```
//ORDERED SET
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
template <typename T>
using indexed_set = tree<T, null_type, less<T>,
rb_tree_tag, tree_order_statistics_node_update>;
```

```
// PBDS (Policy Based Data Structure)
// Ordered Set
// Delete them
//directory change:
//C:\MinGW\lib\gcc\mingw32\6.3.0\include\c++\ext\
pb_ds\detail\resize_policy
//.hpp er porer number gula delete korte hobe
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
template<class T> using oset = tree<T, null_type,
less<T>, rb_tree_tag,
tree_order_statistics_node_update>;
//oset <ll> s;  --> Declare ordered set
//s.order_of_key(val)  --> index of value val
//*(s.find_by_order(ind)) --> value at index ind
```

```
const int limit = 1e7+7;
//Sieve of Eratosthenes
//TimeComplexity O(nloglogn)
//canbeuseduntil10^9
vector<bool> is_prime(limit+1,true);
void sieve_of_eratosthenes(){
    //Finding out the primes in simple way
is_prime[0] = is_prime[1] = false;
    for(int i=2;i*i<=limit;++i){
        if (is_prime[i]) {
            primes.push_back(i);
            for(intj=i*i;j<=limit;j+=i){
                is_prime[j]=false;
            }
        }
    }
}
```

```
const int mx = 1e8 + 9; //max value of n
const int mxprm = 6e6 + 9; //max number
//Sieve Eratosthenes Bitset
int psz = 0; //count the number of primes

bitset <mx> mark; //to keep track of primes
uint primes[mxprm]; //to store the primes

void sieve() { //just a prime sieve code
    mark[0] = mark[1] = 1;
    primes[psz++] = 2;
    int lim = sqrt(mx * 1.0) + 2;
    for (int i=4;i<mx;i+=2) mark[i] = 1;
    for (int i=3;i<mx;i+=2) {
        if (!mark[i]) {
            primes[psz++] = i;
            if (i<=lim)
                for (int j=i*i;j<mx;j+=i)
                    mark[j] = 1;
        }
    }
}
```

```
//Prime Factorization
//faster process
//TimeComplexity_O(sqrt(n)/ln(sqrt(n))+log2(n))
vector<long long> primes_factors(long long n){
    vector<long long> factors;
    int root = sqrt(n);
    for(int i=0;i<(int)primes.size() &&
primes[i]<=root;++i){
        if (is_prime[n]) {
            break;
        }
        if(n%primes[i]==0){
            while(n%primes[i]==0){ //log2(n)
                n /= primes[i];
                factors.push_back(primes[i]);
            }
            root=sqrt(n);
        }
    }
    if(n!=1){
        factors.push_back(n);
    }
    return factors;
}

// Calculating SPF (Smallest Prime Factor) for
every number till MAXN. O(nloglogn)
vector<int> spf(MAXN + 1, 1);
void sieve()
{
    spf[0] = 0;
    for (int i = 2; i <= MAXN; i++) {
        if (spf[i] == 1) {
            for (int j = i; j <= MAXN; j += i)
                if (spf[j]== 1) spf[j] = i;
        }
    }
}

vector<int> getFactorization(int x){
    vector<int> ret;
    while (x != 1) {
        ret.push_back(spf[x]);
        x = x / spf[x];
    }
    return ret;
}
```

```
//Returns nCr%p using Fermat's
//little theorem.
unsigned long long nCrModPFermat
(unsigned long long n,int r,int p)
{
    if(n<r) return 0;
    if(r==0)return 1;

    unsigned long long fac[n+1];
    fac[0] = 1;
    for(int i=1;i<=n;i++)
      fac[i]=(fac[i-1]*i)%p;

    return (fac[n]*modInverse(fac[r],p)%p
      *modInverse(fac[n-r],p)%p)%p;
}

int main()
{
    int n=10,r=2,p=13;
    cout << "ValueofnCr%p is"
      << nCrModPFermat(n,r,p);
}
```

```
//A modular inverse based solution to
//compute nCr % p

/* Iterative Function to calculate (x^y)%p in
O(log y) */
unsigned long long power
 (unsigned long long x,int y,int p)
{
    unsigned long long res=1;
    x = x % p;
    while(y>0){
        if(y&1) res = (res*x)%p;
        y = y>>1;//y=y/2
        x = (x * x) % p;
    }
    return res;
}

//Returnsn^(-1)modp
unsigned long long modInverse(unsigned long
long n,int p)
{
    return power(n,p-2,p);
}
```

```
const int maxn=(int)1e5+7;
int phi[maxn];
//TimeComplexity-O(nloglogn)
// EulerTotient
void phi_1_to_n() {
    for(int i=0;i<=maxn;++i){
        phi[i]=i;
    }
    for(int i=2;i<=maxn;++i){
        if (phi[i] == i) {
            for(int j=i;j<=maxn;j+=i){
                phi[j]-=phi[j]/i;
            }
        }
    }
}
//sum of coprimes until n
int sum_of_coprimes_untill_n(int n){
    return (phi[n]/2) * n;
}
```

```
void mat_mul(vector<vector<ll>> &mat1,
vector<vector<ll>> &mat2){
    vector<vector<ll>> newmat(2,
vector<ll>(2, 0));
    for(ll i=0;i<2;i++)
        for(ll j=0;j<2;j++)
            for(ll k=0;k<2;k++)
      newmat[i][j] += mat1[i][k]*mat2[k][j];

    mat1 = newmat;
}

ll fib(ll n){
    if(n==1) return 0;
    if(n==2) return 1;
    if(n==3) return 1;
    vector<vector<ll>> resmat, mat;
    resmat = mat = {{0, 1}, {1, 1}};
    ll i;
    n-=3;
    for(i=0;(1ll<<i)<=n;i++){
        if(n&(1ll<<i)) mat_mul(resmat, mat);
        mat_mul(mat, mat);
        //cout << mat[0][0] << mat[0][1] <<
mat[1][0] << mat[1][1];
    }
    return resmat[1][1];
}
```

```cpp
//EEGCD AND Linear Diophantine
ll gcd(ll a, ll b, ll& x, ll& y) {
    if (b == 0) {
        x = 1;
        y = 0;
        return a;
    }
    ll x1, y1;
    ll d = gcd(b, a % b, x1, y1);
    x = y1;
    y = x1 - y1 * (a / b);
    return d;
}

void solve(ll cs){
    ll j, i, p, q, a, b, c, m, n, k, g, mn = 0, mx = 1e10;

    cin >> n;
    while(n--){
        cin >> a >> b;
        ll x, y;
        ll g = gcd(abs(a), abs(b), x, y);
        if (a < 0) x = -x;
        if (b < 0) y = -y;
        cout << g << " " << x << " " << y;

        cout << " => ";

        // linear diophantine ax + by = c .. q = c/gcd(a,b)
        double c1;
        cin >> c1;
        double q = c1/g;
        cout << "a" << x*q << " b" << y*q
            << endl;
    }

    cout << endl;

}
```

```cpp
//Derangement:
ll mod = 1e9+7;

int main() {
    int n, m;
    cin >> n >> m;

    vector<ll> dp(n);
    dp[2] = 1, dp[1] = 0;

    for (int i = 3; i <= n; i++) {
       dp[i] = ((i - 1) * (dp[i-1] + dp[i-2]))%mod;
    }
    cout << endl;
}
```

```cpp
// Combinatorics: factorial, modular multiplicative inverse, ncr, derangement, catalan numbers

const ll mxN = 2e6+5, MOD = 1e9+7;

ll fact[mxN+5], inv_of_fact[mxN + 5], d[1003][1003];

void pre()
{
    ll i, j, ans = 1;
    // factorial with modulo
    fact[0] = 1;
    for(i=1;i<=mxN;i++)
    {
        fact[i] = fact[i-1] * i;
        fact[i] %= MOD;
    }

    // Modular Multiplicative inverse of factorial[i]
    inv_of_fact[mxN] = binpow(fact[mxN], MOD - 2);
    for(i=mxN-1;i>=0;i--)
    {
        inv_of_fact[i] = inv_of_fact[i+1] * (i+1);
        inv_of_fact[i] %= MOD;
    }

//Derangement: how j numbers can be arrange in i positions such that no one is in it's position (index != value)
//d(x, y) = (x-1)*(d(x-1, y-1) + d(x-2, y-2)) + (y-x) *(d(x-1, y-1);

//d(x, x) = (x-1) * (d(x-1, x - 1) + d(x-2, x-2))
//d(x) = (x-1) * (d(x-1) + d(x-2))

    for(i=0;i<=1000;i++)
    {
        d[0][i] = 1;
        d[1][i] = i - 1;
        d[2][i] = (i - 1) + (i - 2) * (i - 2);
        d[i][0] = 1;
    }
    for(i=3;i<=1000;i++)
    {
        for(j=1;j<=1000;j++)
        {
            d[i][j] = (i - 1)*d[i-2][j-2] + (j-1)*(d[i-1][j-1]);
            d[i][j] %= MOD;
        }
    }
}
```

```
int main() {

int catalan[imx][imx];
  for (int i = 1; i < imx; i++) {
    for (int j = 0; j <= i; j++) {
       if (j == 0) catalan[i][j] = 1;
       else
         catalan[i][j] = (catalan[i][j - 1]
+ catalan[i - 1][j]) % MOD;

    }
 }

    //OR Catalan of n = (ncr(2n, n)/(n+1));
    //OR Catalan of n =
fact[2*n]*inv[n+1]*inv[n];
    //OR Catalan of n =
((2*(2*n-1)*Cat(n-1))/(n+1))

    int n;
    while (cin >> n) {
        if (n == 0) { break; }
        cout << catalan[n][n] << '\n';
    }
}
```

```
ll ncr(ll n, ll r)
{
    if(r>n) re 0;
    if(n==r | r==0) re 1;
    re ((fact[n] *
((inv_of_fact[r]*inv_of_fact[n-r])%MOD))%MOD);
}

ll catalan(ll n)
{
    ll ans = ncr(n+n, n) * binpow(n+1, MOD - 2);
// here inverse is not inv_of_fact
    ans %= MOD;
    re ans;
}
```

```
const ll MXN = 1e6;
ll a[MXN+5], mul[MXN + 5], f[MXN + 5], g[MXN
+ 5], mp[MXN + 5];

void solve()
{
    ll n=0, q=0, i=0, j=0, k=0, m=0, x=0,
ans=0;
    cn n;
    fori
    {
        cn a[i];
        m = max(m, a[i]);
        mp[a[i]]++;
    }
    for(i=1; i<=MXN; i++)
    {
        mul[i] = mp[i];
        for(j=i+i; j<=MXN; j+=i) mul[i] +=
mp[j];
    }

 // Exclusion DP
 // f[i] = number of pairs with gcd =
multiple of i
 // g[i] = number of pairs with gcd = i

    for(i=1;i<=m;i++)
    {
        x = mul[i];
        f[i] = (x * (x-1))/2LL;
    }
```

```
 // Stirling 1st: From n different things
divide into k cycles
 // [n, k] = [n-1, k-1] + (n-1) * [n-1, k]
 //  if k = 1 then [n, 1] = fact[n-1];
 //  if k = n then [n, n] = 1;

 ll fact[mxN+5];
 fact[0] = 1;
 for(ll i=1;i<=mxN;i++)
 {
    fact[i] = fact[i-1] * i;
    fact[i] %= mod;
 }

 ll stnum[imx+5][imx+5];

  for(ll i=1;i<imx;i++){
     for(ll j=1;j<=i;j++){
       if(j==1) stnum[i][j] = fact[i-1];

       else if(j==i) stnum[i][j] = 1;

       else stnum[i][j] = (stnum[i-1][j-1] +
          ((i-1) * stnum[i-1][j])%mod)%mod;
        //cout << stnum[i][j] << " ";
     }
 }

    cout << stnum[3][2] << " " <<
stnum[4][2] << " ";
```

```
    for(i=m;i>0;i--)
    {
        g[i] = f[i];
        for(j=i+i;j<=m;j+=i)
        {
            g[i] -= g[j];
        }
    }
    cout<<g[1];
}


//MOBIUS
mobb[1] = 1;
for(int i=1;i<=m;i++){
    for(int j=i+i;j<=m;j+=i){
        mobb[j] -= mobb[i];
    }
}
```

```
 // Stirling 2nd: From n different things
divide into k
 // {n, k} = {n-1, k-1} + k * {n-1, k}
 // if k = 1 then {n, 1} = 1;
 // if k = n then {n, n} = 1;

 ll stnum[imx+5][imx+5];

 for(ll i=1;i<imx;i++){
  for(ll j=1;j<=i;j++){
   if(j==1 || j==i) stnum[i][j] = 1;
   else stnum[i][j] = (stnum[i-1][j-1] +
         (j * stnum[i-1][j])%mod)%mod;
         //cout << stnum[i][j] << " ";
  }
 }

   cout << stnum[3][2] << " " << stnum[4][2]
<< " ";
```

```
void dijkstra(ll i){
    priority_queue<tup, vector<tup>,
greater<tup>> pq;
    dis[i] = 0;
    pq.push({dis[i], i});
    while(!pq.empty()){
        auto [d, u] = pq.top();
        pq.pop();
        //cout << d << " " << u << endl;
        if(d != dis[u]) continue;
        for(auto x : g[u]){
            if(dis[u] + x.second < dis[x.first]){
                dis[x.first] = dis[u] + x.second;
                pq.push({dis[x.first], x.first});
            }
        }
    }
}
```

```
void mat_mul(vector<vector<ll>> &m1,
vector<vector<ll>> &m2, ll mod){

 ll n = m1.size(), i, j, k;
 vector<vector<ll>> mat(n, vector<ll> (n));

 for(i=0;i<n;i++){
  for(j=0;j<n;j++){
   ll tmp = 0;
   for(k=0;k<n;k++){
    tmp += (m1[i][k]*m2[k][j])%mod;
    tmp %= mod;
   }
   mat[i][j] = tmp;
  }
 }
 m1 = mat;
}

vector<vector<ll>>
binmatpow(vector<vector<ll>> a, ll b,ll mod){
 ll i, n = a.size();
 vector<vector<ll>> res(n, vector<ll> (n,
0));

    for(i=0;i<n;i++) res[i][i] = 1;
    for(i=0;(1LL<<i)<=b;i++){
        if(b&(1LL<<i)) mat_mul(res, a, mod);
        mat_mul(a, a, mod);
    }
    return res;
}
```

```
// Bellman-Ford Algorithm
// Better than Dijkstra if there are negative weights, edges and cycles
// takes three elements in the edges
vector<vector<int>> edge;

void bellman_ford(int node) {
  // firstly, every distance is infinity
  vector<int> dist(n, inf);
  // source nodes distance to itself is 0
  dist[node] = 0;
  // checking for the shortest path distance
  for (int i = 0; i < n - 1; ++i) {
    for (auto& x: edge) {
      int u = x[0], v = x[1], w = x[2];
      dist[v] = min(dist[v], w + dist[u]);
    }
  }
  int ok = 0;
  // checking for negative cycles
  for (auto& x: edge) {
    int u = x[0], v = x[1], w = x[2];
    if (dist[u] != inf && dist[u] + w < dist[v])
    {
      ok = 1;
      break;
    }
  }
  if (ok) {
    cout << "Negative Cycle Found\n";
  }
  else {
    for (int i = 1; i <= n; ++i) {
      cout << dist[i] << ' ';
    }
    cout << '\n';
  }
  // if there is a negative cycle, then the shortest path cannot be found
  // else print the answer
}
```

```
vector<pair<ll, ll>> v[1000006];
// Prim MST
double primmst(ll i, ll n){
    priority_queue<pair<ll, ll>, vector<pair<ll, ll>>, greater<pair<ll, ll>>> pq;

    vector<ll> key(n+1, 1e9);
    vector<ll> par(n+1, -1);
    vector<bool> inmst(n+1, false);

    ll src = 1, tot = 0;
    key[src] = 0;
    pq.push({key[src], src});

    while(!pq.empty()){
        pair<ll, ll> pi = pq.top();
        pq.pop();

        if(inmst[pi.se]) continue;
        inmst[pi.se] = true;
        tot += pi.fi;

        for(auto x : v[pi.se]){
            if(!inmst[x.fi] and key[x.fi]>x.se){
                key[x.fi] = x.se;
                pq.push({key[x.fi], x.fi});
                par[x.fi] = pi.se;
            }
        }
    }


    for(i=2;i<n+1;i++)
        cout << par[i] << " " << i << endl;


    return tot;

}
```

```cpp
// Kruskal (Minimum Spanning Tree)
// Time complexity O(E log E)

struct DSU {
  vector<int> par, rnk, size; int c;
  DSU(int n) : par(n + 1), rnk(n + 1,0), size(n +
1,1), c(n) {
    for (int i = 1; i <= n; ++i) par[i] = i;
  }
  int find(int i) { return (par[i] == i ? i :
(par[i] = find(par[i]))); }
  bool same(int i, int j) { return find(i) ==
find(j); }
  int get_size(int i) { return size[find(i)]; }
  int count() { return c; } //connected
components
  // Path compression
  // O(1)
  int merge(int i, int j) {
    if ((i = find(i)) == (j = find(j))) return
-1; else --c;
    if (rnk[i] > rnk[j]) swap(i, j);
    par[i] = j; size[j] += size[i];
    if (rnk[i] == rnk[j]) rnk[j]++;
    return j;
  }
};

int main() {
  ios_base::sync_with_stdio(0), cin.tie(0);

  int n, m;
  cin >> n >> m;
  vector<array<int, 3>> edges;
  for (int i = 1; i <= m; ++i) {
    int u, v, w;
    cin >> u >> v >> w;
    edges.push_back({w, u, v});
  }
  sort(edges.begin(), edges.end());
  long long ans = 0, cnt_edges = 0;
  DSU dsu(n);
  for (auto& x: edges) {
    int u = x[1], v = x[2], w = x[0];
    if (dsu.same(u, v)) {
      continue;
    }
    ans += w;
    dsu.merge(u, v);
    ++cnt_edges;
  }
  if (ans >= 0 && cnt_edges == n - 1) {
    cout << ans << '\n';
  }
  else {
    cout << "IMPOSSIBLE\n";
  }
  return 0;
}
```

```cpp
//LCA using sparse table
//Complexity: O(NlgN,lgN)

const int MAX = 2e5;
const int LOG = 19;
int up[LOG][MAX], depth[MAX];
vector<int> adj[MAX];
void dfs(int node, int prev, int dist) {
    depth[node] = dist;
    if (prev != -1) { up[0][node] = prev; }
    for (int i = 1; i < LOG; i++) {
up[i][node] = up[i - 1][up[i - 1][node]]; }
    for (int nxt : adj[node]) {
        if (nxt != prev) { dfs(nxt, node,
dist + 1); }
    }
}
int lca(int a, int b) {
    if (depth[a] < depth[b]) { swap(a, b); }
    int dist = depth[a] - depth[b];
    for (int i = LOG - 1; i >= 0; i--) {
        if ((dist >> i) & 1) { a = up[i][a];
}
    }
    if (a == b) { return a; }
    for (int i = LOG - 1; i >= 0; i--) {
        if (up[i][a] != up[i][b]) {
            a = up[i][a];
            b = up[i][b];
        }
    }
    return up[0][a];
}



int main() {
    int n, q;
    cin >> n >> q;
    for (int i = 1; i < n; i++) {
        int b;
        cin >> b;
        adj[--b].push_back(i);
    }
    dfs(0, -1, 0);
    for (int i = 0; i < q; i++) {
        int x, y;
        cin >> x >> y;
        cout << lca(--x, --y) + 1 << "\n";
    }
}
```

```
//Euler Tour SEGTREE (LCA)
vector<ll> g[200006];
vector<ll> et, tin, tout;
ll t = 0;

void dfs(ll i, ll p){
    et.pb(i);
    tin[i] = t++;
    for(auto x : g[i]){
        if(x!=p){
            dfs(x, i);
            et.pb(i);
            tout[i] = t++;
        }
    }
}

struct segtree{
    segtree(ll sz){
        n = sz;
        tree.resize(4*sz, 0);
    }
    ll n;
    vector<ll> tree;

    ll combine(ll l, ll r){
        if(l==-1) return r;
        if(r==-1) return l;
        if(tin[l]<tin[r]) return l;
        else return r;
    }

    void build(ll i, ll l, ll r, vector<ll>
&v){
        if(l==r){
            tree[i] = v[l];
            return;
        }
        ll mid = (l+r)/2;
        build(i*2, l, mid, v);
        build(i*2+1, mid+1, r, v);
        tree[i] = combine(tree[i*2],
tree[i*2+1]);
    }

    ll query(ll i, ll l, ll r, ll b, ll e){
        if(e<l or r<b) return -1;
        if(b<=l and r<=e){
            return tree[i];
        }
        ll mid = (l+r)/2;
        return combine(query(i*2, l, mid, b,
e), query(i*2+1, mid+1, r, b, e));
    }
};

void solve(ll cs){
    ll a=0, x, y, k, b=0, j, i, c, q, n, m,
d, mn = 0, mx;

    cin >> n >> k;
```

```
//Euler Tour SPT (LCA)
vector<ll> g[200006];
vector<ll> et, tin, tout;
ll t = 0;

void dfs(ll i, ll p){
    et.pb(i);
    tin[i] = t++;
    for(auto x : g[i]){
        if(x!=p){
            dfs(x, i);
            et.pb(i);
            tout[i] = t++;
        }
    }
}

struct sptable{
    sptable(ll sz){
        n = sz;
        log = log2(n)+1;
        st = vector<vector<ll>>(n, vector<ll>
(log, 0));
    }
    ll n, log;
    vector<vector<ll>> st;

    ll combine(ll l, ll r){
        if(l==-1) return r;
        if(r==-1) return l;
        if(tin[l]<tin[r]) return l;
        else return r;
    }

    void build(vector<ll> &v){
        for(ll i=0;i<n;i++)
            st[i][0] = v[i];

        for(ll i=1;i<log;i++){
            for(ll j=0;j<n-(1<<i)+1;j++){
                st[j][i] =
combine(st[j][i-1], st[j+(1<<(i-1))][i-1]);
            }
        }
    }

    ll query(ll a, ll b){
        ll i = b-a+1;
        i = log2(i);
        return combine(st[a][i],
st[b-(1<<i)+1][i]);
    }
};



void solve(ll cs){
    ll a=0, x, y, k, b=0, j, i, c, q, n, m,
d, mn = 0, mx;

    cin >> n >> k;
```

```
    for(i=0;i<n-1;i++) {
        cin >> b;
        g[i+2].pb(b);
        g[b].pb(i+2);
    }

    t = 0;
    tin.resize(n+1, 0);
    tout.resize(n+1, 0);
    dfs(1, -1);

    segtree st = segtree(et.size());
    st.build(1, 0, et.size()-1, et);

    while(k--){
        cin >> a >> b;
        if(tin[a]>tin[b]) swap(a, b);
        mn = st.query(1, 0, et.size()-1,
tin[a], tin[b]);
        cout << mn << endl;
    }

}
```

```
    for(i=0;i<n-1;i++) {
        cin >> b;
        g[i+2].pb(b);
        g[b].pb(i+2);
    }

    t = 0;
    tin.resize(n+1, 0);
    tout.resize(n+1, 0);
    dfs(1, -1);

    sptable st = sptable(et.size());
    st.build(et);

    while(k--){
        cin >> a >> b;
        if(tin[a]>tin[b]) swap(a, b);
        mn = st.query(tin[a], tin[b]);
        cout << mn << endl;
    }

}
```

```cpp
// SCC
const int maxn = 2e5 + 5;

int n, m, visited[maxn], ind[maxn];
vector<int> graph[maxn], graph_trans[maxn], s;

void dfs(int node, int pass, int num) {
  visited[node] = 1;
  vector<int> g = (pass == 1 ? graph[node] :
graph_trans[node]);
  for (auto& edge: g) {
    if (!visited[edge]) dfs(edge, pass, num);
  }
  s.push_back(node);
  if (pass == 2) ind[node] = num;
}

int32_t main() {
  ios_base::sync_with_stdio(0), cin.tie(0);

  cin >> n >> m;
  for (int i = 0; i < m; ++i) {
    int u, v;
    cin >> u >> v;
    graph[u].push_back(v);
    graph_trans[v].push_back(u);
  }
  for (int i = 1; i <= n; ++i) {
    if (!visited[i]) dfs(i, 1, 0);
  }
  memset(visited, 0, sizeof(visited));
  int components = 0;
  for (int i = n - 1; i >= 0; --i) {
    if (!visited[s[i]]) {
      ++components;
      dfs(s[i], 2, components);
    }
  }
  cout << components << '\n';
  for (int i = 1; i <= n; ++i) cout << ind[i] <<
" \n"[i == n];
  return 0;
}
```

```cpp
vector<ll> v[1000000];
// toposort
void toposort(ll i, vector<bool> &vis,
stack<ll> &st){
    vis[i] = true;

    for(auto x : v[i]){
        if(!vis[x]) toposort(x, vis, st);
    }

    st.push(i);
}
```

```cpp
//Floyd Warshal
for (int k = 0; k < n; ++k) {
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (d[i][k] < INF && d[k][j] < INF)
                d[i][j] = min(d[i][j],
                  d[i][k] + d[k][j]);
        }
    }
}
```

```cpp
// Sparse Table
// Function to build the sparse table
vector<vector<ll>> buildSparseTable(
             vector<ll>& arr) {
    ll n = arr.size();
    ll k = log2(n) + 1;
    vector<vector<ll>>
      sparseTable(n, vector<ll>(k, -1e9));

    // Initialize sparse table for range with
length 1
    for (ll i = 0; i < n; i++) {
        sparseTable[i][0] = arr[i];
    }

// Build sparse table
 for (ll j = 1; (1LL << j) <= n; j++) {
   for (ll i = 0; (i + (1LL << j) - 1) < n; i++)
   {
     sparseTable[i][j] = max(sparseTable[i][j-1],
sparseTable[i + (1LL << (j - 1))][j - 1]);
   }
}
```

```cpp
//Articulation Point
vector<ll> parent, v[200006], dis, low;
vector<bool> vis, arti_point;
ll t = 0, ans = 0, root = 1;

void dfs(ll i){
    vis[i] = true;
    low[i] = dis[i] = t++;
    for(auto x : v[i]){
        if(vis[x]==false){
          parent[x] = i;
          dfs(x);
          if(root==i){
            if(!arti_point[i] and
             dis[i]<low[x] and v[i].size()>1){
                arti_point[i] = true;
            }
          }
          else if(dis[i]<=low[x]){
              arti_point[i] = true;
          }
          else low[i] = min(low[i], low[x]);
        }
```

```
    return sparseTable;
}


// Function to query the maximum value in a range
ll queryMax(vector<vector<ll>>& sparseTable, ll
l, ll r) {
    ll k = log2(r - l + 1);
    return max(sparseTable[l][k],
           sparseTable[r - (1LL << k) + 1][k]);
}
```

```
        else if(parent[i]!=x){
            low[i] = min(low[i], dis[x]);
        }
    }
}

void solve(ll cs){
    ll n, m, a=0, x, y, k, b=0, j, i, c, q, mn =
1e12, mx, mod = 998244353;

    cin >> n >> m;

    for(i=0;i<=n;i++) v[i].clear();
    vis = vector<bool> (n+1, 0);
    parent = vector<ll> (n+1, 0);
    dis = vector<ll> (n+1, 0);
    low = vector<ll> (n+1, 0);
    arti_point = vector<bool> (n+1, false);

    while(m--){
        cin >> a >> b;
        v[a].pb(b);
        v[b].pb(a);
    }

    ans = 0;

    dfs(1);

    for(i=1;i<=n;i++) {
        cout << arti_point[i] << " ";
        if(arti_point[i]) ans++;
    }
    cout << ans << endl;
```

```
//Articulation Bridge
vector<pair<int, int>> bridges;
int tin[MAXN], low[MAXN], timer;
bool visited[MAXN];

void dfs(int v, int p = -1) {
    visited[v] = true;
    tin[v] = low[v] = timer++;
    for (int to : adj[v]) {
        if (to == p) continue;
        if (visited[to]) {
            low[v] = min(low[v], tin[to]);
        } else {
            dfs(to, v);
            low[v] = min(low[v], low[to]);
            if (low[to] > tin[v]) {
                bridges.push_back({v, to});
            }
        }
    }
}
```

```
void findBridges(int n) {
    timer = 0, bridges.clear();
    memset(visited, false, sizeof(visited));
    memset(tin, -1, sizeof(tin));
    memset(low, -1, sizeof(low));
    for (int i = 0; i < n; i++) {
        if (!visited[i]) {
            dfs(i);
        }
    }
}
int main() {
    findBridges(n);
    cout << "Articulation Bridges:\n";
    for (auto bridge : bridges) {
        cout << bridge.first << " - " <<
bridge.second << "\n";}
}
```

```
vector<ll> par;
// DSU
ll fd(ll r){
    if(r==par[r]) return r;
    par[r] = fd(par[r]);
    return par[r];
}

void uni(ll a, ll b){
    ll u = fd(a);
    ll v = fd(b);
    if(u==v){
        cout << "They are already friends" <<
endl;
    }
    else{
        par[u] = v;
    }
}
```

```
// BIT
int BIT[1000], a[1000], n;

void update(int x, int val)
{
      for(; x <= n; x += x&-x)
        BIT[x] += val;
}

int query(int x)
{
     int sum = 0;
     for(; x > 0; x -= x&-x)
        sum += BIT[x];
     return sum;
}

int main()
{
     scanf("%d", &n);
     int i;
     for(i = 1; i <= n; i++)
     {
           scanf("%d", &a[i]);
           update(i, a[i]);
     }
     printf("sum of first 10 elements is %d\n",
query(10));
     printf("sum of all elements in range [2, 7]
is %d\n", query(7) - query(2-1));
     return 0;
}
```

```
/* Prefix Trie */

struct node{
    bool endmark;
    node *next[26+1];
    node(){
        for(ll i=0;i<26;i++) next[i] = NULL;
        endmark = false;
    }
};

node *root;

void insert(string s){
    ll n = s.size();
    node *curr = root;
    for(ll i=0;i<n;i++){
        if(curr->next[s[i]-'a']==NULL)
            curr->next[s[i]-'a'] = new node();
        curr = curr->next[s[i]-'a'];
    }
    curr->endmark = 1;
}

bool search(string s){
    ll n = s.size();
    node *curr = root;
    for(ll i=0;i<n;i++){
        if(curr->next[s[i]-'a']==NULL)
                return false;
        curr = curr->next[s[i]-'a'];
    }
    return curr->endmark;
}

void del(node* curr){
    for(ll i=0;i<26;i++){
        if(curr->next[i]!=NULL)
            del(curr->next[i]);
    }
    delete (curr);
}


int main(){
    ios_base::sync_with_stdio(false);
cin.tie(NULL); cout.tie(NULL);
    //seive(1e6+2);
    root = new node();
    ll i, n;
    cin >> n;
    for(i=0;i<n;i++){
        string s;
        cin >> s;
        insert(s);
    }
    ll q;
    cin >> q;
    while(q--){
        string s;
        cin >> s;
        if(search(s)) cout << "Found" << endl;
        else cout << "Not Found" << endl;
    }
    del(root);
}
```

```
struct segtree{
    segtree(ll sz){
        n = sz;
        tree.resize(4*sz, 0);
        lazy.resize(4*sz, 0);
    }
    ll n;
    vector<ll> tree;
    vector<ll> lazy;
    void push(ll i, ll l, ll r){
        if(lazy[i]==0) return;
        if(l!=r){
            lazy[i*2] += lazy[i];
            lazy[i*2+1] += lazy[i];
        }
        tree[i] += lazy[i]*(r-l+1);
        lazy[i] = 0;
    }

    ll combine(ll l, ll r){return l+r;}

    void build(ll i, ll l, ll r, vector<ll> &v){
        if(l==r) {
            tree[i] = v[l];
            return;
        }
        ll mid = (l+r)/2;
        build(i*2, l, mid, v);
        build(i*2+1, mid+1, r, v);
        tree[i] = combine(tree[i*2],
tree[i*2+1]);
    }
    void Build(vector<ll> &v)
         {build(1, 0, n-1, v);}

    void update(ll i, ll l, ll r, ll b, ll e, ll
val){
        push(i, l, r);
        if(b<=l and r<=e) {
            lazy[i] = val;
            push(i, l, r);
            return;
        }
        if(e<l or r<b) return;
        ll mid = (l+r)/2;
        update(i*2, l, mid, b, e, val);
        update(i*2+1, mid+1, r, b, e, val);
        tree[i] = combine(tree[i*2],
tree[i*2+1]);
    }
    void Update(ll i, ll j, ll val){
        update(1, 0, n-1, i, j, val);}

    ll query(ll i, ll l, ll r, ll b, ll e){
        push(i, l, r);
        if(b<=l and r<=e) {
            return tree[i];
        }
        if(e<l or r<b) return 0;
        ll mid = (l+r)/2;
        return combine(query(i*2, l, mid, b, e),
query(i*2+1, mid+1, r, b, e));
    }
    ll Query(ll i, ll j){
        return query(1, 0, n-1, i, j);}
};
```

```
struct node{
    ll l, r, mid, tree, op;
    node *left, *right;
    node(ll s, ll e){
        l = s, r = e, mid = (l+r)/2;
        tree = op = 0;
        left = right = NULL;
    }

    void push(ll val){
        op = val;
        tree += val;
    }

    void prop(){
        if(left==NULL) left = new node(l,
mid);
        if(right==NULL) right = new
node(mid+1, r);
        if(op==0) return;
        left->push(op);
        right->push(op);
        op = 0;
    }

    ll combine(ll a, ll b){
        return (a+b);
    }

    void update(ll b, ll e, ll val){
        prop();
        if(b<=l and r<=e){
            push(val);
            return;
        }
        if(e<l or r<b) return;
        left->update(b, e, val);
        right->update(b, e, val);
        tree = combine(left->tree,
right->tree);
    }

    ll query(ll b, ll e){
        prop();
        if(b<=l and r<=e){
            return tree;
        }
        if(e<l or r<b) return 0;
        return combine(left->query(b, e),
                right->query(b, e));
    }
};
```

```
void solve(ll cs)
{
    ll n, m, i, ans=0, mn = 1e15, mx, cnt, q;

    cin >> n >> k;
    vector<ll> v(n, 0);

    segtree st = segtree(n);
    st.Build(v);

    while(k--){s
        cin >> i;
        if(i==0){
            cin >> a >> b >> q;
            st.Update(a, b, q);
        }
        else{
            cin >> a >> b;
            cout << st.Query(a, b) << endl;
        }
    }

}
```

```
//LCS
int main(){
    ll i, j, n, m;

    string s1, s2;
    cin >> s1 >> s2;

    n = s1.size();
    m = s2.size();

    vector<vector<ll>> mem(n+1, vector<ll> (m+1,
0));

    for(i=n-1;i>-1;i--){
        for(j=m-1;j>-1;j--){
            if(s1[i]==s2[j]){
                mem[i][j] = 1 + mem[i+1][j+1];
            }
            else{
                mem[i][j] = max(mem[i+1][j],
mem[i][j+1]);
            }
        }
    }

    cout << mem[0][0] << endl;

}
```

```
//Knapsack
vector<ll> pv, wv;
ll mem[1000][1000];

ll dpop(ll i, ll n, ll w){
    if(i==n) return 0;
    if(w==0) return 0;
    if(mem[i][w]!=0) return mem[i][w];

    ll r1=0, r2=0;
    if(w-wv[i]>=0) r1 = pv[i] + dpop(i+1, n,
w-wv[i]);
    r2 = dpop(i+1, n, w);

    return mem[i][w] = max(r1, r2);
}

int main(){

    ll i, num_of_int, j, k, weight, ans;

    cin >> num_of_int >> weight;
    pv = vector<ll>(num_of_int);
    wv = vector<ll>(num_of_int);
    for(auto &x : pv) cin >> x;
    for(auto &x : wv) cin >> x;

    ans = dpop(0, num_of_int, weight);
    cout << ans << endl;

}
```

```
vector<ll> v;
ll mem[1000][1000];
//COIN CHANGE
ll dp(ll i, ll n, ll k){
    if(i==n and k!=0) return 1e9;
    if(k==0) return 0;
    if(mem[i][k]!=0) return mem[i][k];
```

```
vector<ll> v;
ll mem[1000][1000];
//COIN CHANGE
ll dpop(ll i, ll n, ll w, ll k){
    if(w<0) return 1e9;
    if(i==n and w!=0) return 1e9;
    if(w==0) return 0;
    if(mem[i][w]!=0) return mem[i][w];
```

```
    ll r1 = 1e9, r2 = 1e9;
    if(k-v[i]>=0) r1 = 1+dp(i+1, n, k-v[i]);
    r2 = dp(i+1, n, k);

    return mem[i][k] = min(r1, r2);

}
```

```
    ll ans = 1e9;
    for(ll j=0;j<=k;j++){
        ans = min(ans,
           j+dpop(i+1, n, w-j*v[i], k));
    }

    return mem[i][w] = ans;
}
```

```
vector<ll> v;
ll mem[10000];
//COIN CHANGE
ll dpop(ll n, ll k){
    if(k<0) return 1e9;
    if(k==0) return 0;
    if(mem[k]!=0) return mem[k];

    ll ans = 1e9;
    for(ll i=0;i<n;i++){
        ans = min(ans, 1+dpop(n, k-v[i]));
    }

    return mem[k] = ans;

}
```

```
vector<vector<ll>> mem(1000,
              vector<ll> (1000, -1));
//STRING COMMON LCS Type
int dp(int i, int j, string s1, string s2){
    if(i==n) return m-j;
    if(j==m) return n-i;

    if(mem[i][j]!=-1) return mem[i][j];

    int ans = 0;
    if(s1[i]==s2[j])
       ans = dp(i+1, j+1, s1, s2);
    else{
        ans = 1 + min(dp(i+1, j, s1, s2),
              min(dp(i, j+1, s1, s2),
               dp(i+1, j+1, s1, s2)));
    }
    return mem[i][j] = ans;
}
```

```
#define EMPTY_VALUE -1
#define MAX_N 10
#define INF 1061109567
// BITMASK_Subset_DP
//Travelling_Salesman
int w[MAX_N][MAX_N];
int mem[MAX_N][1<<MAX_N];

int turnOn(int x, int pos) {
    return N | (1<<pos);
}

bool isOn(int x ,int pos) {
    return (bool)(x & (1<<pos));
}
int n;
int f(int i, int mask) {
    if (mask == (1<<n) - 1) {
        return w[i][0];
    }
```

```
    if (mem[i][mask] != -1) {
        return mem[i][mask];
    }

    int ans = INF;
    for (int j = 0;j < n;j++) {
        if (w[i][j] == INF) continue;

        if (isOn(mask,j) == 0) {
            int result = f(j, turnOn(mask,
j)) + w[i][j];
            ans = min(ans, result);
        }
    }

    return mem[i][mask] = ans;
}
```

```cpp
// Knapsack (Tabulation)
// Time Complexity O(n * w)
// Space Complexity O(n * w)
const int maxn = 102;
const int maxm = 1e5 + 5;
int n, w;
int dp[maxn][maxm];
int wt[maxn], val[maxn];

int32_t main() {
  ios_base::sync_with_stdio(0), cin.tie(0);

  cin >> n >> w;
  for (int i = 1; i <= n; ++i) {
    cin >> wt[i] >> val[i];
  }
  for (int i = 1; i <= n; ++i) {
    for (int cap = 0; cap <= w; ++cap) {
      if (cap < wt[i]) {
        dp[i][cap] = dp[i - 1][cap];
      }
      else {
        dp[i][cap] = max(val[i] + dp[i - 1][cap -
wt[i]], dp[i - 1][cap]);
      }
    }
  }
  cout << dp[n][w] << '\n';
  return 0;
}
```

```cpp
// Subset Sum (Tabulation)
// Time Complexity - O(n * target)
const int maxn = 1e2 + 5;
const int maxm = 1e5 + 5;
int nums[maxn];
int dp[maxn][maxm];
int32_t main() {
  ios_base::sync_with_stdio(0), cin.tie(0);

  int n, m;
  cin >> n >> m;
  for (int i = 1; i <= n; ++i) cin >> nums[i];
  // 1. base case
  dp[0][0] = 1;
  for (int i = 1; i <= m; ++i) dp[0][i] = 0;
  for (int i = 1; i <= n; ++i) dp[i][0] = 1;
  for (int i = 1; i <= n; ++i) {
    for (int target = 1; target <= m; ++target) {
      int ans1 = dp[i - 1][target];
      if (target < nums[i]) {
        dp[i][target] = ans1;
      }
      else {
        int ans2 = dp[i - 1][target - nums[i]];
        dp[i][target] = ans1 || ans2;
      }
    }
  }
  cout << dp[n][m] << '\n';
  return 0;
}
```

```cpp
int lisNlogN(vector<int> const& a) {
    int n = a.size();
    const int INF = 1e9;
    vector<int> d(n+1, INF);
    d[0] = -INF;

    for (int i = 0; i < n; i++) {
        int l = upper_bound(d.begin(),
               d.end(), a[i]) - d.begin();
        if (d[l-1] < a[i] && a[i] < d[l])
            d[l] = a[i];
    }

    int ans = 0;
    for (int l = 0; l <= n; l++) {
        if (d[l] < INF)
            ans = l;
    }
    return ans;
}
```

```cpp
int LISNlogN(){
    ll i, n, j, k, ans;
    cin >> n;

    vector<ll> v(n);
    for(auto &x : v) cin >> x;
    vector<ll> mem;

    for(auto x : v){
        ll ind = lower_bound(mem.begin(),
mem.end(), x) - mem.begin();

        if(ind==mem.size()) mem.push_back(x);
        else mem[ind] = x;
        for(auto xx : mem) cout << xx << " ";
        cout << endl;
    }

    cout << mem.size();

}
```

```cpp
// DIGIT DP
pair<string, ll> dp[19][2][2][2];

pair<string, ll> DP(ll i, ll l, ll h, ll st,
string &s1, string &s2){
    if(i==s1.size()) return {"", 1};
    if(dp[i][l][h][st].se!=-1)
        return dp[i][l][h][st];

    ll s = s1[i]-'0', e = s2[i]-'0';
    if(l) s = 0;
    if(h) e = 9;

    pair<string, ll> ans = {"", -1};
    for(ll j=s;j<=e;j++){
        ll is_l = l;
        if(j>s1[i]-'0') is_l = 1;

        ll is_h = h;
        if(j<s2[i]-'0') is_h = 1;

        ll is_st = st;
        if(j!=0) is_st = 1;

        pair<string, ll> pi = DP(i+1, is_l, is_h,
is_st, s1, s2);
        char ch = '0'+j;
        pi.fi = ch + pi.fi;
        if(is_st) pi.se = pi.se * j;

        //cout << pi.fi << " " << pi.se << endl;
        if(ans.se<pi.se) ans = pi;
        //ans %= mod;
    }

    return dp[i][l][h][st] = ans;
}

void solve(ll cs)
{
    ll n, m, i, a, b, c, d, j, k, ans=0, mn =
1e15, mx, cnt, q;

    string s, s1, s2;

    cin >> a >> b;

    //vector<ll> v(n);

    s1 = to_string(a);
    s2 = to_string(b);

    s = "";
    for(i=s1.size();i<19;i++) s += '0';
    s1 = s + s1;

    s = "";
    for(i=s2.size();i<19;i++) s += '0';
    s2 = s + s2;


    for(auto &x : dp)
     for(auto &xx : x)
      for(auto &xxx : xx)
       for(auto &xxxx : xxx)
         xxxx = {"", -1};
```

```cpp
// DIGIT DP
ll dp[51][17][17][17][2];

ll DP(ll i, ll c1, ll c2, ll c3, ll u, string
&s){
    //cout << sz << " ";
    ll mx = max(c1, max(c2, c3));
    mx = 3*mx - (c1+c2+c3);
    if(51-i<mx) return 0;
    if(i==s.size()) {
        if(c1==c2 and c2==c3 and c1>=1)
return 1;
        else return 0;
    }
    //Optimization technique
    if(dp[i][c1][c2][c3][u]!=-1 and u)
return dp[i][c1][c2][c3][u];

    ll ans = 0;

    for(ll j=0;j<=9;j++){
        ll num = (s[i]-'0');
        if(!u and j>num) break;

        ll is_u = u;
        if(j<num) is_u = 1;
/*
        ll is_st = st;
        if(j!=0) is_st = 1;
*/
        ll nc1 = c1 + ((j==3) ? 1 : 0);
        ll nc2 = c2 + ((j==6) ? 1 : 0);
        ll nc3 = c3 + ((j==9) ? 1 : 0);
        ans += DP(i+1, nc1, nc2, nc3, is_u,
s);
        ans %= mod;
        //cout << ans << " " << i << " " <<
pos << " " << j << endl;
    }

    return dp[i][c1][c2][c3][u] = ans;
}

void solve(ll cs)
{
    ll n, m, i, a, b, c, d, j, k, ans=0, mn
= 1e15, mx, cnt, q;
    string s, s1, s2;
    cin >> s1 >> s2;

    mn = 51 - s1.size(), s = "";
    while(mn--) s += '0';
    s1 = s + s1;

    mn = 51 - s2.size(), s = "";
    while(mn--) s += '0';
    s2 = s + s2;

    ans = DP(0, 0, 0, 0, 0, s2);
    ans -= DP(0, 0, 0, 0, 0, s1);
```

```
    pair<string, ll> pi = DP(0, 0, 0, 0, s1, s2);

    bool ok = 1;
    for(i=0;i<pi.fi.size();i++){
        if(pi.fi[i]=='0' and ok) continue;
        else ok = 0;
        cout << pi.fi[i];
    }

    cout << endl;

}
```

```
    ll c1 = 0, c2 = 0, c3 = 0;
    for(auto x : s1) {
        c1 += ((x=='3') ? 1 : 0);
        c2 += ((x=='6') ? 1 : 0);
        c3 += ((x=='9') ? 1 : 0);
    }

    if(c1==c2 and c2==c3 and c1>0) ans++;
    if(ans<0) ans += mod;

    cout << ans;
    cout << endl;

}
```

```
// KMP
// Time Complexity - O(m + n)
vector<int> prefix_function(string s) {
  int n = (int)s.size();
  vector<int> pi(n, 0);
  for (int i = 1; i < n; ++i) {
    int j = pi[i - 1];
    while (j > 0 && s[i] != s[j]) {
      j = pi[j - 1];
    }
    if (s[i] == s[j]) {
      ++j;
    }
    pi[i] = j;
  }
  return pi;
}

int main(int argc, char const *argv[]) {
  string s = "na";
  vector<int> prefix = prefix_function(s);
  string t = "apnacollege";
  int pos = -1;
  int i = 0, j = 0;
  while (i < (int)t.size()) {
    if (t[i] == s[j]) {
      ++j;
      ++i;
    }
    else {
      if (j != 0) {
        j = prefix[j - 1];
      }
      else {
        ++i;
      }
    }
    if (j == (int)s.size()) {
      pos = i - (int)s.size();
      break;
    }
  }
  cout << pos << '\n';
}
```

```
vector<ll> z_function(string s) {
    ll n = s.size();
    vector<ll> z(n);
    ll l = 0, r = 0;
    for(ll i = 1; i < n; i++) {
        if(i < r) {
            z[i] = min(r - i, z[i - l]);
        }
        while(i + z[i] < n && s[z[i]] == s[i
+ z[i]]) {
            z[i]++;
        }
        if(i + z[i] > r) {
            l = i;
            r = i + z[i];
        }
    }
    return z;
}
```

```
// HASHING
const ll MOD = 1e9+7;

struct hash_st{
    string s;
    ll mod, base;
    vector<ll> h, hrev, pow;
    hash_st(string s, ll mod, ll base){
        this->mod=mod;
        this->s=s;
        this->base=base;
    }
    void init(){
        h.push_back(0);
        pow.push_back(1);
        for(ll i=0;i<s.size();i++){
            pow.push_back((pow.back()
                * base) % mod);
            ll tmp = ((h.back()*base)%mod
                  +(s[i]-'0'+1))%mod;
            h.push_back(tmp);
        }
    }
    void init_reverse(){
        string srev=s;
        reverse(srev.begin(),srev.end());
        hrev.push_back(0);
        for(ll i=0;i<srev.size();i++){
            ll tmp = (hrev.back()*base
                   +(srev[i]-'0'+1))%mod;
            hrev.push_back(tmp);
        }
    }
    ll get_hash(ll l, ll r){
        if(l<=r){
            return (h[r]-(h[l-1]
                *pow[r-l+1])%mod+mod)%mod;
        }
        else {
            l=s.size()-l+1;
            r=s.size()-r+1;
            return (hrev[r]-(hrev[l-1]
                *pow[r-l+1])%mod+mod)%mod;
        }
    }
};

void solve(ll cs)
{
    ll n, q, a, b;
    string s;
    cin >> s ;
    n = s.size();

    // 1 based indexing
    hash_st h1 = hash_st(s, MOD, 71);
    h1.init();
    h1.init_reverse();
```

```
// AhoCorasick
const int MAX_N = 6e5+5, SIGMA = 26;
int nodes = 1;
int trie[MAX_N][SIGMA], fail[MAX_N], nxt[MAX_N],
seen[MAX_N], ans[MAX_N], isEnd[MAX_N];
vector<int> leaf[MAX_N], occ[MAX_N], g[MAX_N];
// fail[u]=the failure link for node
// seen[u]=check if a node has been visited
// ans[i]=the number of occurrences of word i
// leaf[node] stores the indices of the words
ending in node

void add_word(const string &word, const int &idx)
{
    int node = 1;
    for (char ch : word) {
        if (trie[node][ch - 'a'] == 0)
            trie[node][ch - 'a'] = ++nodes;
        node = trie[node][ch - 'a'];
    }
    isEnd[node] = idx;
    leaf[node].push_back(idx);
}

void build() {
    queue<int> q;
    int node = 1;
    fail[node] = 1;
    for (int i = 0; i < SIGMA; i++) {
        if (trie[node][i]) {
            fail[trie[node][i]] = node;
            q.push(trie[node][i]);
        }
        else trie[node][i] = 1;
    }

    while (!q.empty()) {
        int node = q.front(); q.pop();
        for (int i = 0; i < SIGMA; i++) {
            if (trie[node][i]) {
                fail[trie[node][i]] =
                        trie[fail[node]][i];
                q.push(trie[node][i]);
            }
            else trie[node][i] =
                        trie[fail[node]][i];
        }
    }

    //Next fail link that has an endpoint word
    /*vector<int> inQue(MAX_N);
    inQue[1] = 1;
    nxt[1] = 0;
    q.push(1);
    while (!q.empty()) {
        int node = q.front(); q.pop();
        if(~isEnd[fail[node]]) nxt[node] =
                                    fail[node];
        else nxt[node] = nxt[fail[node]];
        for (int i = 0; i < SIGMA; i++) {
            if (trie[node][i] and
                    !inQue[trie[node][i]]) {
                q.push(trie[node][i]);
                inQue[trie[node][i]] = 1;
            }
        }
    }*/
```

```
    hash_st h2 = hash_st(s, MOD, 73);
    h2.init();
    h2.init_reverse();

    ll hs = h1.get_hash(1, n), hsr =
h1.get_hash(n, 1);
    ll hss = h2.get_hash(1, n), hssr =
h2.get_hash(n, 1);

    cin >> q;
    while(q--){
        cin >> a >> b;

        cout << h1.get_hash(a, b) << " ";
        cout << h2.get_hash(a, b) << endl;
    }
}
```

```
    for (int i = 2; i <= nodes; i++)
        g[fail[i]].push_back(i);

}

void search(string &s) {
    int node = 1, idx = 1;
    for (char ch : s) {
        node = trie[node][ch - 'a'];
        seen[node]++;

        /* Indexes of occurrences
        for(int i=node; i; i=nxt[i])
            if(~isEnd[i])
               occ[isEnd[i]].push_back(idx); */
        idx++;
    }
}

int dfs(int node) {
    int sol = seen[node];
    for (int son : g[node]) sol += dfs(son);
    for (int idx : leaf[node]) ans[idx] = sol;
    return sol;
}


int main() {
    ll j, i, n;

    string s;
    cin >> s >> n;

    vector<string> vs(n);
    memset(isEnd, -1, sizeof(isEnd));
    for(i=0;i<n;i++){
        cin >> vs[i];
        add_word(vs[i], i);
    }

    build();
    search(s);
    dfs(1);

    for (int i = 0; i < n; i++) {
        /* cout << occ[i].size() << endl;
        for(auto x : occ[i])
            cout << x-vs[i].size()+1 << " "; */
        if(ans[i]) cout << "YES" << endl;
        else cout << "NO" << endl;
    }
}
```

```
//K-nary XOR Hashing
//https://codeforces.com/contest/1418/problem/G
long long rng() {
  static std::mt19937 gen(
std::chrono::steady_clock::now().time_since_epoch().count());
return std::uniform_int_distribution<long long>(0, INT64_MAX)(gen);
}

int main() {
    using hash_t = uint64_t;

    int N;
    cin >> N;
    vector<hash_t> hash_values(N + 1);

    for (int i = 0; i <= N; i++)
        hash_values[i] = rng();

    vector<int> freq(N + 1, 0);
    vector<queue<int>> indices(N + 1);
    vector<hash_t> hashes(N + 1, 0);
    unordered_map<hash_t, int> hash_freq;
    hash_freq[hashes[0]]++;
    int64_t answer = 0;
    int start = 0;

  for (int i = 0; i < N; i++) {
      int a; cin >> a;
      if (indices[a].size() >= 3) {
          int remove = indices[a].front();
          indices[a].pop();

            while (start <= remove) {
                hash_freq[hashes[start]]--;
                start++;
            }
        }

      int before = freq[a];
      freq[a] = (freq[a] + 1) % 3;
      hashes[i + 1] = hashes[i]
       + (freq[a]-before)*hash_values[a];
      answer += hash_freq[hashes[i + 1]];
      hash_freq[hashes[i + 1]]++;
      indices[a].push(i);
  }

  cout << answer << '\n';
}
```

```
// Function to return the lexicographically
minimal rotation of a string
string minimal_rotation(string s) {
    int n = s.size();
    s += s;
    vector<int> f(2 * n, -1);
// Failure function
    int k = 0;
// Least rotation of string found so far

  for (int j = 1; j < 2 * n; ++j) {
     char sj = s[j];
     int i = f[j - k - 1];
     while (i != -1 && sj != s[k + i + 1]) {
         if (sj < s[k + i + 1]) {
             k = j - i - 1;
         }
         i = f[i];
     }
     if (sj != s[k + i + 1]) {
         if (sj < s[k]) {
                k = j;
         }
         f[j - k] = -1;
     } else {
         f[j - k] = i + 1;
     }
  }

  return s.substr(k, n);
}

int main() {
    string s;
    cin >> s;
    cout << minimal_rotation(s) << endl;
    return 0;
}
```

```
void manacher(string s)
{
  string arr;
  for (int i = 0; i < s.size(); i++){
      arr.push_back('#');
      arr.push_back(s[i]);
  }
  arr.push_back('#');
  // dp[i] = palindromic substring length
  //     centered at i (with '#')
  vector<int> dp(arr.size());
  int left = 0, right = 0;
  for (int i = 0; i < arr.size();){
      while (left > 0 && right < arr.size() - 1
           && arr[left - 1] == arr[right + 1])
             left--, right++;
      dp[i] = right - left + 1;
   int new_center = right + (i%2 == 0 ? 1 : 0);
    for (int j = i + 1; j <= right; j++){
        dp[j] = min(dp[i - (j - i)], 2 *
                    (right - j) + 1);
        if (j + dp[i - (j - i)] / 2 == right){
            new_center = j;
            break;
        }
     }
     i = new_center;
     right = i + dp[i] / 2;
     left = i - dp[i] / 2;
 }
```

```
 int mx = 0, idx;
 for(int i=0;i<dp.size();i++){
     cout<<dp[i]<<' ';
     if(mx<dp[i]){
         mx = dp[i];
         idx = i;
     }
 }
 string ans = "";
 for (int j = idx-dp[idx]/2;j<=idx+dp[idx]/2;
           j++){
   if (arr[j] != '#') ans.push_back(arr[j]);
 }
 cout<<ans;
}
```

```
// MISC. Algo
// MinMaxSubSum
void solve(ll cs){
    ll j, i, d, m, n, k, mn = 0, mx =
998244353;

    cin >> n;
    vector<ll> v(n), pre(n);
    for(auto &x : v) cin >> x;

    // at least length 2
    mn = 0, mx = 0;
    ll sum = 0;
    for(i=0;i<n;i++){
        sum += v[i];
        pre[i] = sum;
        if(-1<i-2) mx = max(mx, pre[i-2]);
        if(i>0) mn = min(mn, sum-mx);
    }

    p = 0;
    for(auto x : v) p += x;
    p -= mn;

    cout << p+abs(mn) ;

}
```

```
//Bitset c++
    bitset <5> a("11010");
    bitset <5> b(14);
    cout<<b<<" is b\n";
    for(int i=4;i>=0;i--) cout<<b[i]<<" \n"[i];
    cout<<b.count()<<" set bits\n";
    b.set(); // sets all bits to 1 (11111)
    b.reset(); // resets all bits to 0(00000)
    b.flip(); // flips all bits (11111)

    bitset <3> b8(8); // overflow (b = 000)
    bitset<5> c(10); // (01010)
    c<<=1; // (10100)
    c>>=2; // (00101)
    c &= 6; // (00100)
    c |= 10; // (01110)
    c = ~c; // (10001)
    c ^= 5; // (10100)
    bitset<5> d(12); // (01100)
    c |= d; // (11100)
    cout<<c.to_string()<<" is c in string\n";
    cout<<c.to_ullong()<<" is c in unsigned long
\n";

for(int i=c._Find_first();
    i<c.size();i=c._Find_next(i)) cout<<i<<' ';
    cout<<" are set bits in c\n";
    if(c.all()) cout<<"all are set in c\n";
    if(c.any()) cout<<"c has atleast one set
bit\n";
```

```cpp
//NextGreaterElements
ll NGE[300005];

void nextGreaterElement(int arr[], int n)
{
    stack<pair<int, int>> s;
    s.push({arr[0], 0});

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

        if (s.empty()) {
            s.push({arr[i], i});
            continue;
        }
        while (s.empty() == false && s.top().fi <
arr[i]) {
            nxt[s.top().se] = i;
            s.pop();
        }
        s.push({arr[i], i});
    }
    while (s.empty() == false) {
        nxt[s.top().se] = n;
        s.pop();
    }
}
```

```cpp
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
using vi = vector<ll>;
using pii = pair<ll, ll>;

#define pb push_back
#define fi first
#define se second
#define endl '\n'

const int MAXN = 3e5 + 5;
const int MOD = 1e9 + 7;

void run_case(int cases) {

}

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);

    int T = 1;
    cin >> T;

    for (int cases = 1; cases <= T; ++cases)
        run_case(cases);

    return 0;
}
```

```cpp
//Chinese Remainder Theorem O(n*log(max_modVal))
const int N = 20;
ll GCD(ll a, ll b){return (b==0)?a:GCD(b, a%b); }
ll LCM(ll a, ll b) {return a/GCD(a, b)*b; }
ll normalize(ll x, ll mod){
  x%=mod; if(x<0) x+= mod; return x;}
struct GCD_type { ll x, y, d; };
GCD_type ex_GCD(ll a, ll b){
  if (b == 0) return {1, 0, a};
  GCD_type pom = ex_GCD(b, a % b);
  return {pom.y, pom.x-a/b*pom.y, pom.d};
}
ll rem[N], modVal[N], ans, lcm;
int main()
{
  ios_base::sync_with_stdio(0); cin.tie(0);
  int n; cin >> n;
  for(int i = 1; i <= n; i++)
  cin>>rem[i]>>modVal[i];
  ans = rem[1]; lcm = modVal[1];
  for(int i = 2; i <= n; i++){
    auto pom = ex_GCD(lcm, modVal[i]);
    int x1 = pom.x, d = pom.d;
    if((rem[i] - ans) % d != 0)
      return cerr<<"No solutions"<<endl, 0;
    ans=normalize(ans+x1*(rem[i]-ans)/d%
         (modVal[i]/d)*lcm, lcm*modVal[i]/d);
    lcm = LCM(lcm, modVal[i]);
  }
  cout << ans << " " << lcm << endl;
}
```