```
///         Bismillahi-r-Rahmani-r-Rahim
#include<bits/stdc++.h>
using namespace std;
#define debug(args...){ string _s = #args;replace(_s.begin(),_s.end(),',', '
');stringstream _ss(_s);istream_iterator<string>_it(_ss);err(_it, args);} cout<<endl;
void err(istream_iterator<string> it) {}
template<typename T, typename... Args>
void err(istream_iterator<string> it, T a, Args... args) {cerr << *it << "=" << a << ", ";
err(++it, args...);}
#define ll                long long int
#define MAX               2134567891
#define PF(a)             cout<<a<<endl;
#define pf(a)             printf("%lld", a);
#define sf(a)             scanf("%lld", &a);
#define fr(i,n)           for(i=0;i<n;i++)
#define rep(i,n)          for(i=1;i<=n;i++)
#define rev(i,a,n)        for(i=n;i>=a;i--)
#define FOR(i,a,n)        for(i=a;i<=n;i++)
#define ALL(n)            n.begin(),n.end()
#define mem(x,n)          memset(x,n,sizeof(x));
//int fx[]={+1,-1,+0,+0};
//int fy[]={+0,+0,+1,-1};
//int fx[]={+0,+0,+1,-1,-1,+1,-1,+1};   // Kings Move
//int fy[]={-1,+1,+0,+0,+1,+1,-1,-1};  // Kings Move
//int fx[]={-2, -2, -1, -1,  1,  1,  2,  2};  // Knights Move
//int fy[]={-1,  1, -2,  2, -2,  2, -1,  1}; // Knights Move
#define TC(t)             printf("Case %lld: ",t);
#define ans(t,c)          printf("Case %lld: %lld\n",t,c);
#define SETP(n)           cout<<setprecision(n)<<fixed;
#define READ              freopen("F:\\Project\\Test_Case.txt","r",stdin)
#define WRITE             freopen("F:\\Project\\Output_Test.txt","w",stdout)
#define IO                ios_base::sync_with_stdio(0); cin.tie(0);cout.tie(0);
#define PAIR              pair<ll,ll>
#define MP                make_pair
#define pb                push_back
#define ff                first
#define ss                second
#define NL                printf("\n");
#define bug(a)            cout<<#a<<" "<<a<<" ";
#define hlw               printf("hlw\n");
#define hii               printf("hii\n");
#define NN                111
#define MOD               (ll)1e9+7   /// 10^9+7
#define N                 (ll)1e6+7   ///10^6->6 zero after 1 **
ll x[N],y[N],z[N],n;
string s,S;
vector<ll>v;
//bitset<N>B;
//map <ll,ll> mp;
//priority_queue<ll, vector<ll>, greater<ll> > pq;
int main()
{    //IO;   //while(1)//    READ;WRITE;
{
    ll a=0,b=0,c=0,d,e,f,g,i,j,k,l,m,p,q,r,u,w,t,tc=1;
    ll in,loc,val,sz,lo,hi,mid,mn=MAX,mx=0,sum=0,ans=0;
//cin>>tc;
rep(t,tc)
{
}
}
    return 0;
}
/// Division MOD needs BigMod(a,n-2)
///------------->>>>>       BIT       <<<<<-------------
void update(ll pos,ll val){
    while(pos<=n){x[pos]+=val;  pos+=(pos & -pos);}
}
```

```
ll query(ll pos){
    ll sum=0;
    while(pos){ sum+=x[pos];    pos-=(pos & -pos);}
    return sum;
}
rep(i,n){cin>>a; update(i,a);}    /// 1-based
cout<<query(4)<<" "<<query(2)<<" Ans "<<query(4)-query(2)<<endl;
///------------->>>>>       SPARSE TABLE        <<<<<-------------
ll st[22][N],x[N],logs[N];
void build(ll n){    /// 0
    ll i,j,k;
    logs[1]=0; for(i=2;i<=n;i++)logs[i]= logs[i/2]+1;
    for(i=0;i<n;i++)st[0][i]=x[i];
    for(i=1; (1<<i) <n; i++){
        for(j=0; j+(1<<i)<=n; j++){st[i][j]=min(st[i-1][j], st[i-1][j + (1<<i-1)]);}}
}
ll query(ll l, ll r){
    ll pow = logs[r-l+1];   return min(st[pow][l], st[pow][r-(1<<pow)+1]);
}
build(n);   cout<<query(l,r)<<endl;
///------------->>>>>       SEGMENT TREE        <<<<<-------------
ll tree[4*N],tr[N],lazy[4*N];
void build(ll in,ll L,ll R){
    if(L==R){tree[in]=tr[L];    return;}
    ll mid=(L+R)/2; build(in*2,L,mid);  build(in*2+1,mid+1,R);
    tree[in]=min(tree[in*2],tree[in*2+1]);
}
ll query(ll L,ll R,ll in,ll i,ll j){
    if(j<L||i>R)return MAX; if(L>=i&&j>=R)return tree[in];
    ll p,q,mid=(L+R)/2; p=query(L,mid,in*2,i,j);    q=query(mid+1,R,in*2+1,i,j);
    return min(p,q);
}
void update(ll in,ll L,ll R,ll pos,ll val){
    if(pos>R||L>pos)return;
    if(L==R&&pos==L){tree[in]+=val;     return;}
    ll mid=(L+R)/2;     update(in*2,L,mid,pos,val);     update(in*2+1,mid+1,R,pos,val);
    tree[in]=tree[in*2]+tree[in*2+1];
}
void lazy_update (ll in,ll L,ll R,ll x,ll y,ll val){
    if(x>y)return;
    if(lazy[in]!=0){tree[in]+=lazy[in];
        if(L!=R){
            lazy[in*2]+=lazy[in];   lazy[in*2+1]+=lazy[in];}
        lazy[in]=0;}
    if(x>R || y<L)return;
    if(x<=L && y>=R){
        tree[in]+=val;
        if(L!=R){lazy[in*2]+=val;   lazy[in*2+1]+=val;}
        return;}
    ll mid=(L+R)/2; lazy_update(in*2,L,mid,x,y,val); lazy_update(in*2+1,mid+1,R,x,y,val);
    tree[in]=tree[in*2]+tree[in*2+1];
}
ll lazy_query(ll in,ll L,ll R,ll x,ll y){
    if(x>y)return 0;
    if(lazy[in]!=0){tree[in]+=lazy[in];
        if(L!=R){lazy[in*2]+=lazy[in];  lazy[in*2+1]+=lazy[in];}
        lazy[in]=0;}
    if(x>R || y<L)return 0;
    if(x<=L && y>=R)return tree[in];
    ll p,q,mid=(L+R)/2; p=lazy_query(in*2,L,mid,x,y);   q=lazy_query(in*2+1,mid+1,R,x,y);
    return p+q;
}
build(1,1,n);   cout<<query(1,n,1,a,b)<<endl;   /// Call Function
lazy_update(1,1,n,a,b,c);   cout<<lazy_query(1,n,1,a,b)<<endl;
///------------->>>>>       MATH        <<<<<-------------
ll spf[N]; vector<ll>primes;
void sieve() ///with SPF    {
    for(int i = 2; i < N; i++){if (spf[i] == 0) spf[i] = i, primes.push_back(i);
        int sz = primes.size();
        for (int j=0; j<sz && i*primes[j]<N && primes[j]<=spf[i]; j++)
            spf[i * primes[j]] = primes[j];}
}
int gcd(int a,int b){while(b)a %= b, swap(a, b);return a;}
ll nCr(ll n,ll r){ /// nCr DP
    ll &ret=dp[n][r];if(~ret)return ret;if(n==r)return ret=1;if(r==1)return ret=n;
    return ret=nCr(n-1,r)+nCr(n-1,r-1);
}
ll bigmod(ll n,ll p,ll MOD) {       /// finds n ^ p % MOD
    if(p==0)return 1;ll x=bigmod(n,p/2,MOD);x=(x*x)%MOD;
    if(p%2)x=(x*n)%MOD;return x;
}
ll precal_nCr(ll n, ll r){ /// larger inputs and MOD required
    /// Precal Starts Here
    fact[1] = 1;for(ll i=2; i<n; i++)  fact[i] = (i*fact[i-1])%MOD;
    invfact[n-1] = bigmod(fact[n-1], MOD-2, MOD);
    for (ll i=n-2; i>=0; i--)  invfact[i] = (invfact[i+1]*(i+1))%MOD;
    /// Precal Ends Here
    if (r<0 || r>n) return 0;   return (fact[n]*(invfact[r]*invfact[n-r])%MOD)%MOD;
}
ll binarySearch(ll lo,ll hi,ll key){
    while(lo<=hi){
        ll mid=(lo+hi)/2;
        if(x[mid]==key){ll ans=mid;lo=mid+1;}   else hi=mid-1;}
}
void permutation(string s,int i,int n){
    if(i==n){cout<<s<<endl;return ;}
    for(int j=i;j<=n;j++){swap(s[i],s[j]); permutation(s,i+1,n);}
}
ll mod_inverse(ll a,ll mod){return bigmod(a,mod-2,mod);}
void allPossibleSubset(int n){
    for(ll mask = 0; mask < (1 << n); mask++){ll sum_of_this_subset = 0;
        for(int i = 0; i < n; i++){if(mask & (1 << i))sum_of_this_subset += x[i];}}
}
/// Find numbers of co-prime of N which are less than N
void totient(){
    ll i,j,k;for(i=1;i<=N;i++)phi[i]=i;
    for(i=2;i<=N;i++){
        if(phi[i]==i){
            for(j=i;j<=N;j+=i)
                phi[j]= (phi[j]*(i-1))/i;}}
}
/// Find eulerphi for any numbers with prime pre-calculated
int eulerPhi ( int n ) {
    int res = n;int sqrtn = sqrt ( n );
    for ( int i = 0; i < prime.size() && prime[i] <= sqrtn; i++ ) {
        if ( n % prime[i] == 0 ) {while ( n % prime[i] == 0 ) n /= prime[i];
            sqrtn = sqrt ( n );res /= prime[i];res *= prime[i] - 1;}}
    if ( n != 1 ) {res /= n;res *= n - 1;}
    return res;
}
ll Inclusion_Exclusion(){
    ll a=0,b,c=0,cnt,i,j,k,m,n;cnt=pow(2,m);
    rep(i,cnt-1){a=1;
      fr(j,m){if(i & 1<<j)a=(a*x[j])/__gcd(a,x[j]);}
      a=n/a;b=__builtin_popcountll(i);
      if(b%2)c+=a;else c-=a;}
    return n-c;
}
double Angle(double Ax,double Ay,double Bx,double By,double Cx,double Cy){
    double a1,a2,b1,b2,u,v,p,ang; a1=Ax-Bx; b1=Ay-By; a2=Cx-Bx; b2=Cy-By;p=a1*a2+b1*b2;
    u=sqrt(a1*a1+b1*b1);v=sqrt(a2*a2+b2*b2);ang = acos(p/(u*v));return (ang*180)/acos(0.0);
}
///Calculate Time Complexity
clock_t t1,t2;  double t;   t1=clock();fr(i,10000)fr(j,10000)x[i]=rand();t2=clock();
```

```
t=(t2-t1)/(CLOCKS_PER_SEC); cout<<"Time: "<<t<<endl;
///------------->>>>>       DP       <<<<<-------------
ll LCS(char p[],char q[],int a,int b){
    ///All loop will work through 1 to n/m here...
    int i,j,k;  rep(i,a)x[i][0]=0;  rep(i,b)x[0][i]=0;
    rep(i,a)rep(j,b){
                if(p[i]==q[j])x[i][j]=x[i-1][j-1]+1;    else
x[i][j]=max(x[i][j-1],x[i-1][j]);}
    return x[a][b];
}
ll LIS(ll n){
    ll i,a,in=0,st,en,mid,ans=-1;ar[1]=INT_MIN;
    rep(i,n){a=x[i];
        if(in==0 || a>ar[in])ar[++in]=a;
        else if(a<x[1])ar[1]=a;
        else{st=1,en=in;
            while(st<=en){
                mid=(st+en)/2;  if(ar[mid]<a)st=mid+1;  else en=mid-1;}
            ar[st]=a;}
        //cout<<"i "<<i<<" a "<<a<<" in "<<in<<endl;
        }   return in;
}
///------------->>>>>       GRAPH THEORY      <<<<<-------------
void DFS(int s){
    if(vis[s])return;vis[s]=1;for(int i=0;i<adj[s].size();i++)DFS(adj[s][i]);
}
void BFS(int s){
    int i;mem(vis,0);queue<int>q;q.push(s);vis[s]=1;
    while(!q.empty()){int u=q.front();q.pop();
        fr(i,adj[u].size()){int v=adj[u][i];
            if(!vis[v])q.push(v),vis[v]=1;}}
}
/// DIsjoint Set Union - DSU
void make_set(ll a){par[a]=a;sz[a]=1;}
ll find_par(ll a){if(a==par[a])return a;return par[a]=find_par(par[a]);}
void union_set(ll a,ll b){a=find_par(a);b=find_par(b);if(a==b)return;
    if(sz[a]<sz[b])swap(a,b);par[b]=a;sz[a]+=sz[b]; }
/// Topological Sort-> First top Sort, then DFS, Sort vertices according to path
(Father-Child),Need to be acyclic
void dfs(ll s){vis[s]=1;ll i;
    fr(i, v[s].size()){ll to=v[s][i];
    if(!vis[to])dfs(to);}ans.pb(s);}
void top_sort(){
    mem(vis,0);ans.clear();ll i;fr(i,n)if(!vis[i])dfs(i);reverse(ALL(ans));
}
///Bipartite checking(check if all edges can be divided in two diff sets)
bool bipartite(ll s){ll i,to;
    fr(i, v[s].size()){to=v[s][i];
        if(!vis[to]){vis[to]=1;color[to]=!color[s];
            if(bipartite(to)==false)return false;}
        if(color[s]==color[to])return false;}
    return true;
}
void APSP(int x[V][V]){int i,j,k;
    fr(k,V)fr(i,V)fr(j,V){
                if(graph[i][j] > graph[i][k]+graph[k][j])graph[i][j] =
graph[i][k]+graph[k][j];}
}
/// Dijkstra Function for Single Source Shortest Path
ll minimum(ll dist[],ll tree[]){ /// part of Dijkstra
    int i,min=INF,min_index;
    fr(i,V){if(!tree[i] && dist[i]<min)min=dist[i],min_index=i;}
    return min_index;
}
void Dijkstra(int x[V][V],int s){int u,i,j,k;
    fr(i,V)dist[i]=INF,tree[i]=0;dist[s]=0;
    fr(i,V){        ///Find Minimum
```

```
        u=minimum(dist,tree);tree[u]=1;

        fr(k,V){    ///Relaxation
            if(!tree[k] && dist[k]!=INF && graph[u][k] && dist[k]>dist[u]+graph[u][k])
                dist[k] = dist[u]+graph[u][k];}}
}
///Bellman Ford Algo for SPSP (Can work with neg-weight)
struct edg{int u,v,w;};vector<edg>edge;edg e;
void BellFord(int graph[][V],int s){
    int i,j,k;  fr(i,V)dist[i]=INF;dist[s]=0;
    fr(j,V-1)   ///Relaxation with Edges
        fr(i,edge.size()){
            if(dist[edge[i].v] > dist[edge[i].u]+edge[i].w)edge[i].v =
dist[edge[i].u]+edge[i].w;}
}
/// Prims Algo for Minimum Spanning Tree
int printMST(int parent[], int n, int graph[V][V]){printf("Edge   Weight\n");
   for (int i = 1; i < V; i++)printf("%d - %d    %d \n", parent[i], i, graph[i][parent[i]]);
}
void Prims(int graph[V][V]){int i,j,u;ll tree[V],dist[V],parent[V];
    fr(i,V)dist[i]=INF,tree[i]=0;dist[0]=0,tree[0]=-1;
    for(j=0;j<V-1;j++){u=minimum(dist,tree);tree[u]=1;
        fr(i,V){
            if(!tree[i] && graph[u][i] &&
graph[u][i]<dist[i])dist[i]=graph[u][i],parent[i]=u;}
    }//  printMST(parent, V, graph);
}
///   Articulation Graph
set<ll>ans;
void DFS(ll in,ll par){
    en[in]=mn[in]=cnt++;vis[in]=1;ll p=0,a,i,l=v[in].size();
    fr(i,l){
        ll to=v[in][i];if(to==-1)continue;
        if(!vis[to]){
            DFS(to,in);p++;mn[in]=min(mn[in],mn[to]);
            if(par!=-1 && en[in]<=mn[to])ans.insert(in);}
        else mn[in]=min(mn[in],en[to]);}
    if(par==-1 && p>1)ans.insert(in);
}
    rep(i,n){if(!vis[i])DFS(i,-1);} /// Call Function
///

///------------->>>>>     LCA by SPARSE TABLE    <<<<<-------------
void walk(ll s, ll d){
    ll i, in;   last[s]=k;  nodes[k]=s; depth[k++]=d;
    fr(i,v[s].size()){in=v[s][i];if(vis[in])continue;vis[in]=1;
        walk(in,d+1);nodes[k]=s;depth[k++]=d;}
}
void sparse_table(ll n){/// 0 based indexing
    ll node_a,node_b,i,j,k;for(i=0;i<n;i++)st[0][i]=i;
    for(i=1; (1<<i) <n; i++){for(j=0; j+(1<<i)<=n; j++){node_a=st[i-1][j];node_b=st[i-1][j
+ (1<<i-1)];
            st[i][j] = depth[node_a]<=depth[node_b]? node_a:node_b;}}
}
ll LCA(ll l,ll r){
    l=last[l],r=last[r];if(l>r)swap(l,r);ll pow = log2(r-l+1);ll a,b;
    a=st[pow][l];   b=st[pow][r-(1<<pow)+1];return nodes[depth[a]<=depth[b]? a:b];
}
int main(){     /// Code for LCA. [0 based indexing]
    vis[0]=1;walk(0,0);sparse_table(2*n-1);cin>>a>>b;   cout<<LCA(a-1,b-1)+1<<endl;}

///------------->>>>>       STRING        <<<<<-------------
unsigned bernstein_hash ( void *key, int len ){
    unsigned char *p = key; unsigned h = 0; int i;
    for ( i = 0; i < len; i++ )h = 33 * h + p[i];
    return h;
}
```

```
/// string matching
vector<int> rabin_karp_HASH(string const& s, string const& t) {
const int p = 31,const int m = 1e9 + 9;int S = s.size(), T = t.size();
vector<long long> p_pow(max(S, T));vector<long long> h(T + 1, 0);
long long h_s = 0;vector<int> occurences;p_pow[0] = 1;
for (int i = 1; i < (int)p_pow.size(); i++)p_pow[i] = (p_pow[i-1] * p) % m;
for (int i = 0; i < T; i++)h[i+1] = (h[i] + (t[i] - 'a' + 1) * p_pow[i]) % m;
for (int i = 0; i < S; i++)h_s = (h_s + (s[i] - 'a' + 1) * p_pow[i]) % m;
for (int i = 0; i + S - 1 < T; i++){long long cur_h = (h[i+S] + m - h[i]) % m;
    if (cur_h == h_s * p_pow[i] % m)occurences.push_back(i);}
return occurences;
}
///KMP with LPS (find pattern)
void LPS(){
    ll i,j,l=pat.size();i=0,j=-1;   lps[i]=j;
    while(i<l){
        while(pat[i]!=pat[j] && j>=0)j=lps[j];
        i++,j++;lps[i]=j;}
}
ll KMP(string txt){
    pat=txt; reverse(ALL(pat));
    LPS(pat);
    ll i,j,n,m;n=txt.size();   m=pat.size();i=j=0;
    while(i<n){
        while(j>=0 && txt[i]!=pat[j])j=lps[j];
        i++,j++;}
    return j;
}
string sub_pal(string s){ ///Find Prefix Sub Palindrome Linear
    string a = s;   reverse(a.begin(), a.end());
    a = s + "#" + a;     ll c = 0,pref[99]={0};
    for (int i = 1; i < (int)a.size(); i++){
        while (c != 0 && a[c] != a[i])c = pref[c - 1];
        if (a[c] == a[i])c++;   pref[i] = c;}
    return s.substr(0, c);
}
string Manacher(string s){  /// longest subpalindrome
    string T="#";// Transform S to T
    for(int i=0;i<s.size();i++)T+=s.substr(i,1)+"#";
    int P[T.size()+5]={0}; // Array to record longest palindrome
    int center=0,boundary=0,maxLen=0,resCenter=0;
    for(int i=1;i<T.size()-1;i++){int iMirror=2*center-i;
        if(i<boundary)P[i]=min(boundary-i,P[iMirror]);
        while(i-1-P[i]>=0 && i+1+P[i]<=T.size()-1 && T[i+1+P[i]]==T[i-1-P[i]])P[i]++;
        if(i+P[i]>boundary){center = i;boundary = i+P[i];}
        if(P[i]>maxLen){maxLen = P[i];resCenter = i;}
    }return s.substr((resCenter - maxLen)/2, maxLen);
}
vector<int>z_algo(string s){  /// finds all occurrences of a pattern linear
    int i,l,r,n;    n=s.length();   vector<int> z(n);
    for (i = 1, l = 0, r = 0; i < n; ++i){
        if (i <= r)z[i] = min (r - i + 1, z[i - l]);
        while (i + z[i] < n && s[z[i]] == s[i + z[i]]) ++z[i];
        if (i + z[i] - 1 > r)l = i, r = i + z[i] - 1;
    }return z;
}
/* author : s@if */
#include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds; using namespace std;
#define         NIL             -1
#define         INF             1e9
#define          EPS             1e-9
#define          SAIF            main
#define         fi              first
#define         sec             second
#define         MAX             INT_MAX
```

```
#define         ll              long long
#define         PI              acos(-1.0)
#define         MOD             1000000007
#define         PLL             pair<ll,ll>
#define         PII             pair<int,int>
#define         ull             unsigned long long
#define         For(i,a,b)      for(int i=a;i<=(int)b;i++)
typedef tree<int, null_type, less<int>, rb_tree_tag, tree_order_statistics_node_update>
new_data_set;
//*find_by_order(k)  gives the kth element;
//order_of_key(item)  gives the index(number of element strictly less than item) of item;
inline int in() {int x; scanf("%d", &x); return x; }
bool Check(int N , int pos)    {   return (bool) (N & (1<<pos));}
int Set(int N, int pos) {  return N = N | (1<<pos);}
int fx[]={+0,+0,+1,-1,-1,+1,-1,+1};    // King's move
int fy[]={-1,+1,+0,+0,+1,+1,-1,-1};
int hx[]={-2,-2,-1,+1,+2,+2,-1,+1};    // Knight's move
int hy[]={+1,-1,+2,+2,-1,+1,-2,-2};
int dx[]={+1,-1,+0,+0};
int dy[]={+0,+0,+1,-1};
const int MAXN = (int)2e5+9;
// Hashing
ll base = 247, M = 1000000007;  ll Hash[MAXN], power[MAXN], L;
void init(void)  { power[0] = 1; Hash[0] = 0;
 for(int i=1; i<MAXN; i++){power[i] = (power[i-1]*base)%M;  }
}
void Hashing(string s){ L = s.size();   ll h = 0;
for(int i=1; i<=L; i++) { ll tmp = (h*base)%M; tmp = (tmp+s[i-1]-'a'+1)%M; Hash[i] = h =
tmp;   return; }
ll HashOf(string p) {  int l = p.size();  ll h = 0;
 for(int i=1; i<=l; i++){ ll tmp = (h*base)%M;  tmp = (tmp+p[i-1]-'a'+1)%M;  h = tmp;  }
return h; }
ll HashOfSubstring(int l, int r){  ll a, b, ret; a = Hash[l-1], b = Hash[r];  a =
(a*power[r-l+1])%M;                     ret = (b-a+M)%M;  return ret;}
int FindPattern(string p){ int i, l = p.size(); ll h1 = HashOf(p);
for(i=1; i<=L-l+1; i++) { int x = i, y = i+l-1; ll h2 = HashOfSubstring(x, y); if(h1==h2)
return i-1; } return -1; }
// trie
struct node{ bool mark; node *next[30]; node() { mark=false; for(int i=0;i<26;i++) {
next[i]=NULL;  }  } }; node *root;
void add(string s){int l=s.size(); node *curr=root; for(int i=0;i<l;i++) { int id=s[i]-'a';
 if(curr->next[id]==NULL)  curr->next[id]=new node(); curr=curr->next[id]; }
curr->mark=true;}
bool _search(string s){  int l=s.size(); node *curr=root;
 for(int i=0;i<l;i++)  {int id=s[i]-'a';  if(curr->next[id]==NULL) curr->next[id]=new
node();
curr=curr->next[id];  }  return curr->mark; }
void del(node *curr){  for(int i=0;i<26;i++)  {if(curr->next[i]) del(curr->next[i]);
delete(curr);}
// KMP
void kmp(string T, string P){  int n=strlen(T);  int m=strlen(P);  int pi[m+9], i, now;
now=pi[0]=-1; for(i=1;i<m;i++) { while(now!=-1 && P[now+1]!=P[i]) { now=pi[now]; }
if(P[now+1]==P[i]) pi[i]= ++now; else pi[i]=now=-1;  } int cnt=0;  now=-1;
 for(i=0;i<n;i++) {while(now!=-1 && P[now+1]!=T[i]) {now=pi[now];  }  if(P[now+1]==T[i])
now++;
 else  now=-1;  if(now==m-1){ cnt++;  now=pi[now]; }  printf("Case %d: %d\n",++t,cnt);
return;}
// Articulation Point
int vis[MAXN], d[MAXN], low[MAXN], art[MAXN], Tm; vector<int>adj[MAXN];
void init(int n){ for(int i=0; i<=n; i++) { vis[i] = 0; art[i] = 0, Tm = 0;
adj[i].clear(); } }
void find_articulation_point(int u){ Tm++; d[u] = low[u] = Tm;  vis[u] = 1; int child = 0;
 for(int i=0; i<adj[u].size(); i++) { int v = adj[u][i]; if(vis[v]==1) { low[u] =
min(low[u], d[v]); }
 else { child++; find_articulation_point(v);  low[u] = min(low[u], low[v]);
if(d[u]<=low[v] && u!=1) art[u] = 1;  } } if(u==1 && child>1) art[u] = 1;}
// SCC
```

```cpp
vector<int>component[MAXN]; vector<int>g[MAXN]; vector<int>rev[MAXN]; stack<int>stk; int
n,mark; int vis[MAXN];
void dfs1(int cur){ vis[cur]=1; for(int i=0;i<g[cur].size();i++) { int v=g[cur][i];
if(!vis[v]) dfs1(v); } stk.push(cur); }
void dfs2(int cur,int mark){ vis[cur]=1; component[mark].push_back(cur);
for(int i=0;i<rev[cur].size();i++) { int v=rev[cur][i]; if(!vis[v]){ dfs2(v,mark); } } }
void SCC(void){  cin>>n>>m; while(m--) { cin>>u>>v; g[u].push_back(v);
rev[v].push_back(u); }
memset(vis,0,sizeof(vis));
for(i=1;i<=n;i++) if(!vis[i]) dfs1(i); memset(vis,0,sizeof(vis)); mark=0;
while(!stk.empty()){ u=stk.top(); stk.pop(); if(!vis[u]){ dfs2(u,++mark);} }
for(i=1;i<=mark;i++){ cout<<"component "<<i<<" : ";
for(j=0;j<component[i].size();j++) cout<<component[i][j]<<" "; cout<<endl; } cout<<endl;}
//LCA
int L[mx],  P[mx][22], T[mx]; vector<int>g[mx];
void dfs(int from,int u,int dep){ T[u]=from; L[u]=dep;  for(int
i=0;i<(int)g[u].size();i++) { int v=g[u][i];  if(v==from) continue;  dfs(u,v,dep+1);    }  }
int lca_query(int N, int p, int q){ int tmp, log, i;  if (L[p] < L[q])  tmp = p, p = q, q
= tmp;
  log=1;  while(1) { int next=log+1;  if((1<<next)>L[p])break;  log++;  }
for (i = log; i >= 0; i--)  if (L[p] - (1 << i) >= L[q])  p = P[p][i];   if (p == q)
return p;
for (i = log; i >= 0; i--)  if (P[p][i] != -1 && P[p][i] != P[q][i]) p = P[p][i], q =
P[q][i];  return T[p]; }
void lca_init(int N) {  memset (P,-1,sizeof(P));  int i, j;
 for (i = 0; i < N; i++) P[i][0] = T[i];
 for (j = 1; 1 << j < N; j++) for (i = 0; i < N; i++) if (P[i][j - 1] != -1) P[i][j] =
P[P[i][j - 1]][j - 1]; }
// Discrete Logarithm
ll Discrete_Log(ll a, ll b, ll m)
{ if(a==0) { if(b==0) return 1; else return -1; }
a%=m, b%=m; ll g, k = 1, add = 0;
while((g=__gcd(a,m))>1) { if(b==k) return add; if(b%g) return -1;
b/=g, m/=g; ++add; k = (k*a/g)%m;}
map<ll,ll>Map; ll n = sqrt(m)+1;
for(ll q=0, curr=b; q<=n; q++){ Map[curr] = q; curr = (curr*a)%m; }
ll an = 1; for(ll p=1; p<=n; p++) an = (an*a)%m;
for(ll p=1, curr=k; p<=n; p++){ curr = (curr*an)%m; if(Map[curr]) return n*p-Map[curr]+add;}
return -1; }
void solve(void)
{
    ll a, b, i,j,k,l,m,n,p,q,x,y,u,v,w,r,tc,t;  return;
}
int SAIF()
{
    int tc, t = 0; cin>>tc; while(tc--) solve(); return 0;
}
// read the question correctly (is y a vowel? what are the exact constraints?)
// look out for SPECIAL CASES (n=1?) and overflow (ll vs int?)
/*Farhad*/
#include<bits/stdc++.h>
// #include<ext/pb_ds/assoc_container.hpp>
// #include<ext/pb_ds/tree_policy.hpp>
// using namespace __gnu_pbds;
// template<typename T>
// using ordered_set = tree <
//                          T,
//                          null_type,
//                          less<T>,
//                          rb_tree_tag,
//                          tree_order_statistics_node_update >;
//ordered_set<int> S;
using namespace std;
#define     flash           ios_base::sync_with_stdio(false);cin.tie(0);
#define     ff              first
#define     ss              second
#define     pb              push_back
```

```
#define    m_p              make_pair
//#define    ret              return 0
#define    MAX(a,b)         max({a,b})
#define    MAX(a,b,c)       max({a,b,c})
#define    MAX(a,b,c,d)     max({a,b,c,d})
#define    MIN(a,b)         min({a,b})
#define    MIN(a,b,c)       min({a,b,c})
#define    MIN(a,b,c,d)     min({a,b,c,d})
#define    GCD(a,b)         __gcd(a,b)
#define    LCM(a,b)         (a*b)/GCD(a,b)
#define    MEM(a,b)         memset(a,b,sizeof a)
#define    SC(a)            scanf("%d",&a)
#define    SC2(a,b)         scanf("%d %d",&a,&b)
#define    SC3(a,b,c)       scanf("%d %d %d",&a,&b,&c)
#define    SC4(a,b,c,d)     scanf("%d %d %d %d",&a,&b,&c,&d)
#define    PCs(a)           printf("Case %d: ",a)
#define    WRITE(a)         freopen(a,"w",stdout)
#define    READ(a)          freopen(a,"r",stdin)
#define    LB(a,x)          (lower_bound(a.begin(),a.end(),x) - a.begin())
#define    UB(a,x)          (upper_bound(a.begin(),a.end(),x) - a.begin())
#define    PI               2.0*acos(0.0)
#define    MOD1             1000000007 // prime
#define    MOD2             1000000009 // prime
#define    MOD3             1000000021 // prime
#define    Base1            10000019
#define    Base2            10000079
#define    Base3            10000103
#define    endl              '\n'
typedef     pair<int, int> pii;
typedef     pair<int, pii> ppi;
typedef     pair<pii, int> pip;
typedef     long long int ll;
typedef     unsigned long long int ull;
typedef     pair<ll, ll> pll;
typedef     vector<int> VI;
typedef     vector<pii> Vii;
typedef     vector<VI> VVI;// 2D
//typedef    priority_queue<int> PQ;// MaxHeap
typedef     priority_queue<int, VI, greater<int> > PQ; // MinHeap
/*inline int StringToInt(String a){int num;StringSeam aw(a);aw>>num;return num;}*/
/*inline ll StringToLL(String a){ll num;StringSeam aw(a);aw>>num;return num;}*/
//Math
/*inline int iPOW(int a,int e){int num=1;while(e){if(e%2){num=num * a;}e/=2;a=a *
a;}return num;}*/
/*inline ll LPOW(ll a,ll e){ll num=1;while(e){if(e%2){num=num * a;}e/=2;a=a * a;}return
num;}*/
/*inline ll BigMod(ll a,ll e,ll mod){ll num=1;while(e){if(e%2){a%=mod;num%=mod;num=num *
a;num%=mod;}e/=2;a%=mod;a=a * a;a%=mod;}return num%mod;}*/
/*inline ll modInverse(ll A,ll P){return BigMod(A,P-2,P);}ll fac[MAX];
inline void factorial(int n,int mod){fac[0]=1;fac[1]=1;for(int i=2;i<=MAX;i++) fac[i]=(
(fac[i-1]%mod)*i)%mod;}
inline int nCr(int n,int r,int mod) // ncr with mod{return ((fac[n] *
modInverse(fac[r],mod)%mod)%mod * (modInverse(fac[n-r],mod)%mod) ) %mod;}*/
/*bool isprime[MAX+1000];int Primes[MAX],id;// for <=10^6
void sieve(){Primes[0]=2;id++;for(int i=4;i<=MAX+100;i+=2) isprime[i]=true;// is not a
primefor(int
i=3;i<=MAX+100;i+=2){if(isprime[i]==false){Primes[id++]=i;if(i+i<=MAX)for(int
j=i+i;j<=100+MAX;j+=i)isprime[j]=true;}}}*/
//bigint
/*code from arpa
overloaded operators:
EQUAL::::::::::::: = (bigint), = (long long) , == (bigint) ,!=(bigint)
ADD::::::::::::::: + (bigint) , += (bigint)
SUB::::::::::::::: - (bigint) , -= (bigint),
MUL::::::::::::::: *=(int) , *(int),*(long long) , *(bigint) , *=(long long) ,*= (bigint)
DIV::::::::::::::: / (int) , / (bigint) , /= (int) ,/= (bigint),
MOD::::::::::::::: %(int), % (bigint),
COMPARE:::::::::: < (bigint) ,> (bigint) ,<= (bigint) ,>= (bigint) ,
ABS:::::::::::::: -() (bigint)
POW:::::::::::::: ^ (bigint);
functions: size() , returns size
to_string() , converts to string
sumof() , returns sum of digits
divmod() , dunno what it does
trim() , trims trailing zeroes
isZero() , zero or not
abs() , absolute value
longValue() , to long
gcd(a,b) , gcd
lcm(a,b) , lcm
convert_base() , converts base
karatsubaMultiply(const vll &a, const vll &b) , dunno what it does
*/
#include<bits/stdc++.h>
using namespace std;
const int base = 1000000000, base_digits = 9;
struct bigint {
    vector<int> a; int sign;
    int size() {if (a.empty())return 0; int ans = (a.size() - 1) * base_digits; int ca =
a.back(); while (ca)ans++, ca /= 10; return ans;}
    bigint operator ^(const bigint &v) {bigint ans = 1, a = *this, b = v; while
(!b.isZero()) {if (b % 2)ans *= a; a *= a, b /= 2;} return ans;}
    string to_string() {stringstream ss; ss << *this; string s; ss >> s; return s;}
    int sumof() {string s = to_string(); int ans = 0; for (auto c : s)  ans += c - '0';
return ans;}
    bigint() : sign(1) {}
    bigint(long long v) {*this = v;}
    bigint(const string &s) {read(s);}
    void operator=(const bigint &v) {sign = v.sign; a = v.a;}
    void operator=(long long v) {sign = 1; a.clear(); if (v < 0)sign = -1, v = -v; for (;
v > 0; v = v / base)a.push_back(v % base);}
    bigint operator+(const bigint &v) const {if (sign == v.sign) {bigint res = v; for (int
i = 0, carry = 0; i < (int) max(a.size(), v.a.size()) || carry; ++i) {if (i == (int)
res.a.size())res.a.push_back(0); res.a[i] += carry + (i < (int) a.size() ? a[i] : 0);
carry = res.a[i] >= base; if (carry)res.a[i] -= base;} return res;} return *this - (-v);}
    bigint operator-(const bigint &v) const {if (sign == v.sign) {if (abs() >= v.abs())
{bigint res = *this; for (int i = 0, carry = 0; i < (int) v.a.size() || carry; ++i)
{res.a[i] -= carry + (i < (int) v.a.size() ? v.a[i] : 0); carry = res.a[i] < 0; if
(carry)res.a[i] += base;} res.trim(); return res;} return -(v - *this);} return *this +
(-v);}
    void operator*=(int v) {if (v < 0)sign = -sign, v = -v; for (int i = 0, carry = 0; i <
(int) a.size() || carry; ++i) {if (i == (int) a.size())a.push_back(0); long long cur =
a[i] * (long long) v + carry; carry = (int) (cur / base); a[i] = (int) (cur %
base);/*//asm("divl %%ecx" : "=a"(carry), "=d"(a[i]) : "A"(cur), "c"(base));*/} trim();}
    bigint operator*(int v) const {bigint res = *this; res *= v; return res;}
    void operator*=(long long v) {if (v < 0)sign = -sign, v = -v; for (int i = 0, carry =
0; i < (int) a.size() || carry; ++i) {if (i == (int) a.size())a.push_back(0); long long
cur = a[i] * (long long) v + carry; carry = (int) (cur / base); a[i] = (int) (cur %
base);/*//asm("divl %%ecx" : "=a"(carry), "=d"(a[i]) : "A"(cur), "c"(base));*/} trim();}
    bigint operator*(long long v) const {bigint res = *this; res *= v; return res;}
    friend pair<bigint, bigint> divmod(const bigint &a1, const bigint &b1) {int norm =
base / (b1.a.back() + 1); bigint a = a1.abs() * norm; bigint b = b1.abs() * norm; bigint q,
r; q.a.resize(a.a.size()); for (int i = a.a.size() - 1; i >= 0; i--) {r *= base; r +=
a.a[i]; int s1 = r.a.size() <= b.a.size() ? 0 : r.a[b.a.size()]; int s2 = r.a.size() <=
b.a.size() - 1 ? 0 : r.a[b.a.size() - 1]; int d = ((long long) base * s1 + s2) /
b.a.back(); r -= b * d; while (r < 0)r += b, --d; q.a[i] = d;} q.sign = a1.sign * b1.sign;
r.sign = a1.sign; q.trim(); r.trim(); return make_pair(q, r / norm);}
    bigint operator/(const bigint &v) const {return divmod(*this, v).first;}
    bigint operator%(const bigint &v) const {return divmod(*this, v).second;}
    void operator/=(int v) {if (v < 0)sign = -sign, v = -v; for (int i = (int) a.size() -
1, rem = 0; i >= 0; --i) {long long cur = a[i] + rem * (long long) base; a[i] = (int) (cur
/ v); rem = (int) (cur % v);} trim();}
    bigint operator/(int v) const {bigint res = *this; res /= v; return res;}
    int operator%(int v) const {if (v < 0)v = -v; int m = 0; for (int i = a.size() - 1; i
>= 0; --i)m = (a[i] + m * (long long) base) % v; return m * sign;}
    void operator+=(const bigint &v) {*this = *this + v;}
    void operator-=(const bigint &v) {*this = *this - v;}
    void operator*=(const bigint &v) {*this = *this * v;}
    void operator/=(const bigint &v) {*this = *this / v;}
    bool operator<(const bigint &v) const {if (sign != v.sign)return sign < v.sign; if
(a.size() != v.a.size())return a.size() * sign < v.a.size() * v.sign; for (int i =
a.size() - 1; i >= 0; i--)if (a[i] != v.a[i])return a[i] * sign < v.a[i] * sign; return
false;}
    bool operator>(const bigint &v) const {return v < *this;}
    bool operator<=(const bigint &v) const {return !(v < *this);}
    bool operator>=(const bigint &v) const {return !(*this < v);}
    bool operator==(const bigint &v) const {return !(*this < v) && !(v < *this);}
    bool operator!=(const bigint &v) const {return *this < v || v < *this;}
    void trim() {while (!a.empty() && !a.back())a.pop_back(); if (a.empty())sign = 1;}
    bool isZero() const {return a.empty() || (a.size() == 1 && !a[0]);}
    bigint operator-() const {bigint res = *this; res.sign = -sign; return res;}
    bigint abs() const {bigint res = *this; res.sign *= res.sign; return res;}
    long long longValue() const {long long res = 0; for (int i = a.size() - 1; i >= 0;
i--)res = res * base + a[i]; return res * sign;}
    friend bigint gcd(const bigint &a, const bigint &b) {return b.isZero() ? a : gcd(b, a
% b);}
    friend bigint lcm(const bigint &a, const bigint &b) {return a / gcd(a, b) * b;}
    void read(const string &s) {sign = 1; a.clear(); int pos = 0; while (pos < (int)
s.size() && (s[pos] == '-' || s[pos] == '+')) {if (s[pos] == '-')sign = -sign; ++pos;} for
(int i = s.size() - 1; i >= pos; i -= base_digits) {int x = 0; for (int j = max(pos, i -
base_digits + 1); j <= i; j++)x = x * 10 + s[j] - '0'; a.push_back(x);} trim();}
    friend istream& operator>>(istream &stream, bigint &v) {string s; stream >> s;
v.read(s); return stream;}
    friend ostream& operator<<(ostream &stream, const bigint &v) {if (v.sign == -1)stream
<< '-'; stream << (v.a.empty() ? 0 : v.a.back()); for (int i = (int) v.a.size() - 2; i >=
0; --i)stream << setw(base_digits) << setfill('0') << v.a[i]; return stream;}
    static vector<int> convert_base(const vector<int> &a, int old_digits, int new_digits)
{vector<long long> p(max(old_digits, new_digits) + 1); p[0] = 1; for (int i = 1; i < (int)
p.size(); i++)p[i] = p[i - 1] * 10; vector<int> res; long long cur = 0; int cur_digits =
0; for (int i = 0; i < (int) a.size(); i++) {cur += a[i] * p[cur_digits]; cur_digits +=
old_digits; while (cur_digits >= new_digits) {res.push_back(int(cur % p[new_digits])); cur
/= p[new_digits]; cur_digits -= new_digits;}} res.push_back((int) cur); while
(!res.empty() && !res.back())res.pop_back(); return res;}
    typedef vector<long long> vll;
    static vll karatsubaMultiply(const vll &a, const vll &b) {int n = a.size(); vll res(n
+ n); if (n <= 32) {for (int i = 0; i < n; i++)for (int j = 0; j < n; j++)res[i + j] +=
a[i] * b[j]; return res;} int k = n >> 1; vll a1(a.begin(), a.begin() + k); vll
a2(a.begin() + k, a.end()); vll b1(b.begin(), b.begin() + k); vll b2(b.begin() + k,
b.end()); vll a1b1 = karatsubaMultiply(a1, b1); vll a2b2 = karatsubaMultiply(a2, b2); for
(int i = 0; i < k; i++)a2[i] += a1[i]; for (int i = 0; i < k; i++)b2[i] += b1[i]; vll r =
karatsubaMultiply(a2, b2); for (int i = 0; i < (int) a1b1.size(); i++)r[i] -= a1b1[i]; for
(int i = 0; i < (int) a2b2.size(); i++)r[i] -= a2b2[i]; for (int i = 0; i < (int)
r.size(); i++)res[i + k] += r[i]; for (int i = 0; i < (int) a1b1.size(); i++)res[i] +=
a1b1[i]; for (int i = 0; i < (int) a2b2.size(); i++)res[i + n] += a2b2[i]; return res;}
    bigint operator*(const bigint &v) const {vector<int> a6 = convert_base(this->a,
base_digits, 6); vector<int> b6 = convert_base(v.a, base_digits, 6); vll a(a6.begin(),
a6.end()); vll b(b6.begin(), b6.end()); while (a.size() < b.size())a.push_back(0); while
(b.size() < a.size())b.push_back(0); while (a.size() & (a.size() - 1))a.push_back(0),
b.push_back(0); vll c = karatsubaMultiply(a, b); bigint res; res.sign = sign * v.sign; for
(int i = 0, carry = 0; i < (int) c.size(); i++) {long long cur = c[i] + carry;
res.a.push_back((int) (cur % 1000000)); carry = (int) (cur / 1000000);} res.a =
convert_base(res.a, 6, base_digits); res.trim(); return res;}
};
// euler totient
//int phi[MAX], mark[MAX];
void func() {for (int i = 1; i < MAX; i++)phi[i] = i; phi[1] = 1; /* should be
defined*/mark[1] = 1; for (int i = 2; i < MAX; i++) {if (!mark[i]) {for (int j = i; j <
MAX; j += i) {mark[j] = 1; phi[j] = phi[j] / i * 1LL * (i - 1);}}}}
//fraction
class fraction {
public:
    ll nom, denom;
    fraction() {nom = denom = 0;} fraction(ll x) {nom = x , denom = 1;}
    fraction(ll x, ll y) {nom = x , denom = y;}
    void norm() {ll g = __gcd(nom, denom); nom /= g; denom /= g; if (nom == 0)denom = 1;
if (denom < 0)denom *= -1, nom *= -1;}
    fraction operator + (fraction obj) {ll lc = lcm(obj.denom, denom); fraction r ( nom *
(lc / denom) + obj.nom * (lc / obj.denom), lc ); r.norm(); return r;}
    fraction operator - (fraction obj) {ll lc = lcm(obj.denom, denom); fraction r ( nom *
(lc / denom) - obj.nom * (lc / obj.denom), lc ); r.norm(); return r;}
    fraction operator * (ll x) {return {nom * x , denom};}
    void print() {cerr << nom << "/" << denom << endl;}
};
// factorial-factorising
vector<pii> factfactorise(int n){vector<pii> F;for(int i=0;i<id&&primes[i]<=n;i++){ll curr
= primes[i];ll num = n ;ll cnt = 0 ;while( num / curr){cnt +=
num/curr;curr*=primes[i];}if(cnt)F.push_back({primes[i] , cnt});}return F;}
// printing r elements from n
/*int r,k;cin>>k>>r;vector<int>a(k);for(int
i=0;i<k;i++)cin>>a[i];vector<bool>v(k);//fill(v.end()-r,v.end(),true);
fill(v.begin(),v.begin()+r,true);do{vector<int>res;for(int
i=0;i<k;i++){if(v[i])res.push_back(a[i]);}for(int i=0;i<r;i++){if(i)cout<<'
';cout<<res[i];}cout<<'\n';}while(prev_permutation(v.begin(),v.end()));*/
// BIT
ll BIT[MAX],N;void update(int i,int x){for(;i<=N;i+=i&-i)BIT[i]+=x;}ll query(int i){ll
res=0;for(;i>=1;i-=i&-i)res+=BIT[i];return res;}
// 2D BIT
int BIT[MAX][MAX];void update(int x,int y,int val){while(x<=MAX){int
y_=y;while(y_<=MAX){BIT[x][y_]+=val;y_+=(y_&-y_);}x+=(x&-x);}}
int query(int x,int y){int res=0;while(x>=1){int
y_=y;while(y_>=1){res+=BIT[x][y_];y_-=(y_&-y_);}x-=(x&-x);}return res;}
// DSU
int par[MAX],sz[MAX];void init(int n){for(int i=1;i<=n;i++)par[i]=i,sz[i]=1;}int
find_par(int x){if(par[x]==x)return x;return par[x] = find_par(par[x]);}
void Union(int u,int v){int par_u = find_par(u);int par_v =
find_par(v);if(par_u!=par_v){if(sz[par_u]>sz[par_v]){sz[par_u]+=sz[par_v];par[par_v] =
par_u;}else{sz[par_v]+=sz[par_u];par[par_u] = par_v;}}}
// segment tree
int A[MAX];pair<int,int> TREE[3*MAX];void build(int node,int l,int
r){if(l==r){TREE[node].first=A[l];TREE[node].second=1;return;}int mid =
(l+r)/2;build(2*node  ,l    ,mid);build(2*node+1,mid+1,r
);if(TREE[node*2].first<TREE[node*2+1].first){TREE[node]=TREE[node*2];}else
if(TREE[node*2].first>TREE[node*2+1].first){TREE[node]=TREE[node*2+1];}else{TREE[node].first
=TREE[node*2].first;TREE[node].second=TREE[node*2].second+TREE[node*2+1].second;}}
void update(int node,int l,int r,int
pos){if(l>pos||r<pos)return;if(l==r){TREE[node]={A[l],1};return;}int mid =
(l+r)/2;if(pos<=mid)update(2*node  ,l    ,mid,pos);else update(2*node+1,mid+1,r
,pos);if(TREE[node*2].first<TREE[node*2+1].first){TREE[node]=TREE[node*2];}else
if(TREE[node*2].first>TREE[node*2+1].first){TREE[node]=TREE[node*2+1];}else{TREE[node].first
=TREE[node*2].first;TREE[node].second=TREE[node*2].second+TREE[node*2+1].second;}}
pair<int,int> query(int node,int l,int r,int L,int R){if(l>R||r<L)return
{1e9+7,0};if(l>=L&&r<=R){return TREE[node];}int mid = (l+r)/2;pair<int,int>x,y,ret;x =
query(2*node  ,l    ,mid,L,R);y = query(2*node+1,mid+1,r
,L,R);if(x.first<y.first){ret=x;}else
if(x.first>y.first){ret=y;}else{ret.first=x.first;ret.second=x.second+y.second;}return ret;}
//int Dx[] ={-1,0, 0,1};int Dy[] ={ 0,1,-1,0};int Dx8[]={-1,-1,-1,0,1,1, 1, 0};int
Dy8[]={-1, 0, 1,1,1,0,-1,-1};int Kx[] ={2,1,-1,-2,-2,-1, 1, 2};int Ky[] ={1,2, 2,
1,-1,-2,-2,-1};
//code starts from here
//const int MAX=1e5+10,MOD=1e9+7;
int main ()
{
    //flash;
    return 0;
}
```