Macro

```
#define nl "\n" #define ll long long #define ull unsigned long long #define pb push_back #define SIZE(a) (int)a.size()
#define SORT(v) sort(v.begin(),v.end())
#define RSORT(v) sort(v.rbegin(),v.rend())
#define REV(v) reverse(v.begin(),v.end())
#define ff first
#define ss second
#define sq(a) ((a)*(a))
#define For(i,a,b) for(i=a;i<=b;i++)
#define Rof(i,a,b) for(i=a;i>=b;i--)
#define Rep(i,b) for(i=0;i<b;i++)
#define MOD 1000000007
#define PI acos(-1.0)
#define eps 1e-9
#define Linf 2e18
#define inf 1<<30
#define MX5 100005
#define MX6 1000006
#define MX3 1005
#define GCD(a,b) __gcd(a,b)
#define Abs(a) abs(a)
#define input(a,b) scanf("%lld%lld",&a,&b)
#define in1(a) scanf("%lld",&a)
#define output(a) printf("%lld\n",a);
#define mem(a) memset(a,-1,sizeof(a))
#define clr(a) memset(a,0,sizeof (a))
#define mk make_pair
#define pLL pair<ll,ll>
#define ff first
#define ss second
#define invcos(a) (acos(a)*(180/3.14159265))
#define rtan(a) (tan(a*3.14159265/180.0))
#define pip printf("pip")
///4direction -> int del_x[]={-1,0,1,0},del_y[]={0,1,0,-1};
///8direction -> int del_x[]={-1,0,1,0,1,-1,-1,1},del_y[]={0,1,0,-1,-1,-1,1,1};
#include <bits/stdc++.h>
using namespace std;
ll bigmod(ll n,ll p){if(p==0) return 1;if(p==1)return (n+MOD)%MOD;if(p%2)return (bigmod(n,p-1)*n+MOD)%MOD;else{ll x=bigmod(n,p/2);return (x*x+MOD)%MOD;}}
ll modinverse(ll n){return bigmod(n,MOD-2)%MOD;}
```

```
/*----------------------Graph Moves----------------*/
//const int fx[]={+1,-1,+0,+0}; //const int fy[]={+0,+0,+1,-1};
//const int fx[]={+0,+0,+1,-1,-1,+1,-1,+1};   // Kings Move
//const int fy[]={-1,+1,+0,+0,+1,+1,-1,-1};  // Kings Move
//const int fx[]={-2, -2, -1, -1,  1,  1,  2,  2};  // Knights Move
//const int fy[]={-1,  1, -2,  2, -2,  2, -1,  1}; // Knights Move
/*-----------------------------------------------*/
```

Extended Euclid

```
int d, x, y;
void extendedEuclid(int A, int B) {if(B == 0) {d = A;x = 1; y = 0;}
 else { extendedEuclid(B, A%B);int temp = x; x = y;y = temp - (A/B)*y;}}
```

BFS

```
vector<int>v[100];Int nodes[100][100], i,j,a,b,node,edge,n,visited[100],label[100];
visited[source]=1;queue<int>Q;Q.push(source);
while(!Q.empty()){int n=Q.front();if(n==destination){cout << label[n] << endl;return
0;}Q.pop();
for(i=0;i<v[n].size();i++){if(visited[v[n][i]]==0){label[v[n][i]]=label[n]+1;visited[v[n][i]]=1;
Q.push(v[n][i]);}}}
```

DFS

```
bool check(ll x,ll y){if(x>=1 && x<=m && y>=1 && y<=n){return true;}return false;}
void dfs(ll a,ll b){for(ll i=0;i<4;i++){ll x=del_x[i]+a;ll y=del_y[i]+b;
if(check(x,y)==true && visited[x][y]==0 && s[x][y]=='.'){visited[x][y]=1;dfs(x,y);}}}}
dfs(x,y);//main function
```

DSU

```
int arr[1006],size[1006];
void initialize(int n){for(int i=1;i<=n;i++){arr[i]=i;size[i]=1;}}
int root(int x){while(arr[x]!=x){arr[x]=arr[arr[x]];x=arr[x];}return x;}
void weighted_union(int a, int b){
int rootA=root(a);int rootB=root(b);int mini=min(rootA,rootB);
int maxi=max(rootA,rootB);arr[mini]=arr[maxi];size[maxi]+=size[mini];size[mini]=0;}
initialize(n);weighted_union(a,b);//main functionroot(a);//main function
```

nCr

```
ll nCr(ll n,ll r){if(n < r) return 0;ll re1 = fact[n];re1 = (re1 * invmod[r]) % MOD;

re1 = (re1 * invmod[n - r]) % MOD;return re1;}
For(i=1;i<=1000000;i++) {fact[i]=(fact[i-1]*i)%MOD;invmod[i]=modinverse(fact[i]);}
```

MST(Kruscal)

```
ll parent[MX6]; vector< pair<ll, pair<ll,ll> > > get;
void init(){ll i;for(i=0;i<get.size();i++){parent[i]=i;}}
ll root(ll r){return (parent[r]==r)?r:root(parent[r]);}
```

```
ll mst(ll n){ll i,counter=0,s=0;init();
for(i=0;i<get.size();i++) {ll u=root(get[i].ss.ff);ll v=root(get[i].ss.ss);
if(u!=v){parent[u]=v;counter++;s+=get[i].ff;if(counter==n-1){break;}}}return s;}

int main(){
ll n,m,w,u,v,i;input(n,m);
for(i=1;i<=m;i++){input(u,v);in1(w);get.pb(mk(w,mk(u,v)));}SORT(get);output(mst(n));}
```

Lower bound

```
vector<int>::iterator it; it=lower_bound(v.begin(),v.end(),a);ans=it-v.begin()
```

Running Median

```
priority_queue<double>max_heap,min_heap;
vector<int>v;
for(i=0;i<n;i++){cin >> a;
int maxi=max_heap.size();int mini=min_heap.size();
if(maxi>mini){if(a<median){
int fe=max_heap.top();min_heap.push(-fe);max_heap.pop();max_heap.push(a);}
else{min_heap.push(-a);}median=(max_heap.top()+abs(min_heap.top()))/2;}
else if(maxi<mini){if(a>median){
int fe=abs(min_heap.top());max_heap.push(fe);min_heap.pop();min_heap.push(-a);}
else{max_heap.push(a);}     median=(max_heap.top()+abs(min_heap.top()))/2;}
else{if(a<median){max_heap.push(a);median=max_heap.top();}
else{min_heap.push(-a);median=abs(min_heap.top());}}cout << median << endl;}
```

Djikstra

```
#define ll long long int  #define inf 2147483647154869
vector<pair<ll,ll> >v[1000000];vector<ll>dis(1000000,inf);ll
node,edge,source=1,a,b,i,j,x;
map<ll,ll>path;
/* Main function */cin >> node >> edge;
for(i=1;i<=edge;i++){cin >> a >> b >> x;v[a].push_back({b,x});v[b].push_back({a,x});}
priority_queue<pair<ll,ll> >Q;dis[source]=0;path[source]=-1;bool check=false;
Q.push({-dis[source],source});
while(!Q.empty()){ll n=Q.top().second;Q.pop();if(n==node) check=true;
for(i=0;i<v[n].size();i++){ll t=v[n][i].first;ll
len=v[n][i].second;if(dis[n]+len<dis[t]){dis[t]=dis[n]+len;path[t]=n;Q.push({-dis[t],t});}}}
if(check==false){cout << "-1" << endl;return
0;}else{vector<ll>ans;for(i=node;i!=-1;i=path[i]){ans.push_back(i);}
reverse(ans.begin(),ans.end());for(i=0;i<ans.size();i++){cout << ans[i] << " ";}}
```

Start and finishing time in dfs

```
#define white 0 #define gray -1 #define black 1 int start[100],finish[100],visited[100];
vector<int>v[100];int node,edge,i,j,a,b,tim=0;
int dfs(int source){int u=source;tim++;start[u]=tim;visited[u]=gray;
```

```
for(i=0;i<v[u].size();i++){if(visited[v[u][i]]==white){dfs(v[u][i]);}}
visited[u]=black;tim++;finish[u]=tim;return finish[u];}
int main(){cin >> node >> edge;
for(i=1;i<=edge;i++){cin >> a >> b;v[a].push_back(b);}
int source;cin >> source;dfs(source);for(i=1;i<=node;i++){cout << start[i] << " " <<
finish[i] << endl;}}
```

Topological Sort

```
int indegree[200000];vector<int>v[200000];int visited[200000];
//main function
int node,edge,i,a,b,counter=0;cin >> node >> edge;vector<int>ans;
for(i=0;i<edge;i++){cin >> a >> b;v[a].push_back(b);indegree[b]++;}
queue<int>Q;for(i=1;i<=node;i++){if(indegree[i]==0){Q.push(i);}}counter=node;
while(!Q.empty()){int n=Q.front();ans.push_back(n);node--;Q.pop();
for(i=0;i<v[n].size();i++){indegree[v[n][i]]--;
if(indegree[v[n][i]]==0){Q.push(v[n][i]);}}}if(node){cout << "Impossible" << endl;}
else{for(i=0;i<ans.size();i++){cout << ans[i] << " " << endl;}}
```

Pattern Finding(Using function in c++)

```
string txt,pat;int found = txt.find(pat);
while (found != string::npos) { cout << "Pattern found at index " << found <<
endl;//starting index of pattern         found = txt.find(pat, found + 1);}
```

Segment Tree

```
int arr[1000],v[1000];int low=0,high,pos=0;
void make_tree(int low,int high, int pos){if(low==high){arr[pos]=v[low];return;}
int mid=(low+high)/2;make_tree(low,mid,(2*pos+1));
make_tree(mid+1,high,(2*pos+2));arr[pos]=min(arr[(2*pos+1)],arr[(2*pos+2)]);}
int range_query(int qlow,int qhigh,int low,int high,int pos){
if(qlow<=low && qhigh>=high){return arr[pos];}
if(qlow>high || qhigh<low){return INT_MAX;}int mid=(low+high)/2;
return
min(range_query(qlow,qhigh,low,mid,(2*pos+1)),range_query(qlow,qhigh,mid+1,high
,(2*pos+2)));}
int main(){int n,a,i;//array size=(2*n)-1;cin >> n;for(i=0;i<n;i++){cin >> v[i];}
make_tree(low,n,pos);cin >> a >> i;cout << range_query(a-1,i-1,0,n,0) << endl;}
```

0-1 large rectangle

```
int main(){
stack<pair<int,int> >S;int n,a,i,maxi=0;cin >> n;
for(i=0;i<n;i++) {cin >> a;if(S.empty()){S.push(make_pair(a,i));}else{int
fe=S.top().first;if(a>=fe){S.push(make_pair(a,i));}else {
int value,index,prev;while(!S.empty()){value=S.top().first;index=S.top().second;
if(index<a)break;S.pop();int calc=(i-index)*value;maxi=max(calc,maxi);prev=index;}
S.push(make_pair(a,prev));}}}
```

```
while(!S.empty()) {int value=S.top().first;int index=S.top().second;S.pop();
int calc=(n-index)*value;maxi=max(calc,maxi);}cout << maxi << endl;}
```

KMP

```
#include <bits/stdc++.h>
using namespace std;
// Fills lps[] for given patttern pat[0..M-1]
void computeLPSArray(string pat, int M, int* lps){
// length of the previous longest prefix suffix
int len = 0;
lps[0] = 0; // lps[0] is always 0
// the loop calculates lps[i] for i = 1 to M-1
int i = 1;
while (i < M) {
if (pat[i] == pat[len]) {
len++;lps[i] = len;i++;}
else // (pat[i] != pat[len]){
// This is tricky. Consider the example.
// AAACAAAA and i = 7. The idea is similar to search step.
if (len != 0) {
len = lps[len - 1];
// Also, note that we do not increment i here}
else // if (len == 0){lps[i] = 0;i++;}}}}
// Prints occurrences of txt[] in pat[]
void KMPSearch(string pat,string txt){
int M=pat.size();int N =txt.size();
// create lps[] that will hold the longest prefix suffix values for pattern
int lps[M];
// Preprocess the pattern (calculate lps[] array)
computeLPSArray(pat, M, lps);
int i = 0; // index for txt[]
int j = 0; // index for pat[]
while (i < N) {if (pat[j] == txt[i]) {j++;i++;}
if (j == M) {printf("Found pattern at index %d\n", i - j);j = lps[j - 1];}
// mismatch after j matches
else if (i < N && pat[j] != txt[i]) {
// Do not match lps[0..lps[j-1]] characters,
// they will match anyway
if (j != 0)j = lps[j - 1];else i++;}}}
int main(){
string pat,txt;cin >> txt >> pat;KMPSearch(pat, txt);return 0;}
```

Big Integer

```
import java.math.BigInteger;
import java.util.*;
public class Main {
public static void main(String[] args) {
Scanner next=new Scanner(System.in);
while(next.hasNext()) {int n=next.nextInt();BigInteger a=next.nextBigInteger();
BigInteger sum=BigInteger.ZERO;BigInteger temp=a;
for(int i=1;i<=n;i++){BigInteger mul=BigInteger.ONE;mul=a;
mul=mul.multiply(new BigInteger(Integer.toString(i)));sum=sum.add(mul);
a=a.multiply(temp);}System.out.println(sum);}}}
```

GCD-> BigInteger gcd_pq=p.gcd(q); System.out.println(p.divide(gcd_pq) + “?/+
q.divide(gcd_pq));
MOD-> System.out.println(x.modPow(y,n));