**Competitive Programming**

**Ch1 - Intro**

Having describes the idea of competitive programming , and this field have many aspect, but two main things in CP are we **design** and **implement algorithms**.

For **designing algorithms** we must have skills in problem solving and mathematical thinking. The algorithm we design has to be both correct and efficient. For solving problems of course we have to know some algorithm.

**The implementation of algorithms** requires good programming skills (soon we learn this). By solving problems it means that not only the idea of algorithm is correct, but also the implementation has to be correct. We solving problem in straightforward and concise style. The writing (or typing) process should be completed quickly.

**Programming Languages**

We use c++ here, but still if you want to using other lang is up to you.

**C++ code template**

#include <bits/stdc++.h>

using namespace std;

int main(){

}

**Input & Output**

Inputs can be read from the cin stream as follows:

    int a,b;

    string x;

    cin >> a >> b >> x;

int stands for integer(1,2,3…, note that 1.5 is not an integer), meanwhile string is just text(abcde1234@?/, etc).

For multiple inputs we can use *space*

123 456 goodluckForyou

The cout stream is used for output as follows

int a = 123, b = 456;

    string x = "goodluckForyou";

    cout << a << " " << b << " " << x << "\n";

note that string has to be written inside “…”.

the following lines at the beginning of the code make input and outpu more efficient

    ios::sync\_with\_stdio(0);

    cin.tie(0);

**Remember** that newline **“\n”** works **faster** than endl.

For whole line input that containing spaces we can use getline function

string s;

    getline(cin,s);

for unknown amount of data:

string x;

    while (cin >> x){

        // code

    }

Ctrl+c to exit

You can use the following code to read and write txt

    freopen("input.txt","r",stdin);

    freopen("output.txt","w",stdout);

**Working with numbers**

**Integers**

Types:

int : has a value range of (−231 . . . 231− 1) or 32 bit

long long : has a value range of (−263 . . . 263− 1) or 64 bit

\_\_int128\_t : 128 bit

    long long x = 123456789123456789LL;

Usually long long is enough. If you are using int somewhere before long long:

int a = 123456;

    long long b = (long long) a\*a;

    cout << b << "\n";

**Modular arithmetic**

We denote by x mod m the remainder when x is divided by m. For example, 18 mod 7 = 4, because 18 = 7·2 + 4.

Here are the properties:

(a+b) mod m = (a mod m + b mod m) mod m

(a-b) mod m = (a mod m - b mod m) mod m

(a·b) mod m = (a mod m · b mod m) mod m

Thus, we can take the remainder after every operation and the number will never become too large. For example, the following code calculates n!, the factorial of n, modulo m:

    long long x = 1;

    int n = 4;

    int m = 5;

    for (int i=2; i<=n; i++){

        x = (x\*i)%m;

    }

    cout << x%m << "\n";

**Shortening Code**

    typedef long long ll;

    ll a = 12345;