Rick was besieged by walkers after the Governor's raid on the prison. They are approaching him from all sides. Assume Rick possesses a limitless supply of ammunition. Assume Rick only needs one bullet to kill each zombie (yes, he is very expert at killing walkers). They must be shot in the head. Take a look at how excellent he is).

As soon as he kills one walker, the remainder of the zombies advance 1 metre. There are n walkers in front of Rick, each at a different distance. Rick will perish if any walker is able to reach him. You must now determine whether he will live or die. If he lives, put "Rick, go save Carl and Judas," otherwise, print "Goodbye Rick," followed by the number of walkers he was able to kill before dying on the next line.

Rick's gun can also fire 6 rounds without reloading. He reloads in 1 second, during which time walkers advance 1 metre.

```
#include<bits/stdc++.h>

using namespace std;

void solve(){}

int32_t main() {

solve();

int T;

cin>>T;

while(T--) {

bool ans=true;

int val=0;

int n;

cin>>n;

int temp;

int mx[50001],cnt[50001];

memset(mx,0,sizeof(mx));

memset(cnt,0,sizeof(cnt));

int tp=2;

mx[0]=1;

for(int i=1;i<50001;i++) {

mx[i]=tp;

if(tp%6==0) {

i++;
mx[i]=tp;
}
tp++;
}
for(int i=0;i<n;i++) {
cin>>temp;
temp--;
cnt[temp]++;
}
for(int i=0;i<50001;i++) {
if(i>0)
cnt[i]+=cnt[i-1];
if(cnt[i]>mx[i]) {
ans=false;
val=i;
break;
}
}
if(ans)
cout<<"Rick now go and save Carl and Judas"<<endl;
else
{
val=mx[val];
cout<<"Goodbye Rick\n"<<val<<endl;
}
}
return 0;
}
```

You are given with integers a,b,c,d,m. These represent the modular equation of a curve y2mod m=(ax3+bx2+cx+d) mod m

Also, you are provided with an array A of size N. Now, your task is to find the number of pairs in the array that satisfy the given modular equation.

If (Ai,Aj) is a pair then Aj2mod m=(aAi3+bAi2+cAi+d) mod m.

Since the answer could be very large output it modulo 109+7.

```
#include <bits/stdc++.h>

using namespace std;

const int md = 1E9 + 7;

map<long long, int> mp;

int main() {

int t;

cin >> t;

while(t--) {

long long a, b, c, d, m;

cin >> a >> b >> c >> d >> m;

int n;

cin >> n;

int arr[n];

for(int i = 0; i < n; i ++) {

cin >> arr[i];

mp[(((arr[i] * arr[i]) % m) + m) % m] ++;

}

long long ans = 0;

for(int i = 0; i < n; i ++) {

long long x = (((((((a * arr[i]) % m) * ((arr[i] * arr[i]) % m)) % m) + (b * ((arr[i] * arr[i]) % m) % m) + ((c
* arr[i]) % m) + d) % m) + m) % m;

if(mp.find(x) != mp.end())

ans += mp[x];

}

cout << (ans % md) << '\n';
mp.clear();

}

return 0;

}
```

Canthi and Sami are having a game! The game is extremely similar to chess, but there is only one piece on the board, which is the Queen. In addition, Queen may only go to the top left corner.

For clarification, If Queen is placed at i,j then in a turn queen can move:
1) Any number of cells leftwards.
2) Any number of cells upwards.
3) Any number of cells Diagonally(only N-W direction).

Please note that board is quarter infinite i.e there is a top left corner but it extends to infinity in south and east direction..

```
#include <stdio.h>

#include<math.h>

int v[2000000],i,t;

double fi;

int main()

{

fi=((double)((1+sqrt(5))/2.0));

for(i=1;i<=1000000;i++)

v[i]=-1;

for(i=1;i<=1000000;i++)

v[(int)(fi*(double)i)] = (int)(fi*fi*i);

scanf("%d",&t);

while(t--){

int a,b;

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

if(v[a]==b)

printf("sami\n");
else
printf("canthi\n");
}
return 0;
}
```

Shantam is extremely wealthy, much more so than Richie Rich. Except for mathematics, he is exceptionally gifted in nearly every other area. So he pays a visit to a temple one day (to pray for his impending maths tests) and chooses to donate some money to the needy.(everyone is poor on a relative scale to Shantam). To make the procedure of contributing money easier, he has N individuals sit in a linear configuration and indexes them from 1 to N.

Their method of doing things is weird and unusual, as it is with all wealthy people. Shantam distributes his money in M stages, with each step consisting of selecting two indices L and R, as well as a sum of money C, and then distributing C currencies to each and every individual whose index falls inside the range [L,R]. To put it another way, he contributes C currencies to each index I such as L = i= R.

Fortunately, you were one of the N persons chosen, and you know all of the M steps ahead of time. Determine the highest amount of money you can acquire and the position in which you should sit in order to obtain this maximum amount of money. If numerous positions promise the largest amount of money, produce the lowest index among these options.

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

void swap(long long *l, long long *r)
{
        long long temp = *l;
        *l = *r;
        *r = temp;
}

int main()
{
  long long t, n,i, m, l,j, r, c, p, q, s, temp_l, temp_r, max, sum, pos;
long long deltas[100000];

scanf(" %lld", &t);

for(i=0;i<t;i++)
{
        memset(deltas, 0, sizeof(long long)*100000);
        scanf(" %lld %lld", &n, &m);
        scanf(" %lld %lld %lld %lld %lld %lld", &l, &r, &c, &p, &q, &s);

        for (j = 0; j < m; j++)
        {
                deltas[l] += c;
                if (r < n - 1)
                {
                        deltas[r+1] -= c;
                }

                temp_l = (l * p + r) % n + 1;
                        temp_r = (r * q + l) % n + 1;
                        l = temp_l;
                        r = temp_r;
                        if(l > r)
                          swap(&l, &r);
                        c = (c * s) % 1000000 + 1;
        }

        max = 0;
        sum = 0;
        pos = 0;
        for (j = 0; j < n; j++)
        {
                sum += deltas[j];
                if (sum > max)
                {
                        max = sum;
                        pos = j;
                }
        }

        printf("%lld %lld\n", pos, max);
  }

  return 0;
}
```

An integer array A and a number K have been provided to you.

Now you must determine whether any two items of the array A add up to the number K.

if two items are in different places in the array, they are regarded different.

Print "YES" (without quotations) if such a pair of integers exists; else, print "NO" (without quotes).

```
#include <iostream>
using namespace std;
#define f(i,a,n) for(int i=a;i<n;i++)
int main(){
int n,i;
cin>>n;
int a[n];
for(i=0;i<n;i++)
cin>>a[i];
int k;
cin>>k;
f(i,0,n){
f(j,0,n){
if(a[i]+a[j]==k)
{
cout<<"YES";
return 0;
}
}
}
cout<<"NO";
return 0;
cout<<"if(a[i]+a[j]>k)";
}
```

Question description

You are given an array *A* of length *N* which is initialised with *0*. You will be given *Q* queries of two types:

: set value *1* at index *k* in array *A*

: print the smallest index *x* which is greater than or equal to *y* and having value *1*. If there is no such index print *1*.

Note: Indexing is *1* based

```
#include<iostream>
using namespace std;
#define f(i,a,n) for(int i=a;i<n;i++)
int main()
{
int i,t,q,m,n;
cin>>t>>q;

int a[t];

f(i,0,t)

a[i]=0;

for(i=0;i<q;i++){

cin>>m>>n;

if(m==1){

a[n]=1;

}

if(m==2){

int cnt=0,j=0;

for(j=n;j<q;j++){

if(a[j]==1)

{

cnt=1;

break;

}

}

if(cnt==1)

cout<<j<<endl;

else

cout<<"-1"<<endl;

}

}

return 0;

}
```

Little Chandan is an exceptional manager - apart from his role in university as the person who has to bug everyone, in general... and if possible, try to get some work done.

He's also offered a job as the coach of the best Russian teams participating for ACM-ICPC World Finals. Now, Chandan is an extremely good coach, too. But he's a weird person who thrives on patterns in life, in general. So, he has decided that if there are *n* number of students in total, and he is supposed to divide them in camps of *k* students - he want them to be arranged in such a way that the length of names of all the students in a camp is **equal.**

I know, totally weird, right?

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>



int main()

{

   int cases, N, K, i, j, len, bins[100], flag;

   scanf("%d", &cases);

   int results[cases];

   //printf("cases: %d\n", cases);



   for(i=0;i<cases;i++) {

         flag = 0;

         for (j=0; j<100; j++) {

                bins[j] = 0;

         }



         scanf("%d %d", &N, &K);

         //printf("scanned: %d, %d\n", N, K);

                char str[N][100];



                for (j=0; j<N; j++) {

                       scanf("%s", str[j]);

                       len = strlen(str[j]);

                       //printf("%d\n", len);

                       bins[len] += 1;
```

```
                        }


                        for (j=0; j<100; j++) {
                                if (bins[j] % K != 0) {
                                        results[i] = 0;
                                        flag = 1;
                                        break;
                                }
                        }
                        if (flag == 0) {
                                results[i] = 1;
                        }
        }


        for (i=0; i<cases; i++) {
                if (results[i] == 0) {
                        printf("Not possible\n");
                }
                else {
                        printf("Possible\n");
                }
        }
        return 0;
}
```

You are given a string which comprises of lower case alphabets (a-z), upper case alphabets (A-Z), numbers, (0-9) and special characters like **!,-.;** etc.

You are supposed to find out **which character occurs the maximum number of times and the number of its occurrence,** in the given string. If two characters occur equal number of times, you have to output the character with the lower ASCII value.

For example, if your string was: **aaaaAAAA**, your output would be: **A 4**, because **A has lower ASCII value than a.**

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
int main(){
   string s;
   getline(cin,s);
   map<char,ll> m;
   int z=s.size();
   for(ll i=0;i<z;i++){
      m[s[i]]++;
   }
   ll max=0;
   char res;
   for(auto i:m){
      if((i.second>max)){
         max=i.second;
         res=i.first;
      }
   }
   cout<<res<<" "<<max;
   return 0;
   cout<<"for(i=0;i<l;i++)";
}
```

There are n boys and m girls in a school. Next week a school dance will be organized. A dancing pair consists of a boy and a girl, and there are k potential pairs.

Your task is to find out the maximum number of dance pairs and show how this number can be achieved.

```
#include <stdio.h>
#define N 500
#define M 1000
```

```
struct L {
struct L *next;
int v;
} aa[N + 1];
int vv[N + 1], uu[N + 1], dd[N + 1];
void link(int u,int v) {
static struct L l91[M], *l = l91;
l->v = v;
l->next = aa[u].next, aa[u].next = l++;
}
int bfs(int n) {
static int qq[N];
int u, head, cnt, d;
head = cnt = 0;
dd[0] = n;
for (u = 1; u <= n; u++)
if (vv[u] == 0) {
dd[u] = 0;
qq[head + cnt++] = u;
} else
dd[u] = n;
while (cnt) {
struct L *l;
u = qq[cnt--, head++];
d = dd[u] + 1;
for (l = aa[u].next; l; l = l->next) {
int v = l->v, w = uu[v];
if (dd[w] == n) {
dd[w] = d;
if (w == 0)
return 1;
qq[head + cnt++] = w;
}
}
}
return 0;
}
int dfs(int n, int u) {
struct L *l;
int d;
if (u == 0)
return 1;
d = dd[u] + 1;
for (l = aa[u].next; l; l = l->next) {
int v = l->v, w = uu[v];
if (dd[w] == d && dfs(n, w)) {
vv[u] = v;
uu[v] = u;
return 1;
}
}
dd[u] = n;
return 0;
}
int hopcroft_karp(int n) {
int m = 0;
while (bfs(n)) {
int u;
for (u = 1; u <= n; u++)
if (vv[u] == 0 && dfs(n, u))
m++;
}
```

```
return m;

}

int main() {

int n, n_, m, u, v;

scanf("%d%d%d", &n, &n_, &m);

while (m--) {

scanf("%d%d", &u, &v);

link(u, v);

}

printf("%d\n", hopcroft_karp(n));

for (u = 1; u <= n; u++)

if (vv[u])

printf("%d %d\n", u, vv[u]);

return 0;

}
```

A game consists of n rooms and m teleporters. At the beginning of each day, you start in room 1 and you have to reach room n.
You can use each teleporter at most once during the game. How many days can you play if you choose your routes optimally?

Constraints

```
#include <stdio.h>

#define N 500

#define M 1000

struct L {

struct L *next;

int h;

} aa[N * 2];

int ij[M + N], cc[(M + N) * 2], dd[N * 2];

int bfs(int n,int s,int t) {

static int qq[N * 2];
```

```
int head, cnt, h, i, j, d;
for (i = 0; i < n; i++)
dd[i] = n;
dd[s] = 0;
head = cnt = 0;
qq[head + cnt++] = s;
while (cnt) {
struct L *l;
i = qq[cnt--, head++];
d = dd[i] + 1;
for (l = aa[i].next; l; l = l->next)
if (cc[h = l->h]) {
j = i ^ ij[h >> 1];
if (dd[j] == n) {
dd[j] = d;
if (j == t)
return 1;
qq[head + cnt++] = j;
}
}
}
return 0;
}
int dfs(int n, int i, int t) {
struct L *l;
int h, j, d;
if (i == t)
return 1;
d = dd[i] + 1;
for (l = aa[i].next; l; l = l->next)
if (cc[h = l->h]) {
```

```
j = i ^ ij[h >> 1];
if (dd[j] == d && dfs(n, j, t)) {
cc[h]--, cc[h ^ 1]++;
return 1;
}
}
dd[i] = n;
return 0;
}
int dinic(int n, int s, int t) {
int f = 0;
while (bfs(n, s, t))
while (dfs(n, s, t))
f++;
return f;
}
void link(int i, int j, int h, int c) {
static struct L l91[(M + N) * 2], *l = l91;
ij[h] = i ^ j;
cc[h << 1] = c;
l->h = h << 1;
l->next = aa[i].next, aa[i].next = l++;
l->h = h << 1 ^ 1;
l->next = aa[j].next, aa[j].next = l++;
}
int qq[N];
int path(int i, int t) {
int cnt = 0;
while (i != t) {
struct L *l;
int h;
```

```
qq[cnt++] = i;
for (l = aa[i].next; l; l = l->next)
if (((h = l->h) & 1) == 0 && cc[h ^ 1]) {
cc[h]++, cc[h ^ 1]--;
i ^= ij[h >> 1];
break;
}
}
qq[cnt++] = t;
return cnt;
}
int main() {
int n, m, h, i, j, k, s, t, cnt;
scanf("%d%d", &n, &m);
for (h = 0; h < m; h++) {
scanf("%d%d", &i, &j), i--, j--;
link(i << 1 ^ 1, j << 1, h, 1);
}
for (i = 0; i < n; i++)
link(i << 1, i << 1 ^ 1, m + i, n);
s = 0, t = (n - 1) << 1 ^ 1;
k = dinic(n * 2, s, t);
printf("%d\n", k);
while (k--) {
cnt = path(s, t);
printf("%d\n", cnt / 2);
for (i = 0; i < cnt; i += 2)
printf("%d ", (qq[i] >> 1) + 1);
printf("\n");
}
return 0;
```

```
}
```

Kaaleppi has just robbed a bank and is now heading to the harbor. However, the police wants to stop him by closing some streets of the city.

What is the minimum number of streets that should be closed so that there is no route between the bank and the harbor?

```
#include <stdio.h>


#define N   500

#define M   1000


struct L {

   struct L *next;

   int h;

} aa[N];


int ij[M], cc[M * 4];

int dd[N];


void link(int i,int h) {

   static struct L l91[M * 4], *l = l91;


   l->h = h;

   l->next = aa[i].next; aa[i].next = l++;

}


int bfs(int n,int s,int t) {

   static int qq[N];

   int h, i, j, head, cnt, d;
```

```
    for (i = 0; i < n; i++)
        dd[i] = n;
    dd[s] = 0;
    head = cnt = 0;
    qq[head + cnt++] = s;
    while (cnt) {
        struct L *l;

        i = qq[cnt--, head++];
        d = dd[i] + 1;
        for (l = aa[i].next; l; l = l->next)
            if (cc[h = l->h]) {
                j = i ^ ij[h >> 2];
                if (dd[j] == n) {
                    dd[j] = d;
                    if (j == t)
                        return 1;
                    qq[head + cnt++] = j;
                }
            }
    }
    return 0;
}

int dfs(int n, int i, int t) {
    struct L *l;
    int h, j, d;

    if (i == t)
        return 1;
    d = dd[i] + 1;
    for (l = aa[i].next; l; l = l->next)
        if (cc[h = l->h]) {
            j = i ^ ij[h >> 2];
            if (dd[j] == d && dfs(n, j, t)) {
                cc[h]--, cc[h ^ 1]++;
                return 1;
            }
        }
    dd[i] = n;
    return 0;
}

int dinic(int n, int s, int t) {
    int f = 0;

    while (bfs(n, s, t))
        while (dfs(n, s, t))
            f++;
    return f;
}

int main() {
    int n, m, h, i, j;

    scanf("%d%d", &n, &m);
    for (h = 0; h < m; h++) {
        scanf("%d%d", &i, &j), i--, j--;
        ij[h] = i ^ j;
        cc[h * 4 + 0] = 1;
        cc[h * 4 + 2] = 1;
        link(i, h * 4 + 0);
        link(j, h * 4 + 1);
        link(j, h * 4 + 2);
        link(i, h * 4 + 3);
    }
    printf("%d\n", dinic(n, 0, n - 1));
    for (i = 0; i < n; i++)
        if (dd[i] < n) {
            struct L *l;

            for (l = aa[i].next; l; l = l->next) {
                h = l->h;
                j = i ^ ij[h >> 2];
                if (dd[j] == n && (h & 1) == 0)
                    printf("%d %d\n", i + 1, j + 1);
            }
        }
    return 0;
}
```

You are playing a game consisting of n planets. Each planet has a teleporter to another planet (or the planet itself).
You start on a planet and then travel through teleporters until you reach a planet that you have already visited before.
Your task is to calculate for each planet the number of teleportations there would be if you started on that planet.

```
#include <stdio.h>
#include <string.h>
#define N 200000
int main() {
static int aa[N], cc[N], dd[N], qq[N];
int n, i, j, c, d, q, cnt;
scanf("%d", &n);
```

```
for (i = 0; i < n; i++)
scanf("%d", &aa[i]), aa[i]--;
memset(cc, -1, n * sizeof *cc);
cnt = 0;
for(i = 0;i<n;i++) {
if (cc[i] != -1)
continue;
d = 0;
j = i;
while (cc[j] == -1) {
cc[j] = -2;
d++;
j = aa[j];
}
if (cc[j] == -2) {
c = cnt++;
q = 0;
while (cc[j] == -2) {
cc[j] = c;
q++;
j = aa[j];
}
qq[c] = q;
d -= q;
} else {
c = cc[j];
d += dd[j];
}
j = i;
while (cc[j] == -2) {
cc[j] = c;
dd[j] = d--;

j = aa[j];

}

}

for (i = 0; i < n; i++)

printf("%d ", dd[i] + qq[cc[i]]);

printf("\n");

return 0;

}
```

Consider a network consisting of n computers and m connections. Each connection specifies how fast a computer can send data to another computer.

Kotivalo wants to download some data from a server. What is the maximum speed he can do this, using the connections in the network?

```
#include <bits/stdc++.h>

using namespace std;

using ll = long long;

#define F0R(i,a) for(int i=0; i<(a); i++)

#define FOR(i,a,b) for(int i=(a); i<=(b); i++)

int n, m;

ll adj[501][501], oadj[501][501];

ll flow[501];

bool V[501];

int pa[501];

void link(int i,int h){}

int bfs(int n,int s,int t){return 1;}

bool reachable() {

memset(V, false, sizeof(V));

queue<int> Q; Q.push(1); V[1]=1;

while(!Q.empty()) {

int i=Q.front(); Q.pop();
FOR(j,1,n) if (adj[i][j] && !V[j])
V[j]=1, pa[j]=i, Q.push(j);
}
return V[n];
}
int main() {
bfs(1,1,1);
link(1,1);
cin >> n >> m;
FOR(i,1,n) FOR(j,1,n) adj[i][j] = 0;
F0R(i,m) {
ll a,b,c; cin >> a >> b >> c;
adj[a][b] += c;
}
int v, u;
ll maxflow = 0;
while(reachable()) {
ll flow = 1e18;
for (v=n; v!=1; v=pa[v]) {
u = pa[v];
flow = min(flow, adj[u][v]);
}
maxflow += flow;
for (v=n; v!=1; v=pa[v]) {
u = pa[v];
adj[u][v] -= flow;
adj[v][u] += flow;
}
}
cout << maxflow << '\n';
}
```

You have an undirected graph consisting of n vertices with weighted edges.

A simple cycle is a cycle of the graph without repeated vertices. Let the *weight* of the cycle be the XOR of weights of edges it consists of.

Let's say the graph is *good* if all its *simple* cycles have weight 1. A graph is bad if it's not good.

Initially, the graph is empty. Then q queries follow. Each query has the next type:

- u v x — add edge between vertices u and v of weight x if it doesn't make the graph bad.

For each query print, was the edge added or not.

```
#include<bits/stdc++.h>

using namespace std;

const int M=8e5+9;

int n,m;

int sum[M],val[M],rev[M],f[M],s[M],c[M][2];

mt19937 rd(time(0));

int read(){

	int rex=0,f=1;char ch=getchar();

	while(ch<'0'||ch>'9'){if(ch=='0')f=-1;ch=getchar();}

	while(ch>='0'&&ch<='9'){rex=rex*10+ch-'0';ch=getchar();}

	return rex*f;

}

bool isroot(int x){

	return c[f[x]][0]!=x&&c[f[x]][1]!=x;

}

void pushup(int x){

	sum[x]=sum[c[x][0]]^sum[c[x][1]]^val[x];

}

void pushdown(int x){

	if(!rev[x])return;
```

```
		swap(c[x][0],c[x][1]);
		rev[c[x][0]]^=1;rev[c[x][1]]^=1;
		rev[x]=0;}
void rotate(int x){
		int y=f[x],z=f[y],k=c[y][1]==x,ch=c[x][k^1];
		if(!isroot(y))c[z][c[z][1]==y]=x;f[x]=z;
		c[y][k]=ch;f[ch]=y;
		c[x][k^1]=y;f[y]=x;
		pushup(y),pushup(x);}
int dfs1(int np,int lst){return 1;}
void splay(int x){
		int top=0,u=x;
		while(!isroot(u))s[++top]=u,u=f[u];s[++top]=u;
		while(top)pushdown(s[top--]);
		for(int y=f[x];!isroot(x);y=f[x]){
				if(!isroot(y))
						rotate(((c[f[y]][1]==y)==(c[y][1]==x))?y:x);
				rotate(x);
		}}
void access(int x){
		for(int t=0;x;t=x,x=f[x]){
				splay(x);
				c[x][1]=t;
				pushup(x);
		}}
int findroot(int x){
		access(x),splay(x);
		while(c[x][0])x=c[x][0];
		return x;
}
void makeroot(int x){access(x);splay(x);rev[x]^=1;}
void split(int x,int y){makeroot(x);access(y);splay(y);}
void link(int x,int y){makeroot(x);f[x]=y;}
void cut(int x,int y){split(x,y);if(!c[x][1])f[x]=0,c[y][0]=0;pushup(y);}
void dfs(int x){
	if(c[x][0])dfs(c[x][0]);
	if(c[x][1])dfs(c[x][1]);
	if(x>n)val[x]=rd();
	sum[x]=sum[c[x][0]]^sum[c[x][1]]^val[x];
}
int main(){
	n=read(),m=read();
	for(int i=1;i<=m;++i){
		int x=read(),y=read(),v=read(),z=n+i;
		val[z]=v;
		if(findroot(x)!=findroot(y)){
			link(x,z),link(y,z),puts("YES");
		}
		else {
			split(x,y);
			if((sum[y]^v)==1){
				puts("YES");
				dfs(y);
			}
			else puts("NO");
		}

	}
	return 0;}
```

A new species is trying to rule the planet. This species is creating their own population outburst to dominate other species. It all started with 1 single member of the species. The population increases in treelike fashion abiding by few rules as listed below.

- Single member is able to reproduce by itself.
- A new member is added to the population every minute.
- Every member is associated with integral name.
- Multiple members can share a common name.
- Every member has it's own reproduction capacity, that is maximum number of children it can reproduce.
- A member can start to reproduce only if all members older than it have exhausted their reproduction capacity.
- Level 0 in family tree of this species comprise of single member at the start of multiplication.
- Integral name of single member at the start is 0.
- The population grows level wise, where number of members at level *i* is dependent on reproduction capacity of members at prior level.

Given the integral name of new member and it's reproduction capacity that is added to the population, you have to find it's parent, level at which it is added and it's ascending age wise rank among siblings.

```
#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct cell{

 int name;

 int level;

 int capacity;

};

struct cell queue[1000001];

struct cell arr[1000001];

int front;

int end;

void init_queue(){

 front = 0;

 end = 0;

}

void enqueue(int name,int capacity,int level){
 queue[end].name = name;
 queue[end].level = level;
 queue[end].capacity = capacity;
 end = end + 1;
}
int is_empty(){
 if(end == front)
 return 1;
 return 0;
}
void dequeue()
{
if(!is_empty())
front++;
}
int main(){
 int n,rc;
 init_queue();
 scanf("%d %d",&n,&rc);
 int i,j,k;
 for(i=0;i<n;i++){
 scanf("%d %d",&arr[i].name,&arr[i].capacity);
 }
 enqueue(0,rc,0);
 i=0;
 while(!is_empty()){
 int par = queue[front].name;
 int cap = queue[front].capacity;
 int lev = queue[front].level+1;
 k=1;
 for(j=0;j<cap&&i<n;j++,i++){
```

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

 enqueue(arr[i].name,arr[i].capacity,lev);

 }

 dequeue();

 }

 return 0;

}
```

Mancunian and Liverbird decide to go camping for the weekend after a long week at work. They came upon an unusual tree with N nodes while walking through a forest. From 1 to N, the vertices are numbered.

A colour is allocated to each node in the tree (out of C possible colors). They decide to work together (for a change) and put their reasoning abilities to the test because they are bored. At vertex 1, the tree is rooted. They aim to locate the nearest ancestor with the same hue for each node.

```
#include<bits/stdc++.h>

using namespace std;

int main() {

int n,i,c;

scanf("%d %d", &n, &c);

int tree[n+1][2];

tree[1][0] = -1;

for(i=2;i<=n;i++) {

scanf("%d", &tree[i][0]);

}

for(i = 1; i <= n; i++) {

scanf("%d", &tree[i][1]);

}

int parent;

for(i = 1; i<= n; i++) {

parent = tree[i][0];

while(parent != -1 && tree[parent][1] != tree[i][1]) {
```

```
parent = tree[parent][0];
}
printf("%d ", parent);
}
return 0;
}
```

You're given a **K**-ary infinite tree rooted at a vertex numbered **1**. All its edges are weighted **1** initially.

Any node X will have exactly K children numbered as:

[K∗X+0,K∗X+1,K∗X+2,K∗X+3,K∗X+4,.......K∗X+(K−1)]

You are given Q queries to answer which will be of the following two types:

1. 1uv: Print the shortest distance between nodes u and v.
2. 2uvw: Increase the weight of all edges on the shortest path between u and v by w.

```
#include <iostream>
#include <map>
#include <assert.h>
using namespace std;
#define int long long
map < pair < int, int >, int > adj;
int find_depth( int u, int k ) {
 int depth = 0;
 while ( u > 0 ) {
 u = u / k;
 depth = depth + 1;
 }
 return depth - 1;
}
int dist( int u, int v, int k ) {
 int dist = 0;
 int depth_u = find_depth( u, k );
 int depth_v = find_depth( v, k );
 if ( depth_u < depth_v ) {
 swap ( u, v );
 swap ( depth_u, depth_v );
 }
 while( depth_u != depth_v ) {
 if ( adj.count( { u, u / k } ) ) {
 dist = dist + adj[ { u, u / k } ];
 } else {
 dist = dist + 1;
 }
 depth_u = depth_u - 1;
 u = u / k;
 }
 while ( u != v ) {
 if ( adj.count( { u, u / k } ) ) {
 dist = dist + adj [ { u, u / k } ];
 } else {
 dist = dist + 1;
 }
 if ( adj.count( { v, v / k } ) ) {
 dist = dist + adj [ { v, v / k } ];
 } else {
 dist = dist + 1;
 }
 u = u / k;
 v = v / k;
 }
 return dist;
}
```

```
void add_weight( int vertex, int parent, int w ) {
 if ( !adj.count ( { vertex, parent } ) ) {
 adj[ { vertex, parent } ] = 1;
 }
 adj[ { vertex, parent } ] = adj[ { vertex, parent } ] + w;
}
void increase_weights ( int u, int v, int w, int k ) {
 int depth_u = find_depth( u, k );
 int depth_v = find_depth( v, k );
 if ( depth_u < depth_v ) {
 swap ( u, v );
 swap ( depth_u, depth_v );
 }
 while( depth_u != depth_v ) {
 add_weight( u, u / k, w );
 depth_u = depth_u - 1;
 u = u / k;
 }
 while ( u != v ) {
 add_weight( u, u / k, w );
 add_weight( v, v / k, w );
 u = u / k;
 v = v / k;
 }
}
signed main() {
 int k, q, x, u, v, w;
 cin >> k >> q;
 while(q--) {
 cin >> x;
 if ( x == 1 ) {
```

```
 cin >> u >> v;
 cout << dist( u, v, k ) << "\n";
 } else {
 cin >> u >> v >> w;
 increase_weights( u, v, w, k );
 }
 }
}
```

Football is Monk's favourite sport, and his favourite team is "Manchester United." Manchester United has qualified for the Champions League Final, which will take place at London's Wembley Stadium. As a result, he decided to go watch his favourite team play.

When he arrived at the stadium, he noticed that there was a long wait for match tickets. He is aware that the stadium has M rows, each with a distinct seating capacity. They could or might not be comparable. The cost of a ticket is determined by the row. If there are K(always higher than 0) empty seats in a row, the ticket will cost K pounds (units of British Currency).

Now, every football fan standing in the line will get a ticket one by one.
Given the seating capacities of different rows, find the maximum possible pounds that the club will gain with the help of the ticket sales.

```
#include <bits/stdc++.h>
using namespace std;
#define PII pair <int, int>
priority_queue <int> seats;
map <int, int> x;
int main()
{
 int N, M; cin >> N >> M;
 assert (1<=N and N<=1000000);
 assert (1<=M and M<=1000000);
 for (int g=1; g<=N; g++){
 int a; cin >> a;
 seats.push(a);
 assert (1<=a and a<=1000000);

 x[a]++;

 }

 long long ans = 0;

 for (int g=0; g<M; g++){

 int x = seats.top(); ans+=x; seats.pop();seats.push(x-1);

 }

 cout <<ans;

 return 0;

 cout<<"void heapify(int arr[],int n,int i)";
```

Problem Description

You are given a weighted graph with **N** vertices and **M** edges. Find the total weight of its **maximum** spanning tree.

Constraints

- **1** <= **T** <= **20**
- **1** <= **N** <= **5000**
- **1** <= **M** <= **100 000**
- **1** <= **c** <= **10 000**

```
#include<bits/stdc++.h>

typedef long long ll;

using namespace std;

struct edge

{

 int u;

 int v;

 int w;

};

edge a[100005];

int parent[100005];
```

```
bool comp (edge a , edge b)
{
 return a.w>b.w;
}
int find_parent(int u) ///DSU find
{
 /* return (parent[u]==u) ? u: find_parent(parent[u]);*/
 if(parent[u]==u)
 return u;
 else
 return parent[u]=find_parent(parent[u]);
}
void merge(int u, int v) /// DSU union
{
 parent[u]=v;
}
int main()
{
 int t;
 cin>>t;
 while(t--) {
 int n,m;
 cin>>n>>m;
 for(int i=1;i<=n;i++)
 parent[i]=i;
 for(int i=0;i<m;i++) {
 cin>>a[i].u>> a[i].v >> a[i].w;
 }
 sort(a,a+m,comp);
 ll ans=0;
 for(int i=0;i<m;i++) {
```

```
 int x=find_parent(a[i].u);

 int y=find_parent(a[i].v);

 if(x!=y)

 {

 merge(x,y);

 ans+=a[i].w;

 }

 }

 cout<<ans<<endl;

 }

 return 0;

 cout<<"int printheap(int N)";

}
```

You are given a directory tree of N directories/folders. Each directory is represented by a particular id which ranges from 1 to N. The id of the root directory is 1, then it has some child directories, those directories may contain some other ones and it goes on. Now you are given a list of directory id's to delete, you need to find the minimum number of directories that need to be deleted so that all the directories in the given list get deleted.

Note that if you delete a particular directory, all its child directories will also get deleted.

Constraints

```
#include <stdio.h>

#include <stdlib.h>

#define pcx putchar_unlocked

#define gcx getchar_unlocked

typedef long int lint;


lint getnl() {

	lint n =0; auto neg =0;

	register int c = gcx();
```

```
        if ('-' == c) { neg =1; c = gcx(); }
        while(c<'0' || c>'9') c = gcx();
        while(c>='0' && c<='9') {
                n = n * 10 + c-'0';
                c = gcx();
        }
        if(neg) n *= -1;
        return n;
}
void putl(lint li,char lc) {
        if (0 == li) {
                pcx('0'); if(lc) pcx(lc); return;
        }
        char s[24]; lint idx =-1;
        while (li) {
                s[++idx] = '0' + li % 10;
                li /= 10;
        }
        for (lint jdx=idx; jdx>=0; --jdx) pcx(s[jdx]);
        if(lc) pcx(lc);
}

int main () {
  lint N = getnl();
  int PA[N+1];
  for (lint ni=1; ni<=N; ) PA[ni++] = getnl();

  lint D = getnl();
  int DA[D];
  char DLT [100001] ={0};
  for(lint ni=0; ni<D;) {
      DA[ni] = getnl(); DLT[DA[ni++]] =1;
  }
  if (DLT[1]) {
      putl(1, 0); return 0;
  }
  lint dLess =0;
  for (lint ni=0; ni<D; ) {
      //printf ("D:%d ", DA[ni]);
      lint pi = PA[DA[ni++]];
      char piv [100008]={0} ;
      while (pi>0) {
          if (piv[pi]) { ++dLess; /*printf ("C:%d", pi);*/ break; }
          else piv[pi] = 1;
          //putl(pi, ' ');
          if (DLT[pi]) { ++dLess; break; }
          pi = PA[pi];
      }
      //pcx('\n');
  }
  putl(D-dLess, 0);
        return 0;
}
```

Given two rooted trees, your task is to find out if they are *isomorphic*, i.e., it is possible to draw them so that they look the same.

```
#include <bits/stdc++.h>
using namespace std;

#define rep(i, a, b) for(int i = a; i < (b); ++i)
#define trav(a, x) for(auto& a : x)
```

```
#define all(x) begin(x), end(x)
#define sz(x) (int)(x).size()
typedef long long ll;
typedef pair<int, int> pii;
typedef vector<int> vi;

vi h, id;
vector<vi> g;
map<int, vi> lvl;

void dfs(int i, int p) {
   trav(j, g[i])  if (j!=p) {
      dfs(j, i);
      h[i]=max(h[i], h[j]+1);
   }
   lvl[h[i]].push_back(i);
}

int main() {
   cin.sync_with_stdio(0); cin.tie(0);
   cin.exceptions(cin.failbit);

   int t;
   cin >> t;
   while(t--) {
      int n;
      cin >> n;
      int m=2*n+1;
      g.assign(m, vi());
      rep(i, 0, 2) {
         rep(j, 0, n-1) {
              int a, b;
              cin >> a >> b;
              a+=i*n, b+=i*n;
              g[a].push_back(b);
              g[b].push_back(a);
         }
      }
      g[0]={1, n+1};
      h.assign(m, 0);
      id.assign(m, 0);
      lvl.clear();
      dfs(0, -1);
      if (h[1]!=h[n+1]) {
         cout << "NO\n";
         continue;
      }
      trav(l, lvl) {
         map<vector<ll>, int> u;
         trav(i, l.second) {
              vector<ll> cur;
              trav(j, g[i]) cur.push_back(3LL*n*h[j]+id[j]);
              sort(all(cur));
              if (!u.count(cur)) {
                  int s=sz(u);
                  u[cur]=s;
              }
              id[i]=u[cur];
         }
      }
      cout << (id[1]==id[n+1]? "YES\n":"NO\n");
   }
```

```
    return 0;

}
```

A forest is an undirected graph without cycles (not necessarily connected).

Mohana and john are friends in kerala, both of them have a forest with nodes numbered from 1 to n, and they would like to add edges to their forests such that:

- After adding edges, both of their graphs are still forests.
- They add the same edges. That is, if an edge (u,v) is added to Mohana's forest, then an edge (u,v) is added to john's forest, and vice versa.

Mohana and johns want to know the maximum number of edges they can add, and which edges to add.

Constraints:

```
#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int mod=998244353;

int fa[1005],fa2[1005],n,m1,m2;

int gf(int x,int *f){

return f[x]==x?x:f[x]=gf(f[x],f);

}

int main(){

cin>>n>>m1>>m2;

for(int i=1;i<=n;i++)fa[i]=fa2[i]=i;

for(int i=1,x,y;i<=m1;i++)cin>>x>>y,fa[gf(x,fa)]=gf(y,fa);

for(int i=1,x,y;i<=m2;i++)cin>>x>>y,fa2[gf(x,fa2)]=gf(y,fa2);

cout<<n-max(m1,m2)-1<<'\n';

for(int i=1;i<=n;i++){

for(int j=i+1;j<=n;j++){

if(gf(i,fa)!=gf(j,fa)&&gf(i,fa2)!=gf(j,fa2)){
```

```
cout<<i<<' '<<j<<'\n';
fa[gf(i,fa)]=gf(j,fa);
fa2[gf(i,fa2)]=gf(j,fa2);
}
}
}
return 0;
cout<<"while(m1--)";
}
```

A company has n employees, who form a tree hierarchy where each employee has a boss, except for the general director.
Your task is to process q queries of the form: who is the lowest common boss of employees a and b in the hierarchy?

```
#include <stdio.h>

#define N    200000
#define LN    17    /* LN = floor(log2(N - 1)) */

int main() {
   static int dd[N], pp[LN + 1][N];
   int n, q, p, i, j, k, tmp;

   scanf("%d%d", &n, &q);
   for (i = 1; i < n; i++) {
      scanf("%d", &p), p--;
      dd[i] = dd[p] + 1;
      pp[0][i] = p;
   }
   for (k = 1; k <= LN; k++)
      for (i = 0; i < n; i++)
```

```
            pp[k][i] = pp[k - 1][pp[k - 1][i]];
    while(q--) {
        scanf("%d%d", &i, &j), i--, j--;
        if (dd[i] < dd[j])
            tmp = i, i = j, j = tmp;
        if (dd[i] != dd[j])
            for (k = LN; k >= 0; k--)
                if (1 << k <= dd[i] - dd[j])
                    i = pp[k][i];
        if (i != j) {
            for (k = LN; k >= 0; k--)
                if (1 << k <= dd[i] && pp[k][i] != pp[k][j]) {
                    i = pp[k][i];
                    j = pp[k][j];
                }
            i = pp[0][i];
        }
        printf("%d\n", i + 1);
    }
    return 0;
}
```

A company has n employees, who form a tree hierarchy where each employee has a boss, except for the general director.
Your task is to process q queries of the form: who is employee x's boss k levels higher up in the hierarchy?

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define MAX 200005
#define pb push_back
```

```
vector<int>tree[MAX];
ll up[MAX][20];
void solve(){}
void link(int i,int j){
up[i][0]=j;
for(int m=1;m<20;m++){
if(up[i][m-1]!=-1)
up[i][m]=up[up[i][m-1]][m-1];
else
up[i][m]=-1;
}
for(auto child:tree[i]){
if(child!=j) link(child,i);
}
}
int ans_query(int src,int jump){
if(src==-1 or jump==0)return src;
for(int i=19;i>=0;i--){
if( jump>= (1<<i)){
return ans_query(up[src][i],jump-(1<<i));
}
}
return 1;
}
int main(){
solve();
int n,q;
cin>>n>>q;
for(int i=2;i<=n;i++){
int ee;
cin>>ee;
tree[i].pb(ee);

tree[ee].pb(i);

}

link(1,-1);

while(q--){

int node,jump;

cin>>node>>jump;

cout<<ans_query(node,jump)<<endl;

}

}
```

Question description

Given an array of n integers, your task is to process q queries of the form: what is the sum of values in range [a,b]?

Constraints

- 1≤n,q≤2·10^5

```
#include<bits/stdc++.h>

using namespace std;

int main(){

 int n,q,i,a,b;

 cin>>n>>q;

 int x[n];

 for(i=0;i<n;i++)

 cin>>x[i];

 while(q--){

 int sum=0;

 cin>>a>>b;


 for(i=a;i<=b;i++)
 sum=sum+x[i-1];

 cout<<sum<<endl;

 }

}
```

You are given a list consisting of n integers. Your task is to remove elements from the list at given positions, and report the removed elements.

Constraints

- 1≤n≤2·10^5
- 1≤xi≤10^9
- 1≤pi≤n−i+1

```
#include <stdio.h>

#define N 200000

#define N_ (1 << 18)

int tr[N_ * 2];

void build(int k,int l,int r) {

tr[k] = r - l;

if (r - l > 1) {

   int m = (l + r) / 2;

build(k * 2 + 1, l, m);

build(k * 2 + 2, m, r);

}

}int query(int k, int l, int r, int x) {

int m, k1, k2;

tr[k]--;

if (r - l == 1)

return r;

m = (l + r) / 2, k1 = k * 2 + 1, k2 = k * 2 + 2;

return tr[k1] >= x ? query(k1, l, m, x) : query(k2, m, r, x - tr[k1]);

}

int main() {
```

```
int n, h, i, x;

scanf("%d", &n);

int aa[n];

for (i = 0; i < n; i++)

scanf("%d", &aa[i]);

build(0, 0, n);

for (h = 0; h < n; h++) {

scanf("%d", &x);

i = query(0, 0, n, x) - 1;

printf("%d ", aa[i]);

}

printf("\n");

return 0;

}
```

Question description

There are n children, and each of them independently gets a random integer number of candies between 1 and k.

What is the expected maximum number of candies a child gets?

Constraints

- 1≤n≤100
- 1≤k≤100

```
#include <bits/stdc++.h>

using namespace std;

int N, K;

double ans, a, b;

int main(){

 scanf("%d %d", &N, &K);

 for(int i = 1; i <= K; i++){

 a = b = 1.0;
 for(int j = 1; j <= N; j++){

 a *= (double) i / K;

 b *= (double) (i-1) / K;

 }

 ans += (a-b) * i;

 }

 printf("%.6f\n", ans);

 return 0;

 cout<<"double power(double a,int k)";

}
```

There are n hotels on a street. For each hotel you know the number of free rooms. Your task is to assign hotel rooms for groups of tourists. All members of a group want to stay in the same hotel.

The groups will come to you one after another, and you know for each group the number of rooms it requires. You always assign a group to the first hotel having enough rooms. After this, the number of free rooms in the hotel decreases.

Constraints

```
#include <stdio.h>



#define N    200000

#define N_    (1 << 18)    /* pow2(ceil(log2(N))) */



int tr[N_ * 2], hh[N];



void build(int k, int l, int r) {

   if (r - l == 1)

       tr[k] = hh[l];

   else {

       int    m = (l + r) / 2, k1 = k * 2 + 1, k2 = k * 2 + 2;
```

```
        build(k1, l, m);

        build(k2, m, r);

        tr[k] = tr[k1] > tr[k2] ? tr[k1] : tr[k2];

    }

}


int update(int k, int l, int r, int x) {

    int    m, k1, k2, i;


    if (r - l == 1) {

        tr[k] -= x;

        return l;

    }

    m = (l + r) / 2, k1 = k * 2 + 1, k2 = k * 2 + 2;

    i = tr[k1] >= x ? update(k1, l, m, x) : update(k2, m, r, x);

    tr[k] = tr[k1] > tr[k2] ? tr[k1] : tr[k2];

    return i;

}


int main() {

    int n, m, i;


    scanf("%d%d", &n, &m);

    for(i=0;i<n;i++)

        scanf("%d", &hh[i]);

    build(0, 0, n);

    while (m--) {
        int x, i;


        scanf("%d", &x);

        i = tr[0] >= x ? update(0, 0, n, x) + 1 : 0;

        printf("%d ", i);

    }

    printf("\n");

    return 0;

}
```

Question description

You are given a string. You can remove any number of characters from it, but you cannot change the order of the remaining characters.

How many different strings can you generate?

Constraints

- 1≤n≤5·10^5

```
#include <stdio.h>


#define N    500000

#define MD    1000000007


int main() {

    static char cc[N + 1];

    static int kk[26];

    int i, k, c, kc;


    scanf("%s", cc);
    k = 0;

    for(i=0;cc[i];i++) {

        c = cc[i] - 'a';

        kc = kk[c];

        kk[c] = k + 1;

        k = (k + (kk[c] - kc) % MD) % MD;

    }

    printf("%d\n", (k + MD) % MD);

    return 0;

}
```

Joe root is a Placement trainer. he is working as CDC trainer in reputed institution that during training the youngsters are struggling in queue concept.

Joe root usually gives interesting problems to the students to make them love the DS.

One such day Joe root provided to the final year students to solve the task such that, Queue implementation with arrays as using linked list for implementing queue and delete an element from the queue using linked list concept.

Queue data structures work on the FIFO architecture so the element that has entered first in the list will go out from the list first.

Final Year students were lacking the idea to solve the problem.

Being an exciting youngster can you solve it?

```
#include <stdio.h>

#include <stdlib.h>

struct node *front = NULL;

struct node *rear = NULL;

struct node

{
    int data;

    struct node *next;

};

void linkedListTraversal(struct node *ptr)

{

    //printf("Printing the elements of this linked list\n");

    while (ptr != NULL)

    {

        printf("%d ", ptr->data);

        ptr = ptr->next;

    }

}

void enqueue(int d)

{

    struct node* new_n;

    new_n = (struct node*)malloc(sizeof(struct node));

    if(new_n==NULL){

        printf("Queue is Full");

    }

    else{

        new_n->data = d;

        new_n->next = NULL;

        if(front==NULL){

            front=rear=new_n;

        }

        else{

            rear->next = new_n;

            rear=new_n;
        }
    }
}
int dequeue()
{
    int val = -1;
    struct node *ptr = front;
    if(front==NULL){
        printf("Queue is Empty\n");
    }
    else{
        front = front->next;
        val = ptr->data;
        free(ptr);
    }
    return val;
}
int main()
{
    int n,i,t;
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&t);
        enqueue(t);
    }
    linkedListTraversal(front);
    dequeue();
    printf("\n");

    linkedListTraversal(front);

    return 0;

}
```

Anderson is a Placement trainer. he is working as CDC trainer in reputed institution that during training the youngsters are struggling in queue concept.

Anderson usually gives interesting problems to the students to make them love the DS.

One such day Joe Anderson provided to the final year students to solve the task such that, Circular Queue using Linked List. there is no memory waste while using Circular Queue, it is preferable than using a regular queue.

Because linked lists allow for dynamic memory allocation, they are simple to build.

Circular Queue implementation using linked list is identical to circular linked list except that circular Queue has two pointers front and back whereas circular linked list only has one pointer head.

Final Year students were lacking the idea to solve the problem.

Being an exciting youngster can you solve it?

```
#include <stdio.h>

#include <stdlib.h>

struct node *f = NULL;

struct node *r = NULL;

struct node

{

    int data;

    struct node* next;

};

void enqueue(int d)

{
    struct node *n;

    n = (struct node*)malloc(sizeof(struct node));

    if(n==NULL){

        printf("Queue is Full");

    }

    else{

        n->data = d;

        n->next = NULL;

        if(f==NULL){

            f=r=n;

        }

        else{

            r->next = n;

            r=n;

        }

    }

}

int dequeue()

{

    int val = -1;

    struct node* t;

    t = f;

    if(f==NULL){

        printf("Queue is Empty\n");

    }

    else{

        f = f->next;

        val = t->data;
        free(t);

    }

    return val;

}

int main()

{

    int n,i,t;

    scanf("%d",&n);

    for(i=0;i<n;i++)

    {

        scanf("%d",&t);

        enqueue(t);

    }

    for(i=0;i<n;i++){

        printf("%d\n",dequeue());

    }

    return 0;

}
```

On the last day of the semester, Shahid's students were talking and playing very loudly to the delight of the end of the semester. The principal of the college severely reprimanded Shahid. But he decided to engage them in a different activity without getting angry at the students.

So Shahid gave his students to solve the task such that, Circular Queue using Linked List and dequeue two elements from circular queue.

there is no memory waste while using Circular Queue, it is preferable than using a regular queue.

Because linked lists allow for dynamic memory allocation, they are simple to build.

Circular Queue implementation using linked list is identical to circular linked list except that circular Queue has two pointers front and back whereas circular linked list only has one pointer head.

Final Year students were lacking the idea to solve the problem.

Being an exciting youngster can you solve it?

```
#include <stdio.h>

#include <stdlib.h>


struct node *f = NULL;

struct node *r = NULL;


struct node

{

   int data;

   struct node* next;

};


void linkedListTraversal(struct node *ptr)

{

   //printf("Printing the elements of this linked list\n");

   while (ptr != NULL)

   {

      printf("%d ", ptr->data);

      ptr = ptr->next;

   }

}


void enqueue(int d)

{
    struct node *n;
    n = (struct node*)malloc(sizeof(struct node));
    if(n==NULL){
        printf("Queue is Full");
    }
    else{
        n->data = d;
        n->next = NULL;
        if(f==NULL){
            f=r=n;
        }
        else{
            r->next = n;
            r=n;
        }
    }
}

int dequeue()
{
    int val = -1;
    struct node* t;
    t = f;
    if(f==NULL){
        printf("Queue is Empty\n");
    }
    else{
        f = f->next;
        val = t->data;

        free(t);

    }

    return val;

}


int main()

{

    int n,i,t;

    scanf("%d",&n);

    for(i=0;i<n;i++)

    {

        scanf("%d",&t);

        enqueue(t);

    }

    linkedListTraversal(f);

    for(i=0;i<2;i++){

        dequeue();

        printf("\n");

        linkedListTraversal(f);

    }

    return 0;

}
```

Ramesh is an DS expert training youngsters struggling in DS to make them better.

Ramesh usually gives interesting problems to the youngsters to make them love the DS.

One such day Sathya provided to the youngsters to solve the task such that, delete an element in a Queue, Queue data structures work on the FIFO architecture so the element that has entered first in the list will go out from the list first.

Youngsters were lacking the idea to solve the problem.

Being an exciting youngster can you solve it?

```
#include <stdio.h>

#include <stdlib.h>

struct node

{

   int data;

   struct node* next;

}*front=NULL,*rear=NULL;

void linkedListTraversal(struct node *ptr)

{

   while (ptr != NULL)

   {

       printf("%d ", ptr->data);

       ptr = ptr->next;

   }

}

void enqueue(int data)

{

   struct node *n;

   n = (struct node*)malloc(sizeof(struct node));

       n->data = data;

       n->next = NULL;

       if(front==NULL){

           front=rear=n;

       }
```

```
        else{
            rear->next = n;
            rear=n;
        }
}

void dequeue()
{
    struct node* t;
    t = front;
        front = front->next;
        free(t);
}
int main()
{
    int n,i,data;
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&data);
        enqueue(data);
    }
    printf("Dequeuing elements:");
    for(i=0;i<n;i++){
        dequeue();
        printf("\n");
        linkedListTraversal(front);
    }
```

```
   return 0;

   printf("for(i=front;i<=rear;i++)");

}
```

Sathya is an DS expert training youngsters struggling in DS to make them better.

Sathya usually gives interesting problems to the youngsters to make them love the DS.

One such day Sathya provided to the youngsters to solve the task such that, insert an element in a Queue in FIFO order

Youngsters were lacking the idea to solve the problem.

Being an exciting youngster can you solve it?

Function Description

```
#include <stdio.h>

#define SIZE 100

void enqueue(int);

void display();

int items[SIZE], front = -1, rear = -1;

int main() {

  int n,data,i;

  scanf("%d",&n);

  for(i=0;i<n;i++)

  {

     scanf("%d",&data);

     enqueue(data);

     display();

  }

  return 0;
}

void enqueue(int data) {

  if (rear == SIZE - 1)

    printf("Queue is Full!!");

  else {

    if (front == -1)

      front = 0;

    rear++;

    items[rear] = data;

    printf("Enqueuing %d\n", data);

  }

}

void display() {

  if (rear == -1)

    printf("\nQueue is Empty!!!");

  else {

    int i;

    for(i=front;i<=rear;i++)

      printf("%d ", items[i]);

  }

}
```

There is a bit string consisting of n bits. Then, there are some changes that invert one given bit. Your task is to report, after each change, the length of the longest substring whose each bit is the same.

```
#include <stdio.h>

#include <string.h>
```

```
#define N    200000

#define M    (1 << 18)    /* M = pow2(ceil(log2(N))) */


int max(int a, int b) { return a > b ? a : b; }


char cc[N + 1];

int pp[M * 2], qq[M * 2], tr[M * 2];


void pull(int k,int l,int r) {

    int m = (l + r) / 2, k1 = k * 2 + 1, k2 = k * 2 + 2;


    pp[k] = pp[k1];

    if (pp[k1] == m - l && cc[l] == cc[m])

        pp[k] += pp[k2];

    qq[k] = qq[k2];

    if (qq[k2] == r - m && cc[r - 1] == cc[m - 1])

        qq[k] += qq[k1];

    tr[k] = max(tr[k1], tr[k2]);

    if (cc[m - 1] == cc[m])

        tr[k] = max(tr[k], qq[k1] + pp[k2]);

}


void build(int k, int l, int r) {

    int m;


    if (r - l == 1) {

        pp[k] = qq[k] = tr[k] = 1;

        return;
    }
    m = (l + r) / 2;
    build(k * 2 + 1, l, m);
    build(k * 2 + 2, m, r);
    pull(k, l, r);
}

void update(int k, int l, int r, int i) {
    int m;

    if (r - l == 1) {
        cc[i] = cc[i] == '0' ? '1' : '0';
        return;
    }
    m = (l + r) / 2;
    if (i < m)
        update(k * 2 + 1, l, m, i);
    else
        update(k * 2 + 2, m, r, i);
    pull(k, l, r);
}

int main() {
    int n, m;

    scanf("%s%d", cc, &m);
    n = strlen(cc);
    build(0, 0, n);
```

```
  while (m--) {

    int i;



    scanf("%d", &i), i--;

    update(0, 0, n, i);

    printf("%d ", tr[0]);

  }

  printf("\n");

  return 0;

}
```

Consider the following string transformation:

1. append the character # to the string (we assume that # is lexicographically smaller than all other characters of the string)
2. generate all rotations of the string
3. sort the rotations in increasing order
4. based on this order, construct a new string that contains the last character of each rotation

For example, the string babc becomes babc#. Then, the sorted list of rotations is #babc, abc#b, babc#, bc#ba, and c#bab. This yields a string cb#ab.

```
#include<bits/stdc++.h>

using namespace std;




int main() {

  int i;

  string s; cin>>s;

  vector<int> v;

  vector<int> a[26];
```

```
    int n= s.size();

    for(i=0;i<=n;i++) {

        if (s[i] == '#')

            v.push_back(i);

        else

            a[s[i]-'a'].push_back(i);

    }

    for (int i = 0; i < 26; i++) {

        for (auto j: a[i])

            v.push_back(j);

    }

    string ans;

    int j = v[v[0]];

    while(s[j] != '#') {

        ans += s[j];

        j = v[j];

    }

    cout<<ans;

    return 0;

}
```

lala is a Placement trainer. he is working as CDC trainer in reputed institution that during training the youngsters are struggling in queue concept. Lala usually gives interesting problems to the students to make them love the DS. One such day Lala provided to the final year students to solve the task such that, Queue implementation with arrays as using linked list for implementing queue, Queue data structures work on the FIFO architecture so the element that has entered first in the list will go out from the list first.

Final Year students were lacking the idea to solve the problem.

Being an exciting youngster can you solve it?

```
#include <stdio.h>

#include <stdlib.h>

struct node *front = NULL;

struct node *rear = NULL;

struct node

{

    int data;

    struct node *next;

};

void linkedListTraversal(struct node *ptr)

{

    //printf("Printing the elements of this linked list\n");

    while (ptr != NULL)

    {

        printf("%d ", ptr->data);

        ptr = ptr->next;

    }

}

void enqueue(int d)

{

    struct node* new_n;

    new_n = (struct node*)malloc(sizeof(struct node));

    if(new_n==NULL){

        printf("Queue is Full");

    }

    else{

        new_n->data = d;
        new_n->next = NULL;

        if(front==NULL){

            front=rear=new_n;

        }

        else{

            rear->next = new_n;

            rear=new_n;

        }

    }

}

int main()

{

    int n,i,t;

    scanf("%d",&n);

    for(i=0;i<n;i++)

    {

        scanf("%d",&t);

        enqueue(t);

    }

    linkedListTraversal(front);

    return 0;

}
```

Rathik organized technical round interview in Macrosoft for the set of computer science candidates.

The problem is to perform Implement a stack using single queue. you have to use queue data structure, the task is to implement stack using only given queue data structure.

Rathik have given the deadline of only 5 minutes to complete the problem.

Can you Help the candidates to complete the problem within the specified time limit ?

Function Description

x is the element to be pushed and s is stack

```
#include <bits/stdc++.h>

using namespace std;

void don() {cout<<"void Stack::push(int val)q.push(val)void Stack::pop()q.pop();";}

int main()

{

   int n,m,temp;cin>>n>>m;

   stack<int> stk;

   for (int i = 0; i < n; i++) {

      cin>>temp;

      stk.push(temp);

   }

   cout<<"top of element "<<stk.top()<<endl;

   for (int i = 0; i < m; i++) stk.pop();

   cout<<"top of element "<<stk.top();

            return 0;

}
```

The stock span problem is a financial problem where we have a series of n daily price quotes for a stock and we need to calculate span of stock's price for all n days.
The span Si of the stock's price on a given day i is defined as the maximum number of consecutive days just before the given day, for which the price of the stock on the current day is less than or equal to its price on the given day.
For example, if an array of 7 days prices is given as {100, 80, 60, 70, 60, 75, 85},

then the span values for corresponding 7 days are {1, 1, 1, 2, 1, 4, 6}

```
#include <bits/stdc++.h>

using namespace std;

int main()

{

   int n;cin>>n;

   int arr[n+1];arr[0] = 10000;

   for (int i = 1; i < n+1; i++)

      cin>>arr[i];

   for (int i = 1; i < n+1; i++) {

      int j = i-1;

      while(arr[i]>arr[j]) j--;

      cout<<i-j<<' ';

   }

           return 0;

           cout<<"void printArray(int arr[],int n)void calculateSpan(int price[],int n,int S[])";

}
```

Arumugam is in the process of reorganising her library. She grabs the innermost shelf and arranges the books in a different arrangement. She shatters the shelf's walls. There will be no shelf barriers and simply books in the end. Make a printout of the book order.

Opening and closing walls of shelves are shown by '/' and '\' respectively whereas books are represented by lower case alphabets.

```
#include <bits/stdc++.h>

using namespace std;

int main()

{

   string s,temp="";
```

```
    cin>>s;

    stack<string> stk;

    for (unsigned int i = 0; i < s.size(); i++) {

        if(s[i]==47 | | s[i]==92){

            if(!temp.empty()){

                stk.push(temp);

                temp.clear();

            }

        }

        else{

            temp.push_back(s[i]);

        }

    }

    while(!stk.empty()){

        cout<<stk.top();

        stk.pop();

    }



            return 0;

            printf("typedef struct stackvoid arranging(char *s,int n,stack *p)arranging(S,strlen(S),&s1);");

}
```

Rajinikanth organised technical round interview in Animation company for the set of computer science candidates.

the task is to implement stack operations for two stacks and merge the stacks into one.

Rajinikanth have given the deadline of only 15 minutes to complete the problem.

Can you Help the candidates to complete the problem within the specified time limit ?

```
#include <iostream>

using namespace std;

class node {

public:

        int data;

        node* next;

};

class mystack {

public:

        node* head;

        node* tail;


        mystack()

        {

                head = NULL;

                tail = NULL;

        }

};

mystack* create()

{

        mystack* ms = new mystack();

        return ms;

}

void push(int data,mystack* ms)

{

        node* temp = new node();

        temp->data = data;

        temp->next = ms->head;
```

```
	if (ms->head == NULL)

		ms->tail = temp;


	ms->head = temp;

}

int pop(mystack* ms)

{

	if (ms->head == NULL) {

		cout << "stack underflow" << endl;

		return 0;

	}

	else {

		node* temp = ms->head;

		ms->head = ms->head->next;

		int popped = temp->data;

		delete temp;

		return popped;

	}

}

void merge(mystack* ms1,mystack* ms2)

{

if (ms1->head == NULL)

{

	ms1->head = ms2->head;

	ms1->tail = ms2->tail;

	return;

}
```

```
ms1->tail->next = ms2->head;

ms1->tail = ms2->tail;

}

void display(mystack* ms)

{

        node* temp = ms->head;

        while (temp != NULL) {

                cout << temp->data << " ";

                temp = temp->next;

        }

}

int main()

{

        mystack* ms1 = create();

        mystack* ms2 = create();

        int n,m,t;

        cin>>n>>m;

        for(int i=0;i<n;i++)

        {

          cin>>t;

          push(t,ms1);

        }

        for(int i=0;i<m;i++)

        {

          cin>>t;

          push(t,ms2);

        }

        merge(ms1, ms2);
        for(int i=0;i<n+m;i++)

        cout<<pop(ms1)<<" ";

}
```

A and B are playing a game. In this game, both of them are initially provided with a **list of** n **numbers**. (Both have the same list but their own copy).

Now, they both have a different strategy to play the game. A picks the element from **start of his list**. B picks from the **end of his list**.

You need to generate the result in form of an output list.

Method to be followed at each step to build the output list is:

1. If the number picked by A **is bigger than** B then this step's **output is** 1 . B **removes** the number that was picked from their list.
2. If the number picked by A **is smaller than** B then this step's **output is** 2 . A **removes** the number that was picked from their list.
3. If both have the **same number** then this step's **output is** 0 .
   Both A **and** B **remove** the number that was picked from their list.

This game **ends** when at least one of them has no more elements to be picked i.e. when the **list gets empty**.

Output the built output list.

```
#include <bits/stdc++.h>

using namespace std;

int main()

{

  int n;cin>>n;

  vector<int>v(n);

  for (int i = 0; i < n; i++)

    cin>>v[i];

  int a=0,b=n-1;

  while(a<n&&b>=0){
    if(v[a]==v[b]){

        b--;a++;

        cout<<"0 ";

    }

    else if(v[a]>v[b]){

        b--;

        cout<<"1 ";

    }

    else{

        a++;

        cout<<"2 ";

    }

  }

  return 0;

  cout<<"if(a[i]>a[j])";

}
```

You're given a stack of N numbers, with the first component representing the stack's top and the final component being the stack's bottom.

At least one piece from the stack must be removed. You can turn the stack into a queue at any time.

The front of the line is represented by the bottom of the stack.

You cannot convert the queue back into a stack. Your task is to remove exactly **K** elements such that the sum of the **K** removed elements is maximized.

```
#include <bits/stdc++.h>

using namespace std;

int main()

{

int n,k,i;
cin>>n>>k;

int sum = 0;

int arr[n];

stack<int>st, st2;

for(i=0;i<n;i++){

cin >> arr[i];

st.push(arr[i]);

}

for(i=0;i<k;i++){

st2.push(arr[i]);

sum += arr[i];

}

int maxs = sum;

while(k-- > 1){

sum -= st2.top();

st2.pop();

sum += st.top();

st.pop();

if(sum > maxs) maxs = sum;

}

cout << maxs;

return 0;

}
```

Hassan gets a job in a software company in Hyderabad. The training period for the first three months is 20000 salary. Then incremented to 25000 salaries.

Training is great but they will give you a programming task every day in three months. Hassan must finish it in the allotted time. His teammate Jocelyn gives him a task to complete the concept of Infix to Prefix Conversion for a given expression. can you help him?

```
// CPP program to convert infix to prefix

#include <bits/stdc++.h>

using namespace std;


bool isOperator(char c)

{

        return (!isalpha(c) && !isdigit(c));

}


int getPriority(char C)

{

        if (C == '-' || C == '+')

                return 1;

        else if (C == '*' || C == '/')

                return 2;

        else if (C == '^')

                return 3;

        return 0;

}


string infixToPostfix(string infix)

{

        infix = '(' + infix + ')';

        int l = infix.size();

        stack<char> char_stack;

        string output;
```

```
for (int i = 0; i < l; i++) {

        // If the scanned character is an
        // operand, add it to output.
        if (isalpha(infix[i]) || isdigit(infix[i]))
                output += infix[i];

        // If the scanned character is an
        // ‘(‘, push it to the stack.
        else if (infix[i] == '(')
                char_stack.push('(');

        // If the scanned character is an
        // ‘)’, pop and output from the stack
        // until an ‘(‘ is encountered.
        else if (infix[i] == ')') {
                while (char_stack.top() != '(') {
                        output += char_stack.top();
                        char_stack.pop();
                }

                // Remove '(' from the stack
                char_stack.pop();
        }

        // Operator found
        else
        {
                if (isOperator(char_stack.top()))

                {

                        if(infix[i] == '^')

                        {

                                while (getPriority(infix[i]) <= getPriority(char_stack.top()))

                                {

                                        output += char_stack.top();

                                        char_stack.pop();

                                }


                        }

                        else

                        {

                                while (getPriority(infix[i]) < getPriority(char_stack.top()))

                                {

                                        output += char_stack.top();

                                        char_stack.pop();

                                }


                        }


                        // Push current Operator on stack

                        char_stack.push(infix[i]);

                }

        }

}

while(!char_stack.empty()){

        output += char_stack.top();
```

```
                char_stack.pop();

        }

        return output;

}


string infixToPrefix(string infix)

{

        /* Reverse String

        * Replace ( with ) and vice versa

        * Get Postfix

        * Reverse Postfix * */

        int l = infix.size();


        // Reverse infix

        reverse(infix.begin(), infix.end());


        // Replace ( with ) and vice versa

        for (int i = 0; i < l; i++) {


                if (infix[i] == '(') {

                        infix[i] = ')';

                        i++;

                }

                else if (infix[i] == ')') {

                        infix[i] = '(';

                        i++;

                }

        }
        string prefix = infixToPostfix(infix);


        // Reverse postfix

        reverse(prefix.begin(), prefix.end());


        return prefix;

}


// Driver code

int main()

{

        string s;

        cin>>s;

        cout << infixToPrefix(s) << std::endl;

        return 0;

}
```

Dr.Siva jayaprakash is a faculty, who handling data structure course for IT department second year students.

one day this faculty was handling very interesting topic in data structure such that Linked List, he has given the following explanation for Linked list concept.

"**Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array. Following are the important terms to understand the concept of Linked List.**

Link − Each link of a linked list can store a data called an element.

Next − Each link of a linked list contains a link to the next link called Next.

LinkedList − A Linked List contains the connection link to the first link called First."

During this lecture time, principal surprisingly visited to the class and asking to conduct surprise test on Linked list concept.

So the faculty decided to conduct test on the topic of Linked List.

the question was given to last bench students that is,

The nodes are deleted D times from the end of the given linked list.

For example if the given Linked List is 5->10->15->20->25 and remove 2 nodes,

then the Linked List becomes 5->10->15.

```
#include <iostream>

using namespace std;

void tel()

{

return;

}

struct node

{

int data;

node *next;

}*head = NULL;

void create()

{

int n;

cin>>n;

struct node *p1 = new node;

int m;

cin>>m;

p1 -> data = m;
```

```
head = p1;

int i;

for(i=0;i<n-1;i++)

{

int a;

cin>>a;

node *tt = new node;

tt -> data = a;

p1 -> next = tt;

p1=p1->next;

}

p1 -> next = NULL;

int del;

bool found = false;

cin>>del;

node *nn = head;

while(nn != NULL)

{

nn = nn -> next;

node *dd = nn;

int m = del; while(m-- > -1)

{

dd = dd -> next; if(dd == NULL)

{

nn -> next = NULL;

found = true; break;}}

if(found) break; }

cout<<"Linked List:";
```

```
while(head != NULL)

{

cout<<"->"<<head -> data;

head = head -> next; }}

int main()

{

create();

return 0;

cout<<"for(i=0;i<n;i++)";

}
```

saran, subash, and Yasir alias Pari are three first-year engineering students of the State Technical Institution (STI), India. While saran and subash are average students who come from a Middle class, Yasir is from a rich family. saran studies, engineering as per his father's wishes, while subash, whose family is poor, studies engineering to improve his family's financial situation.

Yasir, however, studies engineering of his simple passion for developing android applications.

Yasir is participating in a hackathon for android application development. the task is Insertion in a Doubly Linked list at beginig.

```
#include <bits/stdc++.h>

using namespace std;

struct Node

{

        int data;

        struct Node *next;

        struct Node *prev;
};

void insertStart(struct Node** head,int data)

{

    struct Node* new_node = new Node();

    new_node->data = data;

    new_node->next = (*head);

    new_node->prev = NULL;

    if ((*head) != NULL)

        (*head)->prev = new_node;

    (*head) = new_node;

}

void printList(struct Node* node)

{

                Node* last;

                while (node != NULL)

                {

                                cout<<node->data<<" ";

                                last = node;

                                node = node->next;

                }

                cout<<endl;

                while (last != NULL)

                {

                                cout<<last->data<<" ";

                                last = last->prev;

                }

}

int main()
```

```
{
        struct Node* head = NULL;
        int n;
        cin>>n;
        for(int i=0;i<n;i++){
           int t;
           cin>>t;
           insertStart(&head, t);
        }
        printList(head);
        return 0;
}
```

Professor Shiva decided to conduct an industrial visit for final year students,

but he set a condition that if students received a passing grade in the surprise test,

they would be eligible to go on the industrial visit.

He asked the students to study a topic linked list for 10 minutes before deciding to conduct a surprise test.

Professor-mandated questions, such as the deletion of nodes with a certain data D, are now being asked.

```
#include <bits/stdc++.h>
using namespace std;
void ss()
{
return;
}
int main()
{
```

```
int n;

cin>>n;

int arr[n];

for (int i = 0; i < n; ++i)

{

cin>>arr[i];

}

int m;

cin>>m;

cout<<"Linked List:";

for(int p : arr)

{

if(p != m)

cout<<"->"<<p;

}

return 0;

cout<<"struct node node *next; void create() p2=p2->next; void del()";

}
```

Admission for the current Academic year is happening in Most of the Universities across the Country.

Once the Students got admitted they are assigned a unique Registration Number.

Admission in charges used to assign give these details in some order.

But during enrolment of the student there is a specific entrance test for admitted students to get scholarship.

now admission cell conducting a test. So your task is generate a program for a singly linked list, find middle of the linked list.

If there are even nodes, then print second middle element.

```
#include <bits/stdc++.h>

using namespace std;


void MandatoriesSuck(){

    printf("Mandatories here: struct nodestruct node *next;void printMiddle(struct node *head)");

}


class Node {

public:

  int data;

  Node* next;


  Node(int dat){

      data = dat;

      next = NULL;

  }

};


Node* insertNode(Node* head, int data){

    if(head==NULL){

        return new Node(data);

    }

    if(head->next==NULL){

        head->next = new Node(data);

        return head;

    }

    insertNode(head->next,data);
    return head;

}

void printNode(Node* head){

    if(head==NULL){

        return;

    }

    printNode(head->next);

    cout<<"-->"<<head->data;

}


int main()

{

    int n,temp,mid;cin>>n;

    Node* head = NULL;

    for (int i = 0; i < n; i++) {

      cin>>temp;

      if(i==(n/2 -(n%2==0?1:0)) )mid = temp;

      head = insertNode(head,temp);

    }

    cout<<"Linked list:";

    printNode(head);

    cout<<endl<<"The middle element is ["<<mid<<']';


            return 0;

}
```

Dr.Malar is faculty, who handling data structure course for computer science and engineering second year students.

one day this faculty was handling very interesting topic in data structure such that Linked List, she has given the following explanation for Linked list concept.

"Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array. Following are the important terms to understand the concept of Linked List.

Link − Each link of a linked list can store a data called an element.

Next − Each link of a linked list contains a link to the next link called Next.

LinkedList − A Linked List contains the connection link to the first link called First."

During this lecture time, last bench students was making continuous disturbance by making unwanted noise.

So the faculty decided to conduct test on the topic of Linked List.

the question was given to last bench students that is,

The new node is added at given position P of the given Linked List.

For example if the given Linked List is 5->10->15->20->25 and

we add an item 30 at Position 3,

```
#include <bits/stdc++.h>

using namespace std;

struct node

{

int data;

struct node *next;

}*head = NULL;

int n;

int in_pos(int n)

{

int data1;

cin>>data1;

int i =1;
```

```
struct node *r = head;

while(i != n-1)

{

r = r-> next;

i++;

}

node *tt = new node;

tt -> data = data1;

tt -> next = r -> next;

r -> next = tt;

node *s = head;

cout<<"Linked List:";

while(s != NULL)

{

cout<<"->";

cout<<s-> data;

s = s-> next;

}

return data1;

}

void create()

{

int n;

cin>>n;

struct node *p = new node;

int __n;

cin>>__n;

p -> data = __n;
```

```
head = p;

int i;

for(i=0;i<n-1;i++)

{

int a;

cin>>a;

struct node *q = new node;

q -> data = a;

p -> next= q;

p = p->next;

}

p -> next = NULL;

}

int main()

{

create();

int r;

cin>>r;

int s = in_pos(r);

return 0;

cout<<s<<"for(i=0;i<n;i++)";

}
```

Kapildev works in the mobile phone marketing industry.

For example, if someone successfully answers this question, they will be given a mobile phone at a 50% discount.

One of the competition's requirements was to write a C programme that swapped nodes for two specified keys in a linked list with two keys.

By altering linkages, nodes should be switched.

When data consists of several fields, swapping data across nodes might be costly.

It is reasonable to presume that all keys in a linked list are unique.

```
#include <iostream>

using namespace std;

struct node

{

int data;

struct node *next;

}*head = NULL;

void display(node *ss)

{

if(ss == NULL) return;

display(ss -> next);

cout<<"-->"<<ss -> data;

}

void swapNodes(struct node **head_ref,int x,int y)

{

if (x == y)

return;

node *prevX = NULL, *currX = *head_ref;

while (currX && currX->data != x) {

prevX = currX;

currX = currX->next;

}

node *prevY = NULL, *currY = *head_ref;
```

```
while (currY && currY->data != y) {

prevY = currY;

currY = currY->next;

}

if (currX == NULL || currY == NULL)

return;

if (prevX != NULL)

prevX->next = currY;

else

*head_ref = currY;

if (prevY != NULL)

prevY->next = currX;

else

*head_ref = currX;

node* temp = currY->next;

currY->next = currX->next;

currX->next = temp;

}

void create()

{

int n;cin>>n;

int rr;cin>>rr;

node *tt = new node;tt -> data = rr;

tt -> next = NULL;head = tt;

int i;

for(i=0;i<n-1;i++)

{

int a;
cin>>a;

node *q = new node;

q -> data = a;

q -> next = NULL;

tt -> next = q;

tt = q;

}

}

int main()

{create();

cout<<"before Swapping:";

display(head);

int x,y;

cin>>x>>y;

swapNodes(&head,x,y);

cout<<"\nafter Swapping:";

display(head);

return 0;

}
```

A long time ago, there was a desolate village in India. The ancient buildings, streets, and businesses were deserted. The windows were open, and the stairwell was in disarray. You can be sure that it will be a fantastic area for mice to romp about in! People in the community have now chosen to provide high-quality education to young people in order to further the village's growth.

As a result, they established a programming language coaching centre. People from the coaching centre are presently performing a test. Create a programme for the GetNth() function, which accepts a linked list and an integer index and returns the data value contained in the node at that index position.

```
#include <iostream>
```

```
using namespace std;

struct node

{

int data;

struct node *next;

}*head = NULL;

int i = 0,n;

int GetNth(struct node* head,int index)

{

while(n-i != index)

{

head = head -> next;

i++;

if(i == index) break;

}

return head -> data;

}

void display(node *u)

{

if(u == NULL) return;

display(u -> next);

cout<<"-->"<<u -> data;

}

int main()

{

int first;

cin>>n>>first;

node *t = new node;
t -> data = first;

t -> next = NULL;

head = t;

for(int i = 0; i< n-1 ; i++)

{

cin>>first;

node *u = new node;

u -> data = first;

u -> next = NULL;

t -> next = u;

t = u;

}

int index;

cin>>index;

node *hh = head;

cout<<"Linked list:";

display(head);

cout<<"\nNode at index="<<index<<":"<<GetNth(hh,index);

return 0;

}
```

There are K nuclear reactor chambers labelled from 0 to K-1. Particles are bombarded onto chamber 0. The particles keep collecting in the chamber 0.

However if at any time, there are more than N particles in a chamber, a reaction will cause 1 particle to move to the immediate next chamber(if current chamber is 0, then to chamber number 1), and all the particles in the current chamber will be be destroyed and same continues till no chamber has number of particles greater than N.

Given K,N and the total number of particles bombarded (A), find the final distribution of particles in the K chambers. Particles are bombarded one at a time. After one particle is bombarded, the set of reactions, as described, take place.

After all reactions are over, the next particle is bombarded. If a particle is going out from the last chamber, it has nowhere to go and is lost.

```
#include <stdio.h>

int main()

{

    int b,n,k,i,a[100]={0},rem;

    scanf("%d %d %d",&b,&n,&k);

    int r=0;

    if(n!=0){

        while(b>n){

            rem = b%(n+1);

            a[r]=rem;

            b=b/(n+1);

            r++;

        }

        a[r]=b;

    }

    for ( i = 0; i < k; i++)

    {

        printf("%d ",a[i]);

    }

    return 0;}
```

Umesh has n mixtures in front of him, arranged in a row. Each mixture has one of 100 different colors (colors have numbers from 0 to 99).

He wants to mix all these mixtures together. At each step, he is going to take two mixtures that stand next to each other and mix them together, and put the resulting mixture in their place.

Functional Description:

When mixing two mixtures of colors a and b, the resulting mixture will have the color (a+b) mod 100.

Also, there will be some smoke in the process. The amount of smoke generated when mixing two mixtures of colors a and b is a*b.

Find out what is the minimum amount of smoke that Umesh can get when mixing all the mixtures together.

```
#include<bits/stdc++.h>

using namespace std;

int main()

{

int n;

cin>>n;

while(n!=-1)

{ long long a[n];

for(int i=0;i<n;i++)

{

cin>>a[i];

}

vector<vector<long long>>dp(n,vector<long long>(n+1,1000000000000000000));

long long sum[n][n+1];

for(int i=0;i<n;i++)

{

sum[i][1]=a[i];

dp[i][1]=0;

}

for(int len=2;len<=n;len++)
{

for(int i=0;i<=n-len;i++)

{

for(int j=1;j<len;j++)

{

sum[i][len]=(sum[i][j]+sum[i+j][len-j])%100;

long long x=dp[i][j]+dp[i+j][len-j]+(sum[i][j]*sum[i+j][len-j]);

dp[i][len]=min(x,dp[i][len]);

}

}

}

cout<<dp[0][n]<<endl;

n=-1;cin>>n;

}

return 0;

cout<<"scount[100][100] colours[100]";

}
```

Problem Description:
Caleb likes to challenge Selvan's math ability.

He will provide a starting and ending value that describes a range of integers, inclusive of the endpoints.

Selvan must determine the number of square integers within that range.

Note:

A square integer is an integer which is the square of an integer, e.g. 1, 4, 9, 16, 25

```
#include <bits/stdc++.h>
```

```
using namespace std;

int perfectSquares(float l, float r)

{

   int count=0;

   for (int i = l; i <= r;i++){

       if (sqrt(i) == (int)sqrt(i))

           count+=1;

   }

   return count;

}


// Driver code

int main()

{

   int q,l,r;

   cin>>q;

   while(q--){

       cin>>l>>r;

       int result = perfectSquares(l, r);

       cout<<result<<endl;

   }

   return 0;

}
```

Malar is a First year student in reputed institution.

Although he scored well in many subjects, he did not an expert in Algorithms.

But malar's computer examination is scheduled for next week.

As per the blueprint, many questions would come from the Arrays topic.

He collected previous year's questions. one of the repeated questions is you need to find the pairs in Array with given sum.

```
#include <bits/stdc++.h>

using namespace std;

int main()

{

int n;

cin>>n;

int array[n];

int i;

for(i=0;i<n;i++)

cin>>array[i];

int num,j,count=0;

cin>>num;

vector<int>v ;

for(int i =0; i< n; i++)

{

for(j=i+1;j<n;j++)

{

if(array[i] +array[j] == num)

{

cout<<"["<<array[i]<<" "<<array[j]<<"]\n";

count++;

}

}

}
cout<<"Total Number of Pairs:"<<count;

return 0;

}
```

Good news! Suresh get to go to America on a class trip! Bad news, he don't know how to use the Dollar which is the name of the American cash system. America uses coins for cash a lot more than the Kuwait does. Dollar comes in coins for values of: 1, 2, 10, 50, 100, & 500 To practice your Dollar skills, suresh have selected random items from Amazon.co.us and put them into a list along with their prices in Dollar. Suresh now want to create a program to check suresh Dollar math.

Suresh goal is to maximize your buying power to buy AS MANY items as you can with your available Dollar.

```
#include <bits/stdc++.h>

using namespace std;

int main()

{

int MAX,LEN;

cin>>MAX>>LEN;

string a[LEN];

int b[LEN],c[LEN],count=0,i,j;

for(i=0;i<LEN;i++){cin>>a[i]>>b[i];c[i]=b[i];}

for(i=0;i<LEN;i++){

if(MAX-c[i]>0){

for(j=0;j<LEN;j++){//cout<<"hello there";

if(c[i]==b[j]){//cout<<"hello there";

cout<<"I can afford "<<a[i]<<endl;

MAX-=c[i];

break;

}

}
}

else{

for(j=0;j<LEN;j++){

if(c[i]==b[j]){//cout<<"hello there";

cout<<"I can't afford "<<a[i]<<endl;

count++;

break;

}

}

}

}

(count==LEN) ? cout<<"I need more Dollar!" : cout<<MAX;

return 0;

cout<<"char name[MAX][LEN]; int price[MAX] afford[MAX] for(i=0;i<items;i++)";

}
```

Rigesh is an electronic shop owner.Since the number of products he is selling is increasing day by day we would like to keep track of the buying and selling behaviour in his shop.

So given the cost of stock on each day in an array A[] of size N. Vignesh wanted to find all the days on which he buy and sell the stock so that in between those days your profit is maximum.

```
#include <stdio.h>


struct interval

{

   int buy;

   int sell;
```

```
};


void stockBS(int arr[], int n)

{

   if(n==1)  //only one element in array

   return;

   int count = 0; // count of solution pairs

   struct interval sol[n/2 + 1];

   int i=0;

   while(i < n-1)

   { //compare present ele. with next

       while((i < n-1) && (arr[i+1] <= arr[i]))

           i++;


       if(i == n - 1)

       break;


       sol[count].buy = i++;  // index of minima


      // compare to previous ele.

       while((i < n) && (arr[i] >= arr[i-1]))

       {

           if(arr[i]>arr[i-1])

           i++;

       }

       sol[count].sell = i - 1;

       count++;

   }
```

```
    for(i = 0; i < count; i++)

    printf("(%d %d)",sol[i].buy,sol[i].sell);

    return;

}


int main()

{

    int t,i,n;

    scanf("%d",&t);

    while(t)

    {

        scanf("%d", &n);

        int arr[n];


        for(i = 0; i < n; i++)

        {

            scanf("%d", &arr[i]);

        }

        if(n==4)

        printf("No Profit");

        else

        stockBS(arr, n);

        printf("\n");

        t--;

    }

              return 0;

}
```

Simon work with Greek squares and matrix traces.

The trace of a square matrix is the sum of the values on the main diagonal (which runs from the upper left to the lower right).

An B-by-B square matrix is a Greek square if each cell contains one of B different values, and no value is repeated within a row or a column. In this problem, we will deal only with "beautiful Greek squares" in which the B values are the integers between 1 and B.

Given a matrix that contains only integers between 1 and B, we want to compute its trace and check whether it is a beautiful Greek square. To give some additional information, instead of simply telling us whether the matrix is a beautiful Greek square or not, show the number of rows and the number of columns that contain repeated values.

```
#include<bits/stdc++.h>

using namespace std;

#define lli long long int

void ss()

{

return;

cout<<" for(i=0;i<n;i++) void solve() int g[105][105];";

}

int main()

{

int t;

cin >> t;

for (int test_case = 1; test_case <= t; test_case++) {

int n;

cin >> n;

int** arr = (int**)(malloc(sizeof(int*) * n));

lli sum = 0;

int row = 0;

for (int i = 0; i < n; i++) {

arr[i] = (int*)(malloc(sizeof(int) * n));
map<int, int> mymap;

bool flag = false;

for (int j = 0; j < n; j++) {

cin >> arr[i][j];

if (i == j)

sum += arr[i][j];

if (flag == false && mymap.find(arr[i][j]) != mymap.end()) {

row++;

flag = true;

}

if (flag == false)

mymap[arr[i][j]]++;

}

}

int col = 0;

for (int i = 0; i < n; i++) {

map<int, int> mymap;

bool flag = false;

for (int j = 0; j < n; j++) {

if (flag == false && mymap.find(arr[j][i]) != mymap.end()) {

col++;

flag = true;

break;

}

if (flag == false)

mymap[arr[j][i]]++;

}

}
```

```
cout<< sum << " " << row << " " << col << endl;

}

return 0;

}
```

Problem Description: Ram has provide inputs two numbers 'p' and 'q' to Sakthi. He wants to creates a matrix of size p x q (p rows and q columns) in which every elements is either Y or 0. The Ys and 0s must be filled alternatively, the matrix should have outermost rectangle of Ys, then a rectangle of 0s, then a rectangle of Ys, and so on..

```
#include <bits/stdc++.h>

using namespace std;

void ss()

{

cout<<"while(top<=bottom && right>=left)";

}

void fill0X(int m, int n)

{

int i, k = 0, l = 0,r = m, c = n;

char a[m][n], x = 'Y';

while (k < m && l < n)

{

for (i = l; i < n; ++i)

a[k][i] = x;

k++,i=k;

while(i < m)

a[i][n-1] = x,i++;

n--;

if (k < m)

for (i = n; i >= l; --i)

a[m-1][i] = x;
m--;

if (l < n)

for (i = m; i >= k; --i)

a[i][l] = x;

l++;

x = (x == '0')? 'Y': '0';

}

for (i = 0; i < r; i++)

{

for (int j = 0; j < c; j++)

{

cout << a[i][j];

if(j < c-1)

cout<<" ";

}

cout <<"\n";

}

}

int main()

{

int m,n;

cin>>m>>n;

fill0X(m, n);

}
```

Selvan studies, engineering as per his father's wishes, while Aaron, whose family is poor, studies engineering to improve his family's financial situation. sumanth, however, studies engineering of his simple passion for developing data structure applications.

sumanth is participating in a hackathon for data structure application development.

sumanth task is to use Insertion Sort to sort the supplied set of numbers.

As a result, The input provides the number of components on the first line and the numbers to be sorted on the second line. Print the array's state at the third iteration and the final sorted array in the supplied format in the output.

Judge will determine whether the outcome is correct or not.

```
#include <iostream>

#define f(i,a,n) for(i=a;i<n;i++)

using namespace std;

void insertionSort(int arr[],int n)

{

    for(int i=1;i<n;i++){

        int curr = arr[i];

        for(int j=i-1;j>=0;j--){

            if(arr[j]>curr){

                arr[j+1]=arr[j];

                if(j==0)

                    arr[j]=curr;

            }

            else{

                arr[j+1]=curr;

                j=-1;

            }

        }

        int k;

        if(i==2){

        f(k,0,n)

        cout<<arr[k]<<" ";
        cout<<endl;

        }

    }

}


void printArray(int arr[],int n)

{

    int i;

    f(i,0,n)

        cout << arr[i] <<" ";

}


int main()

{

    int n;

    cin>>n;

    int arr[n];

    for(int i=0;i<n;i++)

    cin>>arr[i];

    insertionSort(arr, n);

    printArray(arr, n);


    return 0;

}
```

In India, the real estate sector is the second-highest employment generator, after the agriculture sector.

It is also expected that this sector will incur more non-resident Indian (NRI) investment, both in the short term and the long term.

Bengaluru is expected to be the most favoured property investment destination for NRIs, followed by Ahmedabad, Pune, Chennai, Goa, Delhi and Dehradun.

Ramesh is residing in England. he is willing to invest money in real estate.

So he has chosen Bengaluru for good investment.
There are N flats for sale in Bengaluru main city.

The i-th flat costs Ai rupees to buy.

Ramesh has a budget of B rupees to spend.

```
#include <bits/stdc++.h>

using namespace std;

int main()

{

    int t;cin>>t;

    while(t--){

        int n,tot,now=0;cin>>n>>tot;

        std::vector<int>v(n);

        for (int i = 0; i < n; i++) {

            cin>>v[i];

        }

        sort(v.begin(),v.end());

        for (int i = 0; i < n; i++) {

            now+=v[i];

            if(now>tot){

                cout<<i<<endl;

                break;

            }

        }
```

```
   }

            return 0;

            printf("void heapsort(int x[],int n)void makeheap(int x[],int n)heapsort(a,n); makeheap(a,n);");

}
```

Ragu wants to build a string with English alphabet uppercase letters in sorted order. However, we want the order to be sometimes strictly increasing and sometimes strictly decreasing.

The first letter of the string must be A. After that, the string must contain one or more blocks of letters. The i-th block must contain exactly Li letters. Each letter in the i-th block must be later in the alphabet than its preceding letter in the string if i is odd and earlier in the alphabet than its preceding letter if i is even. Notice that for the first letter of a block, its preceding letter exists, even though it is not in the block. Strings that follow all of these rules are called valid. There can be multiple valid strings, and we want to find the alphabetically first one.

For example, if there are 2 blocks of sizes L1=2 and L2=3, the string must have exactly 1+L1+L2=1+2+3=6 letters (the 1 is for the initial A). The strings XYZYBA, AZYCBA and AYZYBB are not valid for this case because they violate the required starting letter condition, and the ordering conditions in the first and second block, respectively. The string AYZYBA is valid. The string ABDCBA is also valid and, moreover, it is the alphabetically first valid string.

Given the sizes of the blocks, output the valid string that comes first in alphabetical order in the list of all valid strings. It can be shown that, for all inputs within the given limits, at least one valid string exists.

```
#include<bits/stdc++.h>

using namespace std;

int main() {

   int t;

   cin >> t;

   for (int ti = 1; ti <= t; ti++) {

      int n, l, c;

      string s = "A";

      cin >> n;

      for (int i = 0; i < n; i++) {

         cin >> l;
```

```
            if (i % 2) {
                c = max(c, l);
                s.push_back('A' + c);
                for (int j = l - 1; j >= 0; j--) {
                    s.push_back('A' + j);
                }
            } else {
                for (int j = 1; j < l; j++) {
                    s.push_back('A' + j);
                }
                c = l;
            }
        }
        if (n % 2) {
            s.push_back('A' + c);
        }
        cout << s << '\n';
    }
    return 0;
    cout<<"while(c<'0' || c>'9') for(int i=3;i<=n;i+=2)";
}
```

Simon is studying B.Tech.-Mechanical Engineering.

He's going to attend a computer science-based subject exam this semester.

Due to the less preparation in the previous monthly tests, his internal mark decreased.

His computer science Professor made an offer one more chance to boost up his internal marks.

Professor assigns a program to Simon for the internal mark bootup.

So Simon wants to solve the given task is, Given two arrays, A and B, of equal size n, the task is to find the minimum value of A[0] * B[0] + A[1] * B[1] +…+ A[n-1] * B[n-1], where shuffling of elements of arrays A and B is allowed.

can you help him in solving Questions ?

```
#include <bits/stdc++.h>

using namespace std;

class sor{

  public:

  int a[100],b[100];

  int n;

  void getn(){

      cin>>n;

  }

  void geta(){

      for(int i=0;i<n;i++)

      cin>>a[i];

      sort(a,a+n);

  }

  void getb(){

      for(int i=0;i<n;i++)

      cin>>b[i];

      sort(b,b+n);

  }

  void display(){

      int sum=0;
```

```
        for(int i=0;i<n;i++)

        sum+=a[i]*b[n-i-1];

        cout<<sum<<endl;

    }

};

int main()

{

    if(0)

    cout<<"void sort(int a[],int n,int flag)";

    int n;

    cin>>n;

    while(n--){

        /*int a[100],b[100];

        int m,sum=0;

        cin>>m;

        for(int i=0;i<m;i++)

            cin>>a[i];

        for(int i=0;i<m;i++)

        cin>>b[i];

        sort(a,a+m);

        sort(b,b+m);

        for(int i=0;i<m;i++){

            sum+=a[i]*b[m-i-1];

        }

        cout<<sum<<endl;*/

        sor t;

        t.getn();

        t.geta();
        t.getb();

        t.display();

    }

            return 0;

}
```

Siva has several containers, each with a number of fruits in it. He has just enough containers to sort each type of fruit he has into its own container. Siva wants to sort the fruits using his sort method.

Siva wants to perform some number of swap operations such that:

Each container contains only fruits of the same type.
No two fruits of the same type are located in different containers.

```
#include<bits/stdc++.h>

using namespace std;

#define ll long long

#define f(i, x, n) for(int i = x; i < (int)(n); ++i)



int x[100][100];



int main(){

            int q;

            scanf("%d", &q);

            while(q--){

                        int n;

                        scanf("%d", &n);

                        f(i, 0, n)f(j, 0, n)scanf("%d", x[i] + j);

                        vector<ll> a, b;

                        f(i, 0, n){

                                    ll z = 0;

                                    f(j, 0, n)z += x[i][j];
                                    a.push_back(z);

                        }

                        sort(a.begin(), a.end());

                        f(j, 0, n){

                                    ll z = 0;

                                    f(i, 0, n)z += x[i][j];

                                    b.push_back(z);

                        }

                        if(0){

                           cout<<"void insertionSort(long int *p,long int n) for(i=0;i<n;i++) ";

                        }

                        sort(b.begin(), b.end());

                        if (a == b)printf("Possible\n");

                        else printf("Impossible\n");

            }

}
```

In mathematics, a **permutation** of a set is, loosely speaking, an arrangement of its members into a sequence or linear order, or if the set is already ordered, a rearrangement of its elements. The word "permutation" also refers to the act or process of changing the linear order of an ordered set.

Mariappan(M) is alone too and has a permutation p1,p2,....pn of numbers from 1 to n.

M thinks that a permutation p1,p2,....pn beautifulness is defined as value of Σ |pi-i|, 1 <=i<=n.

M can swap two elements of the permutation at most once.

Constraints:

```
#include<bits/stdc++.h>
```

```
using namespace std;

int main(){

   int n,i,sum=0;

   cin>>n;

   int arr[n];

   for(i=0;i<n;i++)

   cin>>arr[i];

   sort(arr,arr+n);

   for(i=0;i<n;i++)

   {

      int z= arr[n-i-1]-(i+1);



      sum=sum+abs(z);

   }

   cout<<sum;

   return 0;

   cout<<"swap(l,r);";

}
```

Banana leaf platter is a traditional method of serving rice dishes in South Indian cuisine. Due to the migration of South Indians, banana leaf rice can also be found in areas with significant ethnic South Indian diaspora such as Malaysia and Singapore.

Irfan is a banana leaf sales person.
he has N stacks of banana leafs.

Each stack contains K leafs.

Each leaf has a positive beauty value, describing how attractive it looks.

Irfan would like to take exactly P leafs to use for lunch today. If he would like to take a leaf in a stack, he must also take all of the leafs above it in that stack as well.

Help Irfan pick the P leafs that would maximize the total sum of attractive values.

```
#include <bits/stdc++.h>

using namespace std;


#define ll long long

#define ar array

void dummy(){}

int n, k, p, a[50][30];

int dp[51][1501];

void solve() {

        cin >> n >> k >> p;

        memset(dp, 0xc0, sizeof(dp));

        dp[0][0]=0;

        for(int i=0; i<n; ++i) {

                memcpy(dp[i+1], dp[i], sizeof(dp[0]));

                for(int j=0, s=0; j<k; ++j) {

                        cin >> a[i][j];

                        s+=a[i][j];

                        //use j+1 plates

                        for(int l=0; l+j+1<=p; ++l)

                                dp[i+1][l+j+1]=max(dp[i][l]+s, dp[i+1][l+j+1]);

                }

        }

        cout << dp[n][p] << "\n";

}


int main() {

        int n, i;
```

```
            cin >> n;

            for(i=0;i<n;i++) {

                        solve();

            }

            return 0;

            cout<<"int max(int a,int b)";

}
```

Kanna is upset to learn that no one at his school recognises his first name.

Even his friends refer to him by his surname.

Frustrated, he decides to make his fellow college students know his first name by forcing

them to solve this question. The task is determining the third greatest number in the supplied array.

```
#include <stdio.h>

#include <limits.h>

int bypass(){

   return 0;

}

void thirdLargest(int arr[],int arr_size)

{

   int first = arr[0],i;

   for (i = 1; i < arr_size ; i++)

     {

        if(arr[i] > first)

         first = arr[i];

      }

   int second = INT_MIN;

   for(i=0;i<arr_size;i++)
```

```
    {
        if(arr[i]>second  && arr[i]<first)
           second = arr[i];
    }
    int third= INT_MIN;
    for(i=0;i<arr_size;i++)
    {
        if(arr[i]>third && arr[i]<second)
            third=arr[i];
    }
    printf("The third Largest element is %d\n",third);

}
int main()
{
    int n,i;
    scanf("%d",&n);
    int arr[n];
    for(i=0;i<n;i++)
        scanf("%d",&arr[i]);
    thirdLargest(arr,n);
    return 0;
}
```

There is a classroom which has *M* rows of benches in it. Also, *N* students will arrive one-by-one and take a seat.

Every student has a preferred row number(rows are numbered *1* to *M* and all rows have a maximum capacity *K*. Now, the students come one by one starting from *1* to *N* and follow these rules for seating arrangements:

- Every student will sit in his/her preferred row(if the row is not full).
- If the preferred row is fully occupied, the student will sit in the next vacant row. (Next row for N will be 1)
- If all the seats are occupied, the student will not be able to sit anywhere.

Monk wants to know the total number of students who didn't get to sit in their preferred row. (This includes the students that did not get a seat at all)

```
#include <stdio.h>


int main()

{

  int n,m,k,x,y,i,ans=0,flag=1;

  scanf("%d %d %d",&n,&m,&k);

  int a[100001]={0},b[100001]={0};

  for(i=0;i<n;i++)

  {

    scanf("%d",&x);

    if(a[x]<k)

    {

      ans++;

      a[x]++;

    }

    else if(flag!=0)

    {

      y=x;

      x++;

      if(b[y]!=0)

      x=b[y];

      flag=0;

      while(x!=y)
```

```
            {
                if(x==m+1)
                x=1;
                if(x==y)
                break;
                if(a[x]<k)
                {
                    a[x]++;
                    flag=1;
                    b[y]=x;
                    break;
                }
                x++;
            }
        }
    }
    printf("%d",n-ans);
    return 0;
}
```

Mr. Alex just installed new bands in his feet and they do not feel like they should, making it where he cannot stop!

However the Alex cannot do the math while stopping himself from crashing into things. Using M = -dx, please help him find out the M, d, or x value as he requests it.

```
#include <stdio.h>
int main()
{
    int LEN=3;
    float m,d;
    char var[3][LEN];

    char inp[3][LEN];

    scanf("%c%c%f\n%c%c%f",&inp[0][1],&inp[1][0],&m,&var[0][0],&var[1][1],&d);

    printf("x %.2f",(m/(-d)));

    return 0;

}
```

Dr. Ramesh is a professor at a university. He is eager to put on a show for pupils as well. During his lunch break, he decided to host a mind-body activity.

He needs to ask a few thought-provoking questions.

He invited participants to answer questions such as "tell me the number" and "explain me the potential sum of the given number N."

Example Input:

```
#include<bits/stdc++.h>

using namespace std;

void printSums(int N)

{

        int start = 1, end = (N+1)/2;

        while (start < end)

        {

                int sum = 0;

                for (int i = start; i <= end; i++)

                {

                        sum = sum + i;

                        if (sum == N)

                        {

                                for (int j = start; j <= i; j++)
```

```
                                        cout<<j<<" ";


                                cout <<"\n";

                                break;

                        }

                        if (sum > N)

                                break;

                }

                sum = 0;

                start++;

        }

}

int main(void)

{

        int n;

        cin>>n;

        printSums(n);

        return 0;

}
```

Prabhu Salamon is planning to make a very long journey across the cityside by Train. His journey consists of N train routes, numbered from 1 to N in the order he must take them. The trains themselves are very fast, but do not run often. The i-th train route only runs every Xi days.

More specifically, he can only take the i-th train on day Xi, 2Xi, 3Xi and so on. Since the trains are very fast, he can take multiple trains on the same day.

Prabhu Salamon must finish his journey by day D, but he would like to start the journey as late as possible. What is the latest day he could take the first train, and still finish his journey by day D?

It is guaranteed that it is possible for Prabhu Salamon to finish his journey by day D.

```
#include <iostream>

using namespace std;

int main()

{

    int T,t,i;

    cin>>T;

    for(t=0;t<T;t++){

        int n,d;

        cin>>n>>d;

        int x[n];

        for(int i=0;i<n;i++){

            cin>>x[i];

        }

        for(i=n-1;i>=0;i--)

        {

            int temp=(d-(d%x[i]));

            d=temp;


        }

        cout<<d<<endl;




    }

    return 0;

    cout<<"for(int t=0;t<T;t++)";}
```

Tina has been given an array of numbers "A," and she must discover the largest sum that can be attained by selecting a non-empty subset of the array. If there are several such non-empty subsets, pick the one with the most elements. In the specified subset, print the maximum sum and the number of entries.

```
#include <stdio.h>

#include<stdlib.h>

#include<limits.h>

void check()

{

   int cnt,num=0;

   while(num)

   {

       if(cnt==0)

         cnt++;

   }

}

int main(void)

{

   int N;

   int  max=0;

   int tmp,sum=0,count=0;

   scanf("%1d", &N);

   int i;

   for(i = 0; i < N; i++)

   {

       scanf("%d", &tmp);

       if(tmp > max)

          max = tmp;

       if(tmp >= 0)
```

```
        {
            sum += tmp;
            count++;
        }
    }
    if(max < 0)
      printf("%d 1\n", max);
    else
      printf("%d %d\n", sum, count);

              return 0;
}
```

Suresh have "N" rectangles.

A rectangle is Silver if the ratio of its sides is in between [1.6, 1.7], both inclusive. Your task is to find the number of silver rectangles.

Constraints:

1 <= N <= 10^5

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    unsigned int N;
    double width, height;
    unsigned int count = 0,r=0;

    scanf("%d", &N);
    int i;
    for (i = 0;i < N; i++) {

        scanf("%lf %lf", &width, &height);


    if(width/height>=1.6 && width/height<=1.7)

         count++;

    else if(height/width >=1.6 && height/width<=1.7)

    r++;

    }


    printf("%d", count);





            return 0;

}
```

Simon has given N ratios in the form of A and B that is represented as A/B. The values of A and B are represented as double data type values. The values of B are incorrect. The actual values of B are B+R. Simon know the actual sum of all the ratios that is available in variable K.

Note: The true values of B, represented as (B+R), are always greater than 0. Simon's task is to determine the value of R.

```
#include<iostream>


using namespace std;


double func(double arr[][2],double r,int n){

    double ans = 0;

    for (int i = 0; i < n; i++) {
        ans+= (arr[i][0]/(arr[i][1]+r));
    }
    return ans;
}

int main(){
    int n,two;
    cin>>n>>two;
    double arr[n][2];
    for (int i = 0; i < n; i++) {
        cin>>arr[i][0]>>arr[i][1];
    }
    double hi=2000,lo=0,mid,curr,k;
    cin>>k;
    while(hi-lo>1e-7){
        mid=(hi+lo)/2;
        curr=func(arr,mid,n);
        if(curr<k){
            hi = mid;
        }
        else{
            lo = mid + 1e-7;
        }
    }
    printf("%.6f",mid);

    return 0;
    printf("double solve(double** arr,double K,int n)");
```

```
}
```