1> Segment tree

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

struct tree{
  int left;
  int right;
  int val;
  tree*l;
  tree*r;
  tree (int a, int b){
      left=a;
      right=b;
      val=0;
      l=NULL;
      r=NULL;
  }
};

tree* cons(vector<int> &V, int lval, int rval){
    if(lval>=rval)
      return NULL;
    tree* T=new tree(lval, rval-1);

    if(lval+1==rval)
    {
        T->val=V[lval];
        return T;
    }
    int mid=lval+(rval-lval)/2;
    T->l=cons(V, lval, mid);
    T->r=cons(V, mid, rval);

    if(T->l)
    T->val=T->l->val;
    if(T->r)
    T->val+=T->r->val;
    return T;
}

int get(tree* T, int a, int b){
    if(!T)
      return -1;

    if(a<T->left || b>T->right)
      return -1;
    if(T->left==a && T->right==b)
       return T->val;
```

```
    if(b <= T->l->right)
      return get(T->l, a, b);
    if(a > T->l->right)
      return get(T->r, a, b);

    return get(T->l, a, T->l->right)+get(T->r, T->r->left, b) ;

}

void update(tree* &T, int index, int change){
    if(!T)
      return;
    if((index < T->left) || (index > T->right))
      return ;
    T->val+=change;

    update(T->l, index, change);
    update(T->r, index, change);

      return ;
}

int main(){
    vector<int> V;
    for(int i=0;i<15;++i)
       V.push_back(i);
    tree*T=cons(V, 0, V.size());

    cout<<get(T,1,1)<<" ";
    update(T, 5, 3);
    cout<<get(T,3,9)<<" ";

    cout<<T->val<<" "<<T->left<<" "<<T->right;
    return 0;
}
```

2> Trie

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

struct trie{
    int count=0;
    trie* next[26]={NULL};
};

void insert(trie* &root, string &S, int index){
    if(S.size()==index)
      return ;
    if(root->next[S[index]-'a'])
    {
       root->next[S[index]-'a']->count++;
       insert(root->next[S[index]-'a'], S, index+1);
       return ;
    }
    trie * temp= new trie;
    temp->count=1;
    root->next[S[index]-'a']=temp;
    insert(root->next[S[index]-'a'], S, index+1);
       return ;
}

int count_p(trie*root, string &S, int index){
    if(index==S.size())
      return root->count;
    if(root->next[S[index]-'a']==NULL)
       return root->count;
    return count_p(root->next[S[index]-'a'], S, index+1);
}

int main(){
    trie * root= new trie;
    string S;
    for(int i=1;i<=5;++i)
    {cin>>S;
    insert(root, S, 0);}
    cin>>S;
    cout<<count_p(root,S,0);
    return 0;
}
```

3> Dsijoint set union

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

int par_find(vector<int>& parent,int a)
{
    if(parent[a] == a)
      return a;
    return parent[a] = par_find(parent,parent[a]);
}

void unify(vector<int> &parent, int x, int y, vector<int> &val){
    x = par_find(parent,x);
    y = par_find(parent,y);
    if(x != y)
    {
        val[x] += val[y];
        parent[y] = x;
    }
}

int main(){
    int n;
    cin>>n;
    vector<int> val(n);
    for(int i=0;i<n;++i)
      cin>>val[i];
    vector<int> parent(n);
    for(int i=0;i<n;++i)
       parent[i]=i;
    int k;
    cin>>k;
    while(k){
        k--;
        int x, y;
        cin>>x>>y;
        unify(parent,x,y,val);
    }
    vector<int> val_final;
    for(int i=0;i<n;++i)
    {
```

```
        if(i==parent[i])
          val_final.push_back(val[i]);
    }
    for(int i=0;i<val_final.size();++i)
       cout<<val_final[i]<<" ";
}
```

4> > Inverse modulo, compute nCk modp

nCk%mod = n!%(mod)/(n-k)!%(mod)k!%mod = n!%(mod) * power(fact(n-k), mod-2)%mod + power(fact(k), mod-2)%mod

inv(a) = inverse modulo a w.r.t. mod
inv(d) = power(d,mod-2)%mod

```
long long int power(long long int A,long long int B)
{
    //base case
    long long result=1;
    if(A==0)
    {
        return 0;
    }
    if(B==0)
    {
        return 1;
    }

    if(B%2==0)
    {
        result  =power(A,B/2);
        result = (result* result )%mod;
    }
    else
    {
        result = ((A %mod)* power(A,B-1)%mod)%mod;
    }
```

```
    return result%mod;
}
```

5>
```
 * Definition for binary tree
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
```

6> Definition of trie

```
struct Trie{
    int c=0;
    Trie *pt[26]={NULL};
};
```

7>
```
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
```

8> Multi child tree

```
struct Node {
     int val;
     vector<Node*> child;
};

Node *newNode(int key)
{
    Node *temp = new Node;
    temp->val = key;
    return temp;
}

use Node*temp=newNode(value)
```

9> Max element in every window

```
#define f first
#define s second

vector<int> Solution::slidingMaximum(const vector<int> &A, int B) {
vector<int> ans;
deque<pair<int, int>> dq;
for (int i = 0; i < A.size(); i++) {

    if (!dq.empty() && dq.front().s == i-B)
        dq.pop_front();

    while (!dq.empty() && dq.back().f < A[i])
        dq.pop_back();

    dq.push_back({A[i], i});

    if (i-B+1 >= 0) ans.push_back(dq.front().f);
}
return ans;
```

```
}
```

10> kth ancestor, lca

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

struct node{       // ye h node
    int val;
    node*left;
    node*right;
    node(int key){
        val=key;
        left=NULL;
        right=NULL;
    }
};

void dfs(node*root){    // yaha apan dekhenge ki tree sahi se bana ya nhi
    if(!root)
      return;
    dfs(root->left);
    cout<<root->val<<" ";
    dfs(root->right);
    return ;
}


vector<vector<int> > up;     // up hmare purvaj hai, up[i][j] is jth purvaj
void TreeAncestor(int n, vector<int>& parent) {
        up = vector<vector<int>> (n, vector<int>(20));  //20=ceil(log2(n))
        for(int src = 0; src < n; src++) up[src][0] = parent[src];

        for(int u = 1; u < 20; u++){
            for(int src = 0; src < n; src++){
                if(up[src][u-1] == -1) up[src][u] = -1;
                else up[src][u] = up[up[src][u-1]][u-1];
            }
        }
```

```
}

int getKthAncestor(int nodes, int k, vector<int> &value) {   //kth purvaj ki value
dedo bhai
        for(int i = 0; i < 20; i++){
            if((1<<i) & k){
                nodes = up[nodes][i];
                if(nodes == -1) return -1;
            }
        }
        return value[nodes];
}

unordered_set<int> S;
int lca(int a, int b){     // lca purvaj ki value dedo bhai
    S.clear();

    for(int i=1;i<20;++i)
    {
        if(up[a][i]==-1)
          break;
        if(up[a][i]==b)
          return b;
        S.insert(up[a][i]);
    }

    for(int i=1;i<20;++i)
    {
        if(up[b][i]==-1)
          break;
        if(up[b][i]==a || (S.find(up[b][i])!=S.end()))
          return up[b][i];
        S.insert(up[b][i]);
    }
    return 0;  //maze me
}

int main(){

    int n;
    cin>>n;
    vector<int> par(n), value(n);    // par purvaj h, value node ki value h
    for(int i=0;i<n;++i)
      cin>>value[i]>>par[i];

    node*root=new node(0);
    int index;
    for(int i=0;i<n;++i)
    {
        if(par[i]==-1)
```

```
    {
       root->val=value[i];
       index=i;
       break;
    }
}

queue<pair<int, node*>> q;       // bfs se tree bnaenge

q.push({index, root});

while(q.size()){
    int ind=q.front().first;
    node*T=q.front().second;
    q.pop();
    int flag=0;
    for(int i=0;i<n;++i)
    {
        if(par[i]==ind)
        {
            node*temp=new node(value[i]);
            if(!flag)
              T->left=temp;
            else
              T->right=temp;
            flag++;
            q.push({i,temp});
        }
    }
}
dfs(root);                // nodes dekho laundo
up.clear();               // purvajo ka dehant
TreeAncestor(n, par);     // purvajo ki sthapna
int k, nodes;

cin>>k>>nodes;           // nodes value h us node ki, kth purvaj chahiye
for(int i=0;i<n;++i)
   if(value[i]==nodes)
   {
       nodes=i;
       break;
   }
cout<<getKthAncestor(nodes, k, value);

int w,d,flag=0;               // lca nikalenge ab
cin>>w>>d;
for(int i=0;i<n;++i)
   if(value[i]==w)
   {
       w=i;
```

```
            flag++;
            break;
        }
    for(int i=0;i<n;++i)
        if(value[i]==d)
        {
            d=i;
            flag++;
            break;
        }

    if(flag<2)
      cout<<endl<<-1;
    else
      cout<<endl<<value[lca(d,w)];
    return 0;

}
```