করতে দিতে হবে। সে জন্য main প্রোগ্রামটিও একটু ঘষা-মাজা করা যেতে পারে। কী করা হয়েছে সেটা বোঝা মোটেও কঠিন হবে না, কারণ এখানে নূতন কিছু করা হয়নি। আগে যেগুলো করা হচ্ছে সেটাই লেখা হয়েছে।

এটা লিখো, সেভ করো, কম্পাইল করো এবং রান করো। তবে প্রোগ্রামটি দেখে দেখে নয়। নিজে নিজে। ভেবে ভেবে। লজিকটা বুঝে থাকলে সেটি মোটেও কঠিন কিছু নয়।

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

```
float plot(float y[])
{
float maxy, miny,mult,zero;
char c[101][101];
int i,j;

printf("maximum y: ");
scanf("%f",&maxy);
printf("minimum y: ");
scanf("%f",&miny);
mult=100/(maxy-miny);
zero=mult*(-miny);

for(i=0;i<=100;i++)for(j=0;j<=100;j++)c[i][j]=
'.';
for(i=0;i<=100;i++)c[i][(int)zero]='|';
for(i=0;i<=100;i++){c[i][(int)(mult*y[i]+zero)
]=220;}
for(i=0;i<=100;i++){printf("%3d",i);

{for(j=0;j<=100;j++)printf("%c",c[i][j]);}prin
tf("\n");
                    }
}

int main()
{
float y[101];
int i;
for(i=0;i<=100;i++){y[i]=sin(.1*i);printf("%f\
n",y[i]);}
plot(y);
}
```

নিজে করো: শুধু sin প্লট না করে অন্য কোনো ফাংশন লিখে সেটা প্লট করে দেখাও।

নিজে করো: তালিকার মাঝে আর কী কী শেখা এখনও বাকি আছে?

লিখো এবং পড়ো

এতক্ষণ পর্যন্ত আমাদের নানা প্রোগ্রামে নানা ধরনের ইনপুট দিতে হয়েছে, আমরা টাইপ করে সেগুলো দিয়েছি (`scanf` সেটা নিয়েছে) আর যখনই কোনো আউটপুট দেখার প্রয়োজন হয়েছে সেটা মনিটরে দেখানো হয়েছে (আমরা `printf` দিয়ে সেগুলো মনিটরে লিখেছি)। কিন্তু যদি অনেক ইনপুট দিতে হয় এবং আউটপুটগুলো সংরক্ষণ করতে হয় তখন? তখন সবচেয়ে সহজ কোনো একটা ফাইল থেকে ইনপুট নেওয়া (`scanf`-এর বদলে আমরা ব্যবহার করব `fscanf`, যেটা প্রায় হুবহু `scanf`-এর মতো) কিংবা একটা ফাইলে ইনপুট লিখা (`printf`-এর বদলে আমরা লিখব `fprinf`, যেটা প্রায় হুবহু `printf`-এর মত)।

কোনো ফাইলে লিখতে হলে শুরুতে ফাইলটার পরিচয় (অনেকটা ফাইলের ID-এর মতো) ঘোষণা দিয়ে জানিয়ে দিতে হয়, আপাতত শুধু এই বিষয়টা তোমাদের জানতে হবে। ধরা যাক, তুমি তোমার ইনপুট নিতে চাও `myinput.txt` নামের একটা ফাইল থেকে। তাহলে প্রোগ্রামের শুরুর দিকে লিখবে:

```
FILE *f1=fopen("myinput.txt", "r");
```

এখানে শেষের "r" বোঝাচ্ছে এই ফাইলটা read করার জন্য বা এখান থেকে ইনপুট নেওয়ার জন্য। দেখাই যাচ্ছে ফাইলের নামটি ডাবল কোটেশনের ভেতর দেওয়া হয়েছে। স্টেটমেন্টের শুরুতে FILE লিখে যে *fl লেখা হয়েছে সেখানকার fl-টি খুব গুরুত্বপূর্ণ, যখনই ফাইল read করতে হবে, এই fl-টি সেখানে এভাবে উল্লেখ করতে হবে। (তুমি fl-এর বদলে অন্য যা ইচ্ছা তাই লিখতে পারো, এটা হচ্ছে ফাইলটার পরিচয় বা ID)

```
fscanf(f1,"%c",&x);
```

প্রথমে fl লিখে নেওয়া ছাড়া scanf-এর সাথে fscanf এর আর কোনো পার্থক্য নেই।

তোমরা নিশ্চয়ই অনুমান করতে পারছো কোনো ফাইলে আউটপুট লিখতে চাইলে আমরা কী ভাবে তার পরিচয় ঘোষণা করব, "r" এর বদলে "w" লিখব।

```
FILE*f2=fopen("myoutput.txt", "w");
```

এবং printf-এর জায়গায় fprintf হবে এরকম:

```
fprintf(f2,"%c",x);
```

ব্যস, তোমাদের সবকিছু শেখা হয়ে গেছে!

কাজেই তুমি তোমার আগের প্রোগ্রামগুলোতে যেখানে যেখানে printf লিখেছো সেগুলোতে printf-এর নিচে আরেকটা fprintf লিখে দাও, অবশ্যই প্রথমে ফাইলের নামটির ঘোষণা

দিয়ে, দেখবে যা কিছু মনিটরে আসছে ঠিক সেগুলো তোমার নাম দেওয়া ফাইলে লেখা হয়ে যাচ্ছে।

এবারে আমরা ফাইলে লেখা এবং ফাইল থেকে পড়া নিয়ে দুটো প্রোগ্রাম লিখি, তাহলে পুরো বিষয়টা একটু প্র্যাকটিস হয়ে যাবে।

প্রথমে একটা `test.text` ফাইল তৈরি করো অথবা কোনো জায়গা থেকে নিয়ে এসো। এর ভেতরে বাংলা (ইউনিকোড) এবং ইংরেজি দুই লেখাই থাকতে পারে। তারপর এভাবে নিচের কয়েক লাইনের প্রোগ্রামটি লিখ:

```
#include <stdio.h>
int main()
{
char x;
int i;
FILE *f1=fopen("test.txt", "r");
FILE *f2=fopen("test_copy.txt", "w");
while(fscanf(f1,"%c",&x)==1){fprintf(f2,"%c",x
);}
printf("copied successfully :)");
}
```

এখানে শুধু একটি নূতন কাজ করা হয়েছে, `fscanf` ফাংশনটি একটা `while`-এর পর বসানো হয়েছে:

```
while(fscanf(f1,"%c",&x)==1)
```

বিষয়টি বোঝার জন্য তোমাদের নূতন একটা বিষয় জানা দরকার, সেটা হচ্ছে, `fscanf` ফাংশনটি যদি ঠিকভাবে `read` করতে পারে তাহলে `1` সংখ্যাটি `return` করে। তাই `fscanf` ফাংশনটি দিয়ে প্রত্যেকটা অক্ষর পড়ার পর পরীক্ষা করে দেখা হচ্ছে

এটি ঠিকভাবে অক্ষরটি পড়তে পেরেছে কি না। যদি ঠিকভাবে পড়ে থাকে `fscanf(f1,"%c",&x)=1` হবে এবং পরের অংশটুকু করবে, অর্থাৎ তাহলে এটি `fprintf` ব্যবহার করে নির্ধারিত `test_copy.text` ফাইলটিতে সেই অক্ষরটি লিখে ফেলবে। এভাবে একটি একটি অক্ষর পড়ে এবং লিখে যখন আর পড়ার মতো কিছু থাকবে না, তখন `fscanf` আর `1 return` করবে না। প্রোগ্রামটি `while` লুপ থেকে বের হয়ে আসবে, অর্থাৎ আর কিছু ফাইলে না লিখে প্রোগ্রামটি শেষ করে দেবে।

প্রোগ্রাম শেষ হয়েছে জানানোর জন্য মনিটরে একটা বাক্য এবং `:)` চাইলে লিখতে পারো। প্রোগ্রাম সেভ করো, কম্পাইল করো এবং রান করো, তারপর তোমার ফোল্ডারে দেখো, সেখানে `test_copy.txt` নামে একটা ফাইল তৈরি হয়েছে যেটা হুবহু `test.txt` ফাইলের মত। অর্থাৎ তুমি একটা প্রোগ্রাম লিখে `txt` ফাইল কপি করা শিখে গেছো!

এই প্রোগ্রামে `fprintf` না লিখে যদি সেখানে `printf` লিখতাম, অর্থাৎ লাইনটি হতো এ রকম:

```
while(fscanf(f1,"%c",&x)==1){printf("%c",x);}
```

তাহলে `test.txt` ফাইলটি `test_copy.txt`-তে না লিখে মনিটরে দেখাতো।

যদি তোমার ফাইলে বাংলা লেখা থাকে তাহলে কিন্তু সেটা মনিটরে দেখাবে না। চেষ্টা করে দেখো কী দেখায়।

নিজে কর: `fprintf`-কে `printf` দিয়ে পালটে দিয়ে আউটপুটগুলো মনিটরে দেখো। যদি বাংলা লেখা থাকে তাহলে মনিটরে কী দেখা যায়?

নিজে কর: এর আগে সব প্রোগ্রামের সব আউটপুট মনিটরে দেখাত। এখন তুমি সেগুলো ফাইলে লিখে রাখতে পারবে?

ভাইরাল সংক্রমণ সিমুলেশন

2020 সালে পৃথিবীব্যাপী করোনা ভাইরাসের সংক্রমন হয়েছিল তখন এই সংক্রমণ কীভাবে ছড়ায় সেটা বের করার জন্য নানা ধরনের মডেলিং করা হয়েছিল, এ রকম বহুল ব্যবহৃত একটা মডেলের নাম SIR মডেল। এই মডেলে একটা এলাকার মানুষকে তিন ভাগে ভাগ করা হয়েছে:

S: (Susceptible): জনসংখ্যার কত অংশ এখনও সংক্রমিত হয়নি, কিন্তু হতে পারে।

I: (Infected): জনসংখ্যার কত অংশ এখন পর্যন্ত সংক্রমিত।

R: (Recovered): জনসংখ্যার কত অংশ ভালো হয়ে গেছে (কিংবা মারা গেছে) সেজন্য আর সংক্রমিত হবে না।

এই মডেল নিয়ন্ত্রণ করে তিনটা সমীকরণ (আমি অনেক সহজ করে লিখছি!) হচ্ছে:

$$\frac{dI}{dt} = R_0 SI - I$$
$$\frac{dS}{dt} = -R_0 SI$$
$$I + R + S = 1$$

যারা কখনো ক্যালকুলাস করেনি, কিংবা যাদের সমীকরণ দেখলে মাথায় রক্ত উঠে যায় তাদের সমীকরণগুলো দেখারও দরকার নেই। পরিশিষ্টে সব ব্যাখ্যা করা আছে, যারা দেখতে চায় দেখবে, বুঝতে চায় বুঝবে। এবারে সরাসারি আমি প্রোগ্রামের এই দুটি লাইন লিখে ফেলি:

```
deli = R0*suscept[i]*infct[i]*delt-infct[i]*delt;
dels = -R0*suscept[i]*infct[i]*delt;
```

এখানে R_0 (`R0`) একটা সংখ্যা, এই সংখ্যাটি সংক্রমণের হারকে বোঝায় (এটি R: Recovered থেকে পুরোপুরি ভিন্ন বিষয়)। $R_0 = 2.5$ অনেক বড় সংক্রমণ। R_0-এর মান 1 থেকে কম হলে সংক্রমণ থেমে যেতে শুরু করে।

এখানে ΔI (`deli`) বলতে বোঝাচ্ছে Δt (`delt`) সময়ে I কতখানি বেড়েছে। একইভাবে ΔS (`dels`) বোঝাচ্ছে Δt সময়ে S কত বেড়েছে। কাজেই আমি যদি কোনো একটা সময়ে I এবং S এর মান জানি তাহলে Δt সময় পরে সেটা বেড়ে কত হবে বের করতে পারব। অর্থাৎ:

```
infct[i+1] = infct[i]+deli;
suscept[i+1] = suscept[i]+dels;
```

তৃতীয় সমীকরণটা বেশি সোজা, এখানে না লিখলেও ক্ষতি ছিল না, তবু লিখে দেওয়া হলো:

```
recovrd[i+1] = 1-infct[i+1]-suscept[i+1];
```

প্রোগ্রামটির বাকি অংশ রুটিন মাফিক কাজ। `i=1` থেকে শুরু করে, এক এক করে বাড়িয়ে যাব। সমীকরণটি এমনভাবে সাজানো হয়েছে যে `delt` (Δt)-এর মান মোটামুটিভাবে 1 দিন। কাজেই `i` এক বাড়ানোর অর্থ সময় একদিন বাড়ানো।

প্রথমে `S=1, I=0` এবং `R=0`, মান দিয়ে শুরু করে আমরা দেখব ধীরে ধীরে `S` কমবে, `I` এবং `R` বাড়বে। একসময় `I` আর বাড়তে পারবে না, সেটা কমতে শুরু করবে, ঠিক সত্যিকারের সংক্রমণে যা হয়।

এবারে পুরো প্রোগ্রামটা লিখে ফেলি। একমাত্র যে বিষয়টা আলাদাভাবে লক্ষ্ করতে পারো শুরুতে `S=1` এবং `I=0` না ধরে আমরা ধরেছি

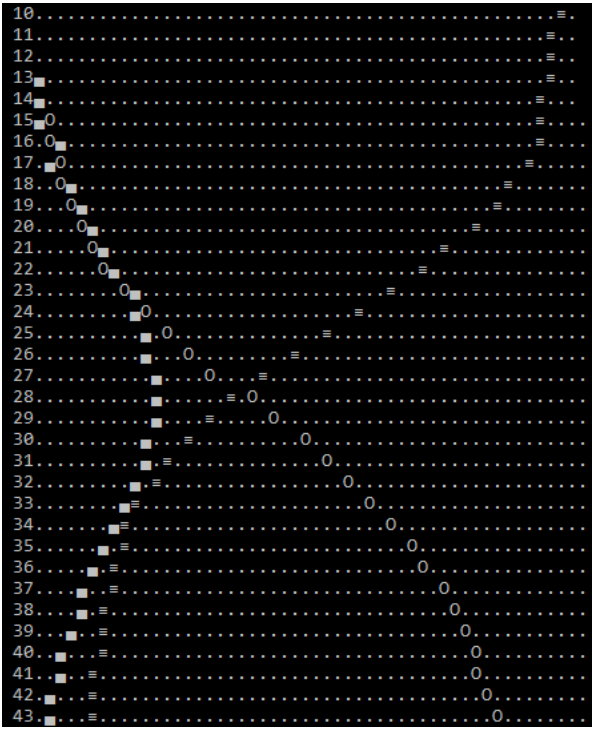
```
suscept[1]=0.999;
infct[1]=0.001;
```

অর্থাৎ পুরোপুরি `1` এবং `0` নয়, তবে খুব কাছাকাছি! এটি করা না হলে প্রোগ্রামটা শুরু করতে পারে না। কেন, নিজেরা চিন্তা করে বের করে নাও!

```
#include<stdio.h>
int main()
{
float          suscept[101],          infct[101],
recovrd[101],delt, deli, dels,R0;
```

```
int i, j;
printf(" Value of R0 : ");
scanf("%f",&R0);
delt=0.2;
suscept[1]=0.999;
infct[1]=0.001;
for(i=1;i<=100;i++){
deli       =       R0*suscept[i]*infct[i]*delt-
infct[i]*delt;
dels = -R0*suscept[i]*infct[i]*delt;
infct[i+1] = infct[i]+deli;
suscept[i+1] = suscept[i]+dels;
recovrd[i+1] = 1-infct[i+1]-suscept[i+1];
}
printf("Day         Susceptible         Infected
Recovered\n");
for(i=1;i<=100;i++)printf("%3d    %f    %f    %f
\n",i,suscept[i+1],infct[i+1],recovrd[i+1]);
printf("Susceptible %c  Infected %c  Recovered
%c \n", 240, 220, 79);
for(j=1;j<=100;j++){printf("%3d",j);for(i=1;i<
=50;i++){if(i!=(int)(50*infct[j]))printf(".");

if(i==(int)(50*infct[j]))printf("%c",220);

if(i==(int)(50*recovrd[j]))printf("%c",79);

if(i==(int)(50*suscept[j]))printf("%c",240);
        }printf("\n");}
}
```

প্রোগ্রামটা রান করার সময় সেটি তোমার কাছে R_0-এর মান চাইবে। আগেই বলেছি $R_0 = 2.5$ বড় সংক্রমণ, সেটা দিয়ে চেষ্টা করো (ছবি 4), তারপর ধীরে ধীরে কমিয়ে দেখো কী হয়!



ছবি 4: ভাইরাসের সংক্রমণ। Susceptible উপর থেকে নিচে নেমে এসেছে। Recovered নিচে থেকে ধীরে ধীরে উপরে উঠেছে। Infected বেড়ে বেড়ে একটা সর্বোচ্চ উচ্চতায় পৌঁছে আবার নিচে নেমে এসেছে।

নিজে করো: প্রোগ্রামটা এমনভাবে পরিবর্তন করো, যেন প্রথম কিছুদিন R_0-এর মান 2.5-এর বেশি দেয়া যায়, তারপর এর মান 1 থেকে কমিয়ে ফেলা যায়। সংক্রমণ কীভাবে বেড়ে উঠে হঠাৎ কীভাবে কমতে থাকে দেখতে পাবে।

ব্ল্যাক হোল ঘিরে স্যাটেলাইট

আমি জানি এখানে ব্ল্যাক হোল শব্দটা ব্যবহার করার কোনো প্রয়োজন ছিল না, শিরোনামটি চমকপ্রদ করা ছাড়া এই শব্দটির কোনো গুরুত্ব নেই! কারণ শুধু ব্ল্যাকহোলের ইভেন্ট হোরাইজনের কাছাকাছি গেলে তার বৈশিষ্ট্যগুলো বোঝা যায়, দূর থেকে ব্ল্যাক হোলের প্রভাব এবং সাধারণ গ্রহ-নক্ষত্রের প্রভাবের মাঝে কোনো পার্থক্য নেই!

তবে আমরা যেহেতু একটা বিন্দুতে কেন্দ্রীভূত অনেক বড় একটা ভরের কাছাকাছি একটা স্যাটেলাইট গেলে তার গতিবিধি কেমন হয় সেটা বের করব, তাই ব্ল্যাক হোল শব্দটা ব্যবহার করে খুব বড় কোনো প্রতারণা করা হয়নি!

ধরা যাক এর ভর M, কাজেই m ভরের একটা স্যাটেলাইট (কিংবা অন্য কিছু) r দূরত্বে থাকলে সেটি একটা বল অনুভব করবে।

$$\mathrm{F} = \frac{GMm}{r^2}$$

এখন সেটাকে যদি ছেড়ে দেয়া হয়, তাহলে এটা সোজাসুজি কেন্দ্রবিন্দুটিতে এসে পড়বে। যদি অন্য কোনো দিকে একটা গতি দিয়ে ছেড়ে দেওয়া হয়, তাহলে এই বিন্দুটির দিকে আকর্ষিত হওয়ার জন্য তার গতিপথের পরিবর্তন হবে। এ প্রোগ্রামটি লেখা হয়েছে এই গতিপথটি বের করার জন্য। যারা পুরোটি বুঝতে চায় তাদের জন্য পরিশিষ্টে সব ব্যাখ্যা করে দেওয়া হয়েছে, নবম কিংবা দশম শ্রেণির পদার্থবিজ্ঞান জানা থাকলেই এটা বুঝতে পারবে।

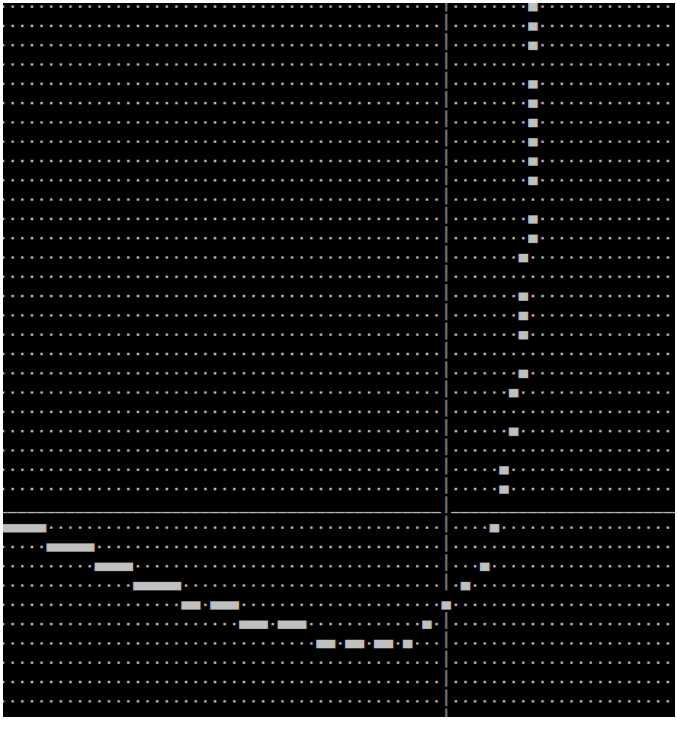
আমরা অন্য কিছু না করে আপাতত চোখ বুজে প্রোগ্রামটা লিখে ফেলি, আর কিছু না হলেও সেটাকে একটা কম্পিউটার গেমের মতো ব্যবহার করা যাবে। প্রোগ্রামটা এরকম:

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

int main()
{
float
x[502],y[502],r[502],vx[502],vy[502],delt, GM;
int i,tag;

tag =1;
while(tag!=0){
printf("x y vx vy: ");// 0 40 1 0
scanf("%f %f %f %f", &x[0], &y[0],  &vx[0],
&vy[0]);

GM=40; //GM (3 solar mass) unit of 10**19,
x,y,v unit of 10**6 in MKS
delt=.5;
for(i=0;i<=500;i++){
r[i] =sqrt(x[i]*x[i]+y[i]*y[i]);
vx[i+1]=vx[i]-(x[i]/r[i])*delt*GM/(r[i]*r[i]);
vy[i+1]=vy[i]-(y[i]/r[i])*delt*GM/(r[i]*r[i]);
x[i+1]=(x[i]+delt*vx[i+1]);
y[i+1]=(y[i]+delt*vy[i+1]);
                    }

/*
float pot, kin, tot;
for(i=0;i<=500;i++){
pot=-GM/r[i];
kin=(vx[i]*vx[i]+vy[i]*vy[i])/2;
tot=pot+kin;
```

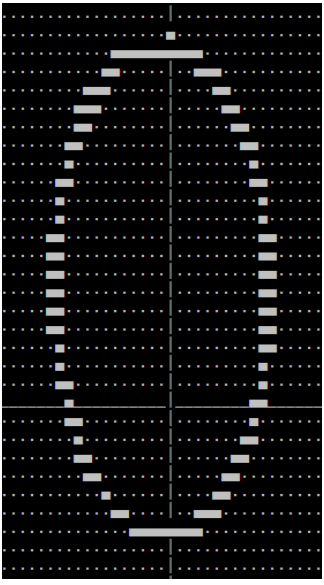
```
printf("%5d %7.3f %7.3f %7.3f %7.3f%7.3f %7.3f
%7.3f %7.3f\n",i, x[i],y[i],r[i],vx[i], vy[i],
kin, pot, tot);
}
*/

int j,ix,iy;
char c[101][101];
for(i=0;i<=100;i++)for(j=0;j<=100;j++)c[i][j]=
'.';
for(i=0;i<=100;i++){c[i][50]='_';c[50][i]='|';
}
c[100][50]='X';c[50][100]='Y';
for(i=0;i<=500;i++){ix=(int)(x[i]+50);
                    iy=(int)(y[i]+50);
                    if(ix>=0 && ix
<=100)if(iy>=0 && iy <=100)c[ix][iy]=220;
                    }
for(i=0;i<=100;i++){

{for(j=0;j<=100;j++)printf("%c",c[j][100-
i]);}printf("\n");
                   }

float error, errortot;
errortot=0;
for(i=0;i<=500;i++){
       error =(-
GM/r[0]+(vx[0]*vx[0]+vy[0]*vy[0])/2)-(-
GM/r[i]+(vx[i]*vx[i]+vy[i]*vy[i])/2);
       errortot=errortot+error*error;
                   }
printf("Error of this simulation:
%f\n\n",errortot/5);

printf("type 1 to continue 0 to quit: ");
scanf("%d",&tag);
}
}
```

প্রোগ্রামটা রান করলে দেখবে এটা তোমার কাছে জানতে চাইবে, শুরুতে এটা কোন অবস্থানে আছে (`x` এবং `y`-এর মান) এবং `x` এবং `y`-এর দিকে এর প্রাথমিক বেগ (`vx[0]` এবং `vy[0]`) কত:

```
x y vx vy:
```



ছবি 5 : স্যাটেলাইটটি সোজা নিচের দিকে ছুড়ে দেওয়া হয়েছে, (x=10, y=40, vx=0, vy=-2) ব্ল্যাকহোলের আকর্ষণে তার গতিপথ বাম দিকে বেঁকে যাচ্ছে।

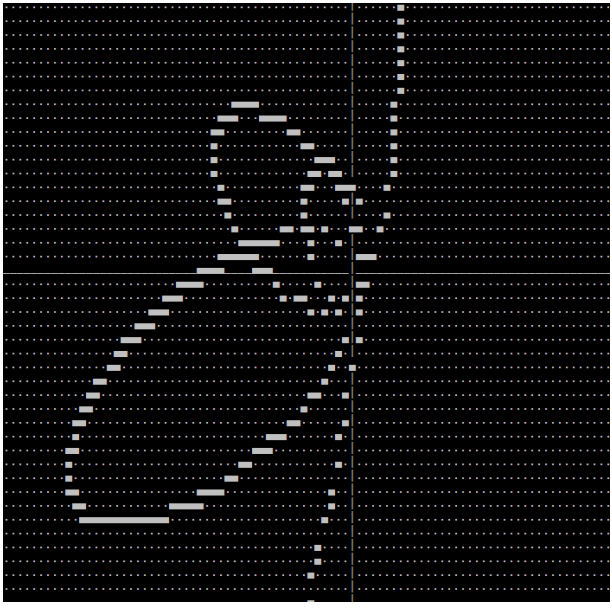


ছবি 6 : বিশেষ ক্ষেত্রে স্যাটালাইটটি ব্ল্যাক হোলের আকর্ষণে আবদ্ধ হয়ে যেতে পারে (x=0, y=20, vx=1, vy=0), এখানে যে রকম হয়েছে!

তুমি প্রথমে টাইপ করো: 10 40 0 -2 (x=10, y=40, vx=0, vy=-2) যার অর্থ এটি কেন্দ্রবিন্দু থেকে x দিকে 10 হাজার কিলোমিটার এবং y থেকে 40 হাজার কিলোমিটার দূরে আছে এবং স্যাটেলাইটটিকে নিচের দিকে (যেহেতু এর মান নেগেটিভ) সেকেন্ডে 2 হাজার কিলোমিটার বেগে ছুড়ে দেওয়া হলো। প্রথমে এটি সত্যি সত্যি ছবি 5 এর মত নিচের দিকে যাবে কিন্তু ব্ল্যাক হোলের আকর্ষণে এটি দিক পরিবর্তন করে অন্যদিকে চলে যাবে।

এর পর 0 40 1 0 লিখো, (x=0,y=40,vx=1,vy=0)। যার অর্থ এটি কেন্দ্রবিন্দু (ব্ল্যাকহোল) থেকে y অক্ষ বরাবর 40 হাজার কিলোমিটার দূরে থাকার সময় x-এর দিকে প্রতি সেকেন্ডে 1 হাজার কিলোমিটার বেগে স্যাটেলাইটটি ছুড়ে দেওয়া হলো। এই অবস্থায় স্যাটেলাইটটি

একটা কক্ষপথে আটকা পড়ে এই ব্ল্যাকহোলকে ঘিরে বৃত্তাকারে ঘুরতে থাকবে। (তোমার এই হাস্যকর প্লট করার প্রোগ্রাম অবশ্য এটাকে বৃত্তাকার না দেখিয়ে ডিমের মতো দেখাবে।) ছবি 6-তে এরকম কিন্তু ভিন্ন আরেকটা উদাহরণ দেখানো হয়েছে।



ছবি 7: এই অতি বিচিত্র গতিপথটি আসলে সত্যি নয়! ব্ল্যাকহোলের বেশি কাছে চলে যাওয়ার কারণে হিসাবে ভুল হয়ে এটা দেখাচ্ছে (`x=0, y=40, vx=0.2546, vy=0`)!

এই দুই উদাহরণেই `x` এবং `y`-এর মান ঠিক রেখে গতিবেগ বাড়াও এবং কমাও, দেখো কী হয়। মূল প্রোগ্রামে মাত্র চারটি লাইনে

পুরো হিসাব করা হয়েছে, কাজেই ব্ল্যাকহোলের বেশি কাছে গেলে ত্রুটি বা error বেড়ে যেতে থাকে। যদি ত্রুটি বা error-এর মান 1 থেকে বেশি হয় তাহলে সেই কক্ষপথ বিশ্বাসযোগ্য নয় (ছবি 7) ধরে নিতে হবে।

বিশ্বাস করো আর নাই করো, এই চার লাইনের প্রোগ্রামে কিন্তু অনেক কিছু শেখার আছে। যদি সত্যি সত্যি একটা প্লট করার সফটওয়্যার ব্যবহার করতে পারো তাহলে দেখবে কত মজা!

শেষ কথা

তুমি "শেষ কথা" পর্যন্ত পৌঁছে গেছো। এখানে নানাভাবে পৌঁছানো সম্ভব—যদি বইটা মনোযোগ দিয়ে পড়তে পড়তে এবং কম্পিউটারে প্রোগ্রামিং করতে করতে পৌঁছে থাকো তাহলে বলা যায় তুমি তোমার নিজের কাজ চালানোর মতো প্রোগ্রামিং শিখে গেছো। এখন সত্যিকার প্রোগ্রামার হতে চাও কি না সেটা তোমার ইচ্ছা। আমি যেহেতু বইটার নাম দিয়েছি শর্টকাট প্রোগ্রামিং এবং আসলেই অনেক কিছু শর্টকাট করেছি, তাই ইচ্ছে করে অনেক গুরুত্বপূর্ণ তথ্য দিইনি। আমি নিশ্চিত যদি সত্যি সত্যি কেউ প্রোগ্রামিংয়ে মজা পেয়ে যাও আর নূতন নূতন প্রোগ্রাম লিখতে থাকো তাহলে মাঝে মাঝেই নূতন নূতন সমস্যায় পড়বে, অনেকগুলো নিজেই সমাধান করে ফেলবে। অনেকগুলোর সমাধান বের করার জন্য ইন্টারনেট ঘাঁটিঘাঁটি করতে হবে কিন্তু সেটা নিয়ে আমি মোটেও মাথা ঘামাচ্ছি না। তোমরা নিশ্চয়ই লক্ষ করেছো প্রোগ্রামিং শেখার জন্য আমি কিন্তু কম্পিউটারের সমস্যা নিইনি,

গণিতের সমস্যা নিয়েছি, বায়োলজি কিংবা পদার্থবিজ্ঞানের সমস্যা নিয়েছি। আমি চাই তোমরা বিশ্বাস করতে শেখো প্রোগ্রামিং শুধু প্রোগ্রামারদের জন্য নয়, সবার জন্য! (যদি সত্যিকার প্রোগ্রামার হতে চাও তাহলে এখানে যা শিখেছো মনে হয় তার সবকিছু ভুলে আবার নূতন করে সবকিছু শিখতে হবে!) সবাই প্রোগ্রামার হবে না—যারা আমার মতো শুধু নিজের প্রয়োজন মেটানোর জন্য এবং কাজ চালানোর জন্য প্রোগ্রামিং করতে চায় তাদের জীবন কিন্তু বেশ মজার—বিশ্বাস করো।

আর তোমরা যদি বইটা গল্প বইয়ের মত পড়ে পড়ে এখানে পৌঁছে থাকো তাহলে আমি খুবই দুঃখিত, তোমার সময় নষ্ট করার জন্য!

প্রোগ্রামিং করতে গিয়ে আমি যখনই আটকে যাই আমার ছাত্রছাত্রীদের সেটা জিজ্ঞেস করতে পারি। তোমাদের সবার আমার মতো ছাত্রছাত্রী নেই কিন্তু ইন্টারনেট বলে একটা জ্ঞানের খনি আছে। এমন কোনো প্রশ্ন নেই যার উত্তর তুমি এখান থেকে পেয়ে যাবে না। কাজেই ঘাবড়ানোর কোনো কারণ নেই।

সত্যি সত্যি তুমি যদি কিছু একটা শিখে থাকো, তাহলে কিন্তু তোমাকে নিজে নিজে অনেক কিছু করতে হবে, তা না-হলে এই শেখার কোনো মূল্য থাকবে না। আমি তোমাকে কিছু আইডিয়া দিই:

(1) Buffon's law বলে একটা সূত্র আছে সেটা ব্যবহার করে দেখো π এর মান বের করতে পারো কি না।

(2) যেকোনো একটা ফাইলকে Encrypt এবং Decrypt করতে পারো কি না!

(3) বিশাল বিশাল দুটো সংখ্যা (একশ ডিজিট বা দুইশ ডিজিট) গুণ করতে পারো কি না!

(4) 10×10 কিংবা আরও বড় linear equation solve করতে পারো কি না।

(5) Mersenne প্রাইম বের করার একটা প্রোগ্রাম লিখতে পারো কি না!

(6) নিশ্চয়ই মনে আছে, কমপ্লেক্স নম্বর ব্যবহার করতে পারিনি বলে আমরা সব ধরনের দ্বিঘাত সমীকরণের সমাধান বের করতে পারিনি? কাজেই দেখা যাক তুমি কমপ্লেক্স নম্বর ব্যবহার করা যায় এ রকম কিছু ফাংশন তৈরি করতে পার কি না!

শেষ পর্যন্ত এগুলো সমাধান করতে পারবে কি না জানি না, কিন্তু এটুকু বলতে পারি এগুলো চেষ্টা করতে গিয়ে তোমার মস্তিষ্ক যথেষ্ট শানিত হবে—এর বেশি আমরা আর কিছু কী চাই?

আমি তো চাই না!

পরিশিষ্ট 1

ভাইরাল সংক্রমণ সিমুলেশন

আমরা আগেই বলেছি সংক্রমণ কীভাবে ছড়ায় সেটা বের করার জন্য বহুল ব্যবহৃত একটা মডেলের নাম SIR মডেল, যেখানে:

S: (Susceptible): জনসংখ্যার কত অংশ এখনও সংক্রমিত হয়নি, কিন্তু হতে পারে।

I: (Infected): জনসংখ্যার কত অংশ এই মুহূর্ত্তে সংক্রমিত।

R: (Recovered): জনসংখ্যার কত অংশ ভালো হয়ে গেছে (কিংবা মারা গেছে) সেজন্য আর সংক্রমিত হবে না।

এই মডেল নিয়ন্ত্রণ করে তিনটা সমীকরণ:

$$\frac{dI}{dt} = R_0 SI - I$$
$$\frac{dS}{dt} = -R_0 SI$$
$$I + R + S = 1$$

এখন প্রথম সমীকরণ দুইটি একটু অন্যভাবে লিখি, dI এবং dS না লিখে ΔI এবং ΔS লিখব, dt না লিখে Δt লিখব:

$$\Delta I = R_0 SI\Delta t - I\Delta t$$
$$\Delta S = -R_0 SI\Delta t$$

এখানে ΔI বলতে বোঝাচ্ছে Δt সময়ে I কতখানি বেড়েছে। একইভাবে ΔS বোঝাচ্ছে Δt সময়ে S কত বেড়েছে। কাজেই আমি যদি কোনো একটা সময়ে I এবং S-এর মান জানি তাহলে Δt সময় পরে সেটা বেড়ে কত হবে বের করতে পারব। এবারে সরাসরি আমি প্রোগ্রামের এই দুইটি লাইন লিখে ফেলতে পারব:

```
deli = R0*suscept[i]*infct[i]*delt-
infct[i]*delt;
dels = -R0*suscept[i]*infct[i]*delt;
```

অর্থাৎ আমি যদি কোনো একটা সময়ে S-এর মান (কোন একটা `i` এ `suscept[i]`-এর মান) এবং I-এর মান (অর্থাৎ কোনো একটা `i`-এ `infect[i]`-এর মান) জানি তাহলে Δt সময় পরে S এবং I কতটুকু বাড়বে (ΔS এবং ΔI) সেটা বের করতে পারব। অর্থাৎ,

```
infct[i+1] = infct[i]+deli;
suscept[i+1] = suscept[i]+dels;
```

এই দুটো যদি জানি তাহলে R বের করা খুবই সোজা:

```
recovrd[i+1] = 1-infct[i+1]-suscept[i+1];
```

প্রোগ্রামটির বাকি অংশ রুটিন মাফিক কাজ। এখানে কী করতে হবে আগেই বলা হয়েছে।

পরিশিষ্ট 2

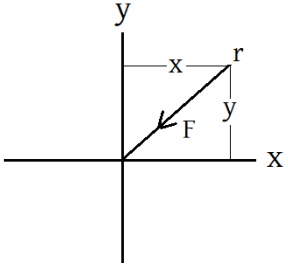
ব্ল্যাক হোল ঘিরে স্যাটেলাইট

ধরা যাক এর ভর M, কাজেই m ভরের একটা স্যাটেলাইট r দূরত্বে থাকলে সেটি একটা বল অনুভব করবে। বলের মান হবে

$$F = \frac{GMm}{r^2}$$

এই বলের জন্য স্যাটেলাইটের ত্বরণ হবে

$$a = \frac{F}{m} = \frac{GM}{r^2}$$



ছবি 8 : কেন্দ্র থেকে r দূরত্বে থাকা একটি ভর কেন্দ্রের দিকে F বল অনুভব করে।

ছবিতে x এবং y দেখানো হয়েছে। r হচ্ছে:

$$r = \sqrt{x^2 + y^2}$$

কাজেই x এবং y দিকের ত্বরণ হবে যথাক্রমে—

$$a_x = \left(\frac{x}{r}\right)\frac{GM}{r^2}$$
$$a_y = \left(\frac{y}{r}\right)\frac{GM}{r^2}$$

x এবং y দিকে যদি স্যাটেলাইটের আদিবেগ হয়ে থাকে v_{x0} এবং v_{y0}, তাহলে Δt সময় পরে তার বেগ হবে:

$$v_x = v_{x0} - a_x \Delta t = v_{x0} - \left(\frac{x}{r}\right)\frac{GM}{r^2}\Delta t$$
$$v_y = v_{y0} - a_y \Delta t = v_{y0} - \left(\frac{y}{r}\right)\frac{GM}{r^2}\Delta t$$

ত্বরণের মানের আগে নেগেটিভ চিহ্ন এসেছে, কারণ x বা y যেদিকে বাড়ে বল এবং ত্বরণের দিক তার বিপরীত দিকে। তখন তার অবস্থান, অর্থাৎ x এবং y-এর মান হবে:

$$x = x_0 + v_x \Delta t$$
$$y = y_0 + v_y \Delta t$$

কাজেই আমরা যদি শুরুতে স্যাটেলাইটের অবস্থান x_0, y_0 এবং আদি বেগ v_{x0} ও v_{y0} জানি তাহলে সময়ের সাথে সাথে তার অবস্থান কী হবে সেটা বের করে ফেলতে পারব।

অর্থাৎ আমরা যদি একটা ইনডেক্স i দিয়ে সময়টা নির্ধারণ করি তাহলে প্রোগ্রামের গুরুত্বপূর্ণ লাইনগুলো লিখে ফেলতে পারি:

```
r =sqrt(x[i]*x[i]+y[i]*y[i]);

vx[i+1]=vx[i]-x[i]*delt*GM/(r*r*r);
vy[i+1]=vy[i]-y[i]*delt*GM/(r*r*r);

x[i+1]=x[i]+delt*vx[i];
y[i+1]=y[i]+delt*vy[i];
```

এখানে আলাদা আলাদাভাবে না করে G এবং M গুণ করে তার মানটা GM হিসেবে ব্যবহার করা হয়েছে। (GM=40 তার অর্থ কোনো একটি এককে M-এর ভর তিনটি সূর্যের ভরের সমান, তখন x ও y-এর মান হাজার কিলোমিটারে এবং vx ও vy এর মান

হবে সেকেন্ডে হাজার কিলোমিটারে)। স্যাটেলাইটের ভরের ওপর যেহেতু গতিপথ নির্ভর করে না, তাই একক ভর (m=1) ধরা হয়েছে।

মহাকর্ষ বলের ভেতরে কিছু চলতে থাকলে তার গতি শক্তি (T) এবং বিভব শক্তি (V) দুটোই থাকে। দুটোর যোগফল (E) হচ্ছে তার পুরো শক্তি। স্যাটেলাইটের একক ভর ধরে আমরা লিখতে পারি:

$$T = \left(v_x^2 + v_y^2\right)/2$$
$$V = -\frac{GM}{r}$$
$$E = T + V$$

সেটা যদি নেগেটিভ হয় তাহলে গতিপথটি আবদ্ধ হয়, স্যাটেলাইটটি ব্ল্যাকহোলের আকর্ষণে চিরকালের জন্য আটকা পড়ে যায়। সেটি যদি পজিটিভ হয় তাহলে ব্ল্যাকহোলের আকর্ষণ উপেক্ষা করে স্যাটেলাইটটি চলে যেতে পারবে। গতি শক্তি এবং বিভব শক্তির পরিবর্তন হতে পারে কিন্তু তাদের যে যোগফল, সেই পুরো শক্তির কোনো পরিবর্তন হয় না। এই প্রোগ্রামে সেই পুরো শক্তি যদি পালটে যেতে দেখা যায়, তাহলে বুঝতে হবে হিসাবে কিছু একটা ভুল হচ্ছে, সে জন্যে পুরো পরিবর্তনের দিকে নজর রেখে ত্রুটি বা error অনুমান করা হয়েছে।

প্রোগ্রামটির বাকি অংশ রুটিন মাফিক কাজ। এখানে কী করতে হবে সেটা আগেই বলা হয়েছে।