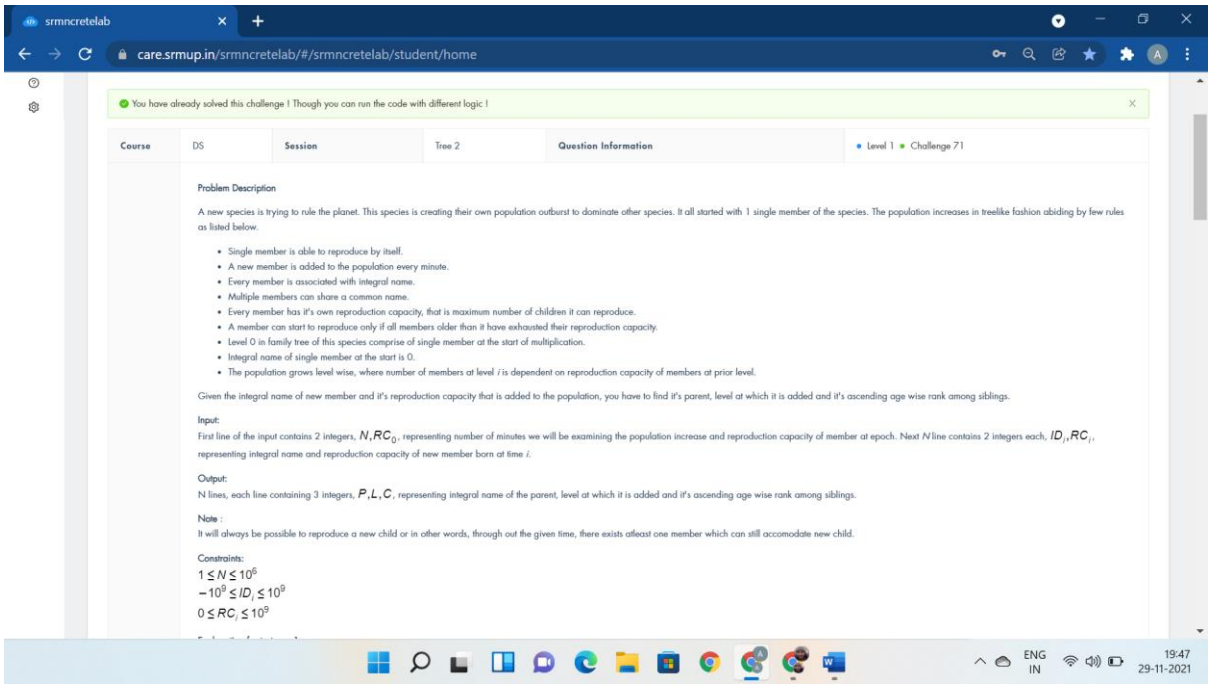
TREE-2:-



srmncretelab

care.srmup.in/srmncretelab/#/srmncretelab/student/home

You have already solved this challenge ! Though you can run the code with different logic !

Course	DS	Session	Tree 2	Question Information	Level 1 • Challenge 71

Problem Description

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

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

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

Input:

First line of the input contains 2 integers, N, RC_0, representing number of minutes we will be examining the population increase and reproduction capacity of member at epoch. Next N line contains 2 integers each, ID_i, RC_i, representing integral name and reproduction capacity of new member born at time i.

Output:

N lines, each line containing 3 integers, P, L, C, representing integral name of the parent, level at which it is added and it's ascending age wise rank among siblings.

Note :

It will always be possible to reproduce a new child or in other words, through out the given time, there exists atleast one member which can still accomodate new child.

Constraints:

$1 \le N \le 10^6$

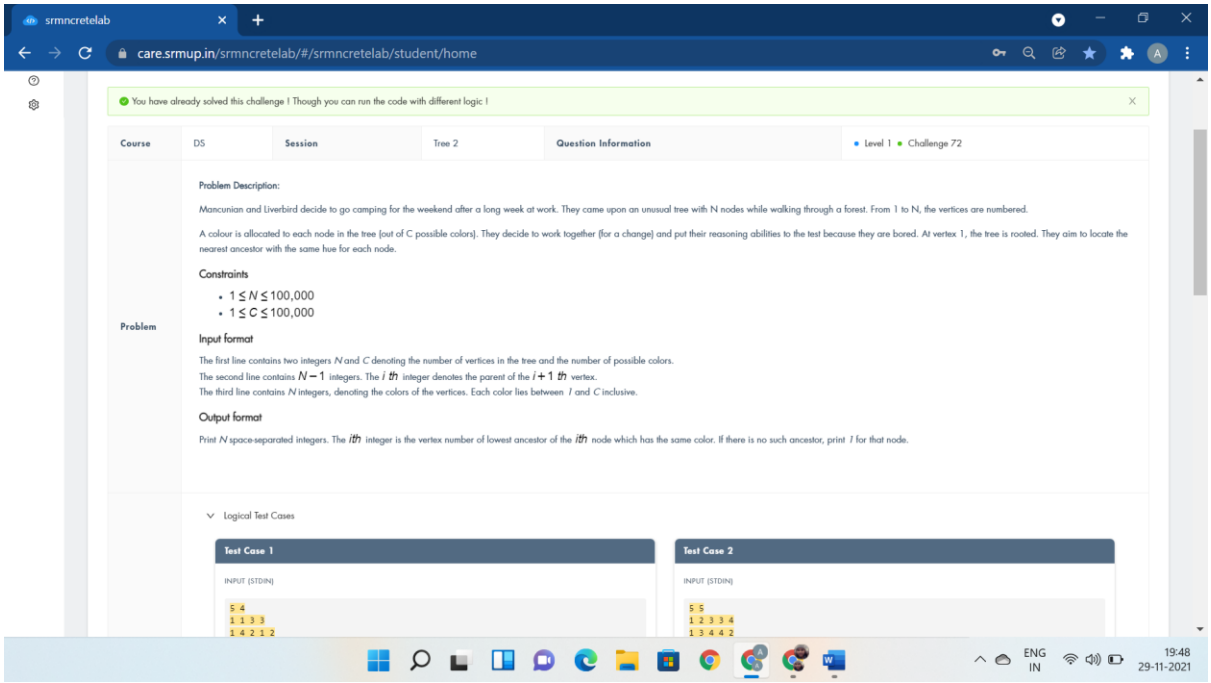
$-10^9 \le ID_i \le 10^9$

$0 \le RC_i \le 10^9$

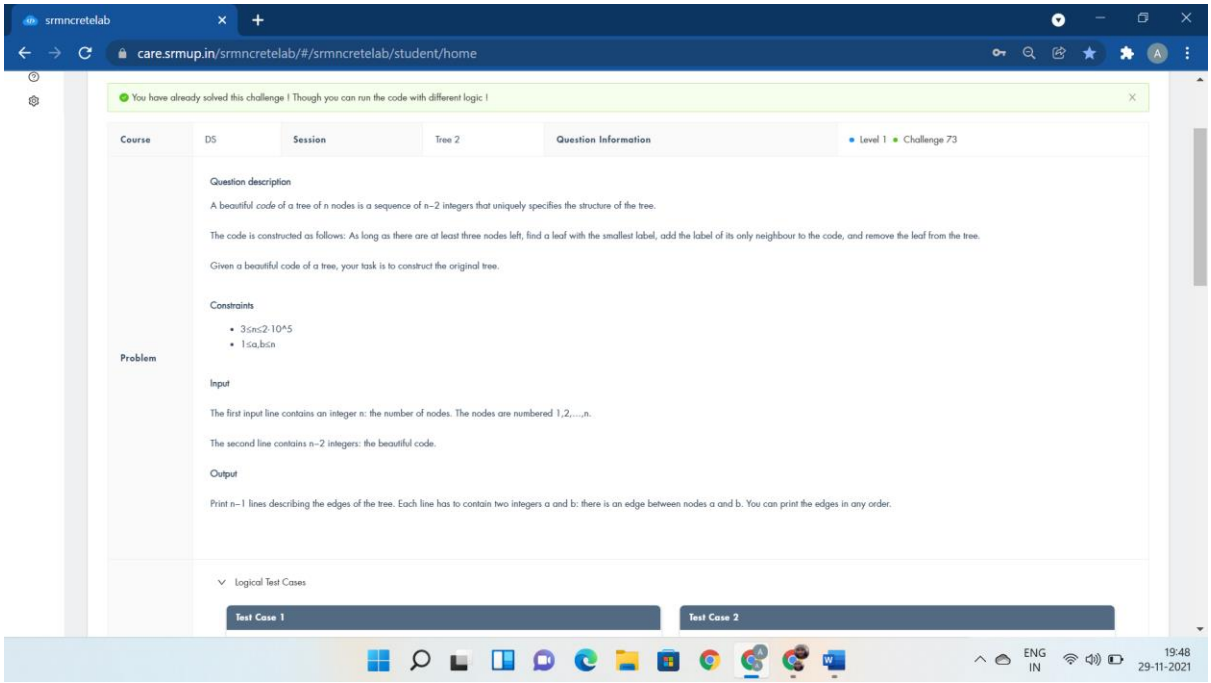
ENG IN 19:47 29-11-2021

```
#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct cell{

   int name;

   int level;

   int capacity;

};

struct cell queue[1000001];

struct cell arr[1000001];

int front;

int end;

void init_queue(){

   front = 0;

   end = 0;

}

void enqueue(int name,int capacity,int level){

   queue[end].name = name;

   queue[end].level = level;

   queue[end].capacity = capacity;

   end = end + 1;
```

```
}
int is_empty(){
   if(end == front)
      return 1;
      return 0;
}
void dequeue()
{
                if(!is_empty())
                 front++;
}


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

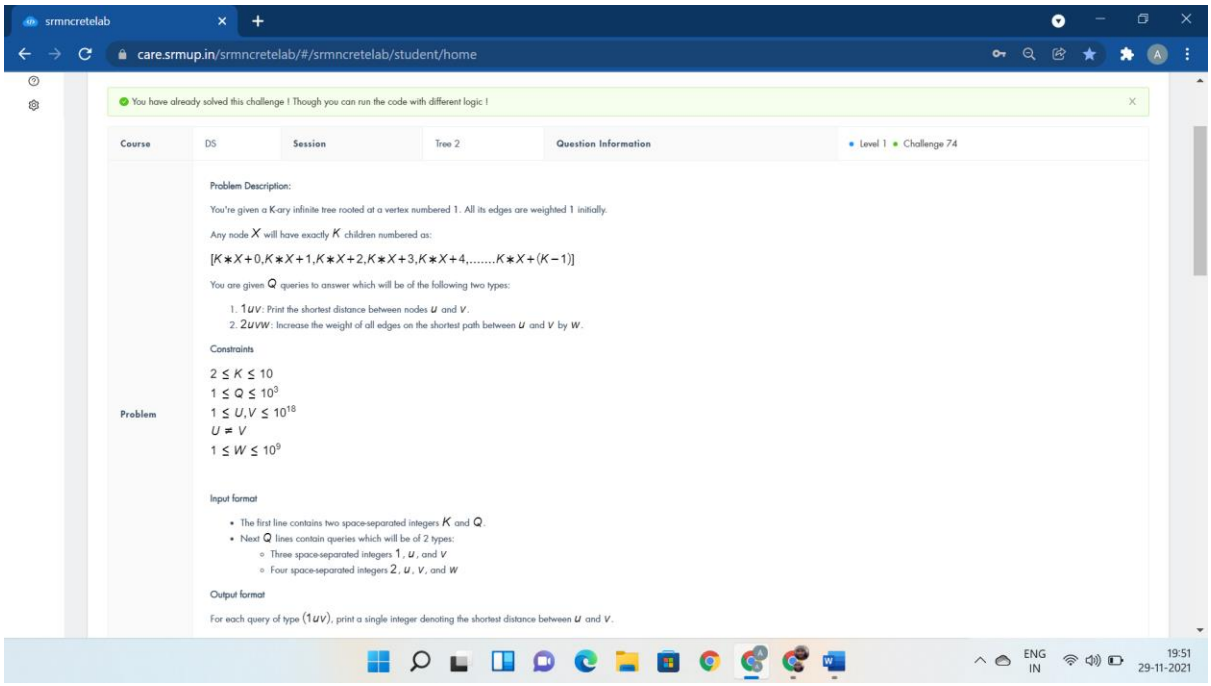



```
#include<bits/stdc++.h>
using namespace std;
int main() {
        int n,i,c;
        scanf("%d %d", &n, &c);
        int tree[n+1][2];
        tree[1][0] = -1;
        for(i=2;i<=n;i++) {
                scanf("%d", &tree[i][0]);
        }
        for(i = 1; i <= n; i++) {
                scanf("%d", &tree[i][1]);
        }
        int parent;
        for(i = 1; i<= n; i++) {
                parent = tree[i][0];
                while(parent != -1 && tree[parent][1] != tree[i][1]) {
                        parent = tree[parent][0];
                }
                printf("%d ", parent);
        }
        return 0;
}
```




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

using namespace std;

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

#define X(a,b) if(a==b)

vector< int > vi;

const int maxN = 2e5+5;


int N, a[maxN], deg[maxN];

priority_queue<int, vector<int>, greater<int>> q;


int main(){

  scanf("%d", &N);

  fill(deg+1, deg+N+1, 1);

  //for(int i = 0; i < N-2; i++)

  f(i,0,N-2){

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

    deg[a[i]]++;

  }


  //for(int i = 1; i <= N; i++)

  f(i,1,N+1)

    //if(deg[i] == 1)

    X(deg[i],1)
```

```
        q.push(i);

  f(i,0,N-2){
    int u = a[i];
    int v = q.top(); q.pop();

    deg[u]--; deg[v]--;
    //if(deg[u] == 1)
    X(deg[u],1)
        q.push(u);

    printf("%d %d\n", v, u);
  }

  //for(int i = 1; i <= N; i++)
  f(i,1,N+1)
    if(deg[i])
        printf("%d ", i);
}
```

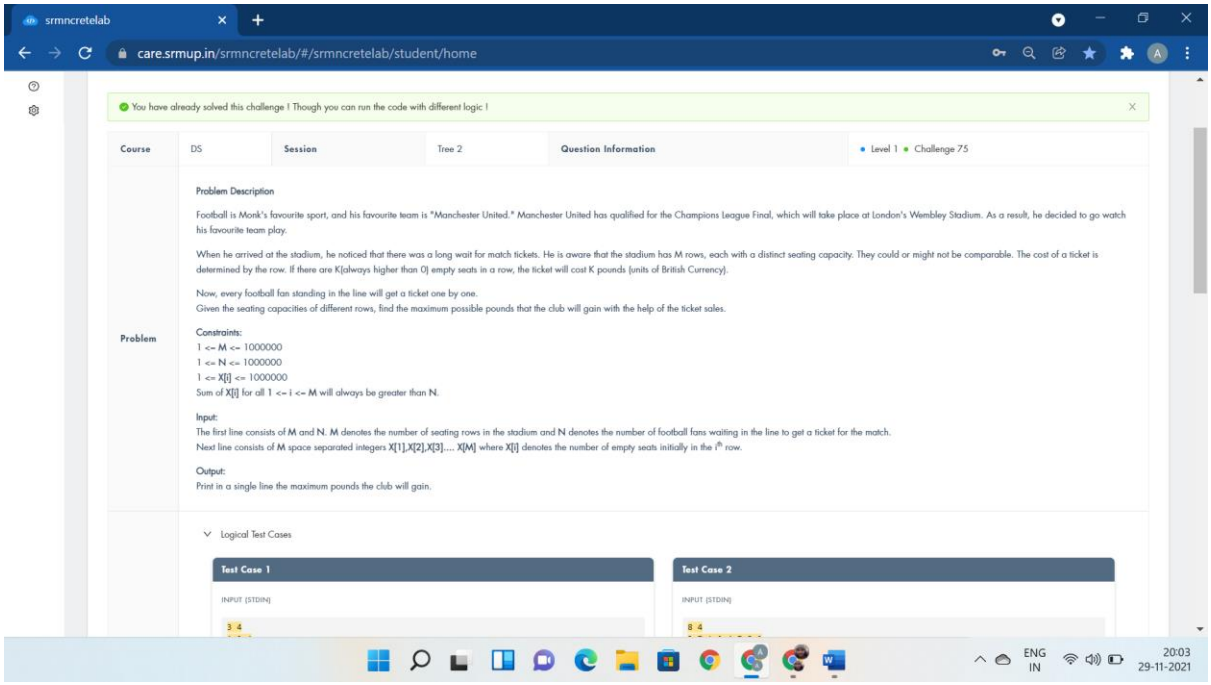



```
#include <iostream>
#include <map>
#include <assert.h>
```

```
using namespace std;
#define int long long

map < pair < int, int >, int > adj;

int find_depth( int u, int k ) {
   int depth = 0;
   while ( u > 0 ) {
      u = u / k;
      depth = depth + 1;
   }
   return depth - 1;
}

int dist( int u, int v, int k ) {
   int dist = 0;
   int depth_u = find_depth( u, k );
   int depth_v = find_depth( v, k );
   if ( depth_u < depth_v ) {
   swap ( u, v );
   swap ( depth_u, depth_v );
   }
   while( depth_u != depth_v ) {
      if ( adj.count( { u, u / k } ) ) {
      dist = dist + adj[ { u, u / k } ];
   } else {
      dist = dist + 1;
   }
   depth_u = depth_u - 1;
   u = u / k;
   }
   while ( u != v ) {
   if ( adj.count( { u, u / k } ) ) {
      dist = dist + adj [ { u, u / k } ];
   } else {
      dist = dist + 1;
   }
   if ( adj.count( { v, v / k } ) ) {
```

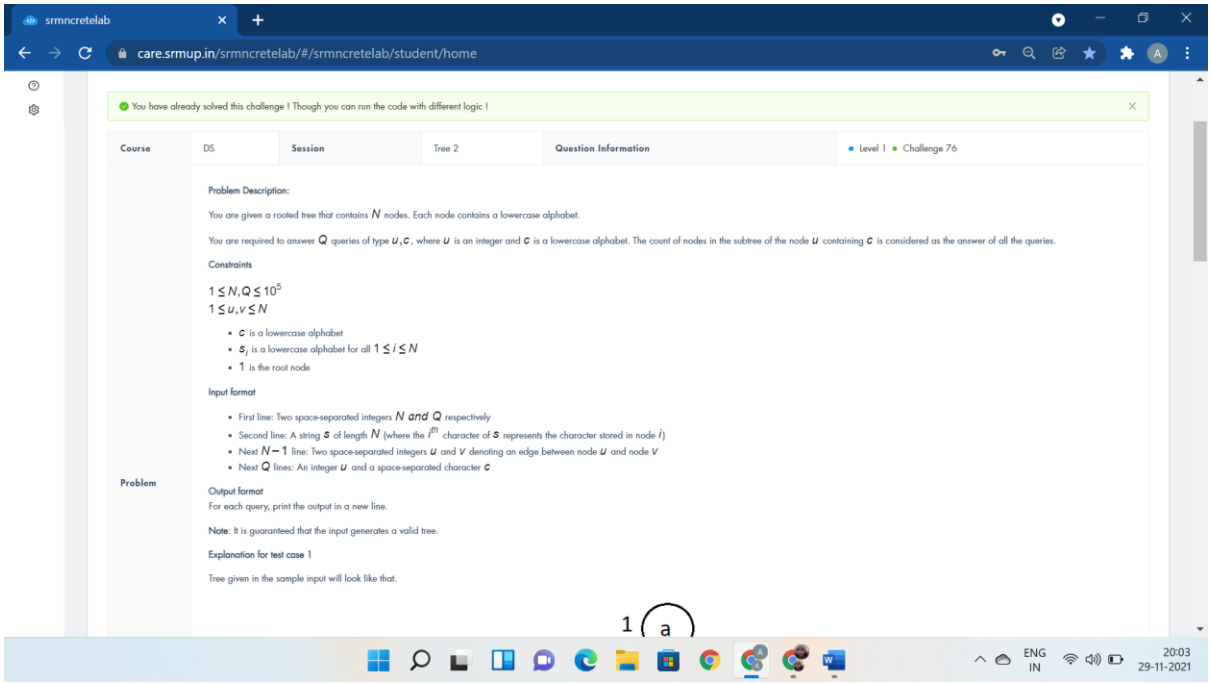
```
            dist = dist + adj [ { v, v / k } ];
        } else {
            dist = dist + 1;
        }
        u = u / k;
        v = v / k;
    }
    return dist;
}

void add_weight( int vertex, int parent, int w ) {
    if ( !adj.count ( { vertex, parent } ) ) {
        adj[ { vertex, parent } ] = 1;
    }
    adj[ { vertex, parent } ] = adj[ { vertex, parent } ] + w;
}

void increase_weights ( int u, int v, int w, int k ) {
    int depth_u = find_depth( u, k );
    int depth_v = find_depth( v, k );
    if ( depth_u < depth_v ) {
    swap ( u, v );
    swap ( depth_u, depth_v );
    }
    while( depth_u != depth_v ) {
        add_weight( u, u / k, w );
    depth_u = depth_u - 1;
    u = u / k;
    }
    while ( u != v ) {
    add_weight( u, u / k, w );
    add_weight( v, v / k, w );
        u = u / k;
        v = v / k;
    }
}

signed main() {
```

```
  int k, q, x, u, v, w;
  cin >> k >> q;

  while(q--) {
    cin >> x;
    if ( x == 1 ) {
      cin >> u >> v;
      cout << dist( u, v, k ) << "\n";
    } else {
      cin >> u >> v >> w;
      increase_weights( u, v, w, k );
    }
  }
}
```




```
#include <bits/stdc++.h>
using namespace std;
#define PII pair <int, int>
priority_queue <int> seats;
map <int, int> x;
int main()
{
```

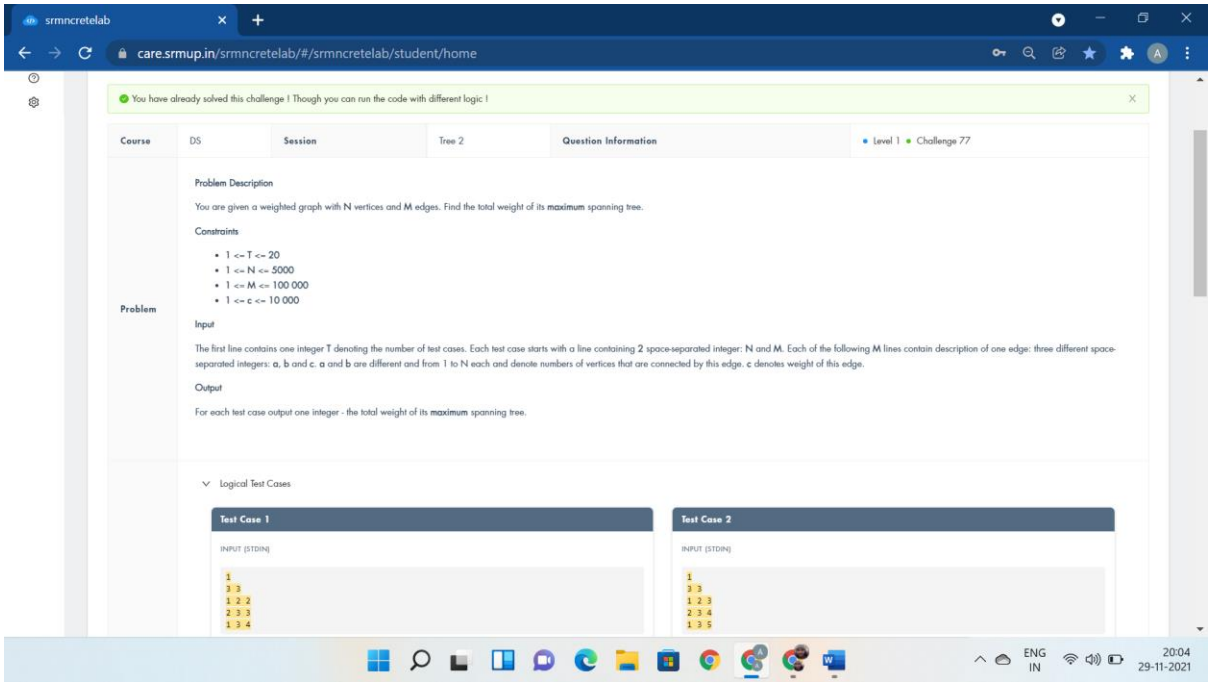
```
    int N, M; cin >> N >> M;
    assert (1<=N and N<=1000000);
    assert (1<=M and M<=1000000);
    for (int g=1; g<=N; g++){
        int a; cin >> a;
        seats.push(a);
        assert (1<=a and a<=1000000);
        x[a]++;
    }
    long long ans = 0;
    for (int g=0; g<M; g++){
        int x = seats.top(); ans+=x; seats.pop();seats.push(x-1);
    }
    cout <<ans;
    return 0;
    cout<<"void heapify(int arr[],int n,int i)";
}
```




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

void dfs(int node,int parent,string &s,vector<vector<int>>&subroot,vector<vector<int>>& v1)
{
```

```
        //visited[node]=1;
        subroot[node][s[node-1]-'a']++;
        //intime[node]=t;
        //t++;
        //z.push_back(node);
        for( auto it:v1[node])
        {
                if(it!=parent)
                {
                dfs(it,node,s,subroot,v1);
                for(int i=0;i<26;i++)
                subroot[node][i]+=subroot[it][i];
                }

        }
        //outtime[node]=t;
        //t++;

}
int main()
{
        int N,i, Q;
        string S;
        cin >> N >> Q;
        cin >> S;
        vector<vector<int>>v1(N+1);
        for(i=0;i<N-1;i++)
        {
                int u, v;
                cin >> u >> v;
          v1[u].push_back(v);
                v1[v].push_back(u);

        }

        vector<vector<int>>subroot(N+1,vector<int>(26,0));
        dfs(1,0,S,subroot,v1);
        while(Q--)
```

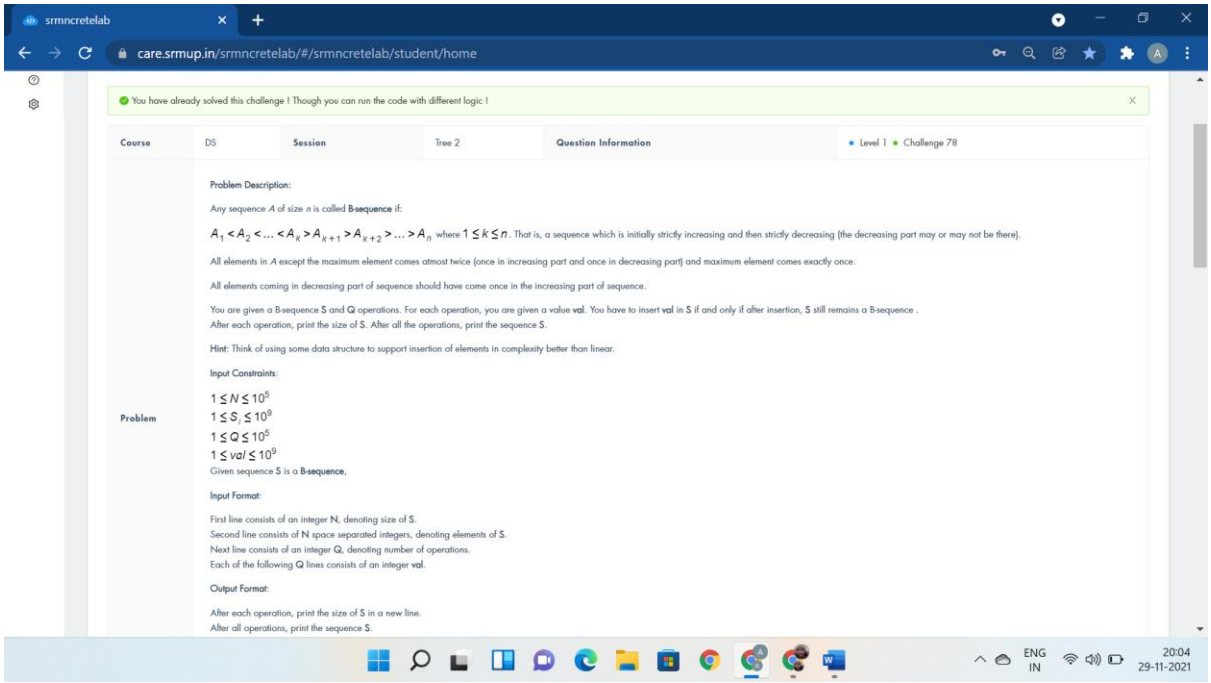
```
        {
                int u;
                char c;
                cin >> u >> c;
                cout<<subroot[u][c-'a']<<"\n";
                //cout<<cnt<<endl;



        }
        return 0;
}
```




```
#include<bits/stdc++.h>
typedef long long ll;
using namespace std;
struct edge
{
   int u;
   int v;
   int w;
};
edge a[100005];
int parent[100005];
```

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

```
    }
    cout<<ans<<endl;
  }
  return 0;
  cout<<"int printheap(int N)";
}
```



srmncretelab

care.srmup.in/srmncretelab/#/srmncretelab/student/home

You have already solved this challenge ! Though you can run the code with different logic !

Course	DS	Session	Tree 2	Question Information	Level 1 • Challenge 78

Problem

Problem Description:

Any sequence A of size n is called **B-sequence** if:

$A_1 < A_2 < \ldots < A_k > A_{k+1} > A_{k+2} > \ldots > A_n$ where $1 \le k \le n$. That is, a sequence which is initially strictly increasing and then strictly decreasing (the decreasing part may or may not be there).

All elements in A except the maximum element comes atmost twice (once in increasing part and once in decreasing part) and maximum element comes exactly once.

All elements coming in decreasing part of sequence should have come once in the increasing part of sequence.

You are given a B-sequence **S** and **Q** operations. For each operation, you are given a value **val**. You have to insert **val** in **S** if and only if after insertion, **S** still remains a B-sequence .
After each operation, print the size of **S**. After all the operations, print the sequence **S**.

Hint: Think of using some data structure to support insertion of elements in complexity better than linear.

Input Constraints:

$1 \le N \le 10^5$
$1 \le S_i \le 10^9$
$1 \le Q \le 10^5$
$1 \le val \le 10^9$
Given sequence **S** is a **B-sequence**,

Input Format:

First line consists of an integer **N**, denoting size of **S**.
Second line consists of **N** space separated integers, denoting elements of **S**.
Next line consists of an integer **Q**, denoting number of operations.
Each of the following **Q** lines consists of an integer **val**.

Output Format:

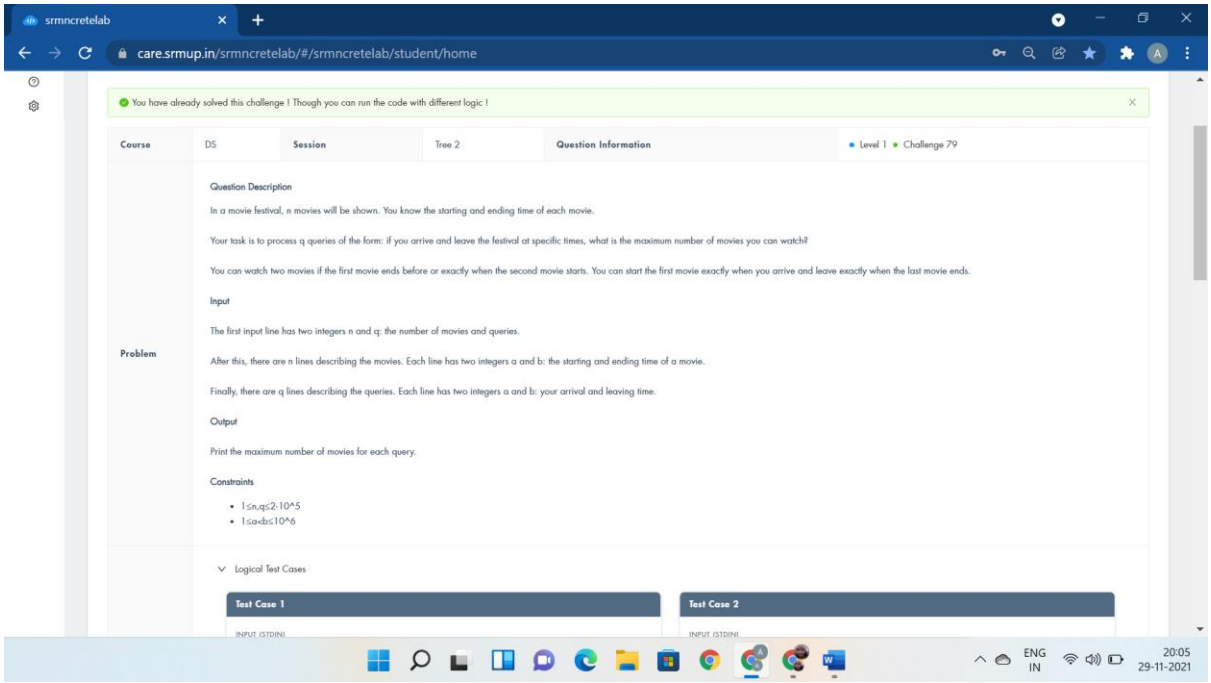
After each operation, print the size of **S** in a new line.
After all operations, print the sequence **S**.

ENG IN 20:04 29-11-2021

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


int main() {

  int N,i,maximum=INT_MIN;

  scanf("%d", &N);

  int S[N];

  map<int,int> map;
```

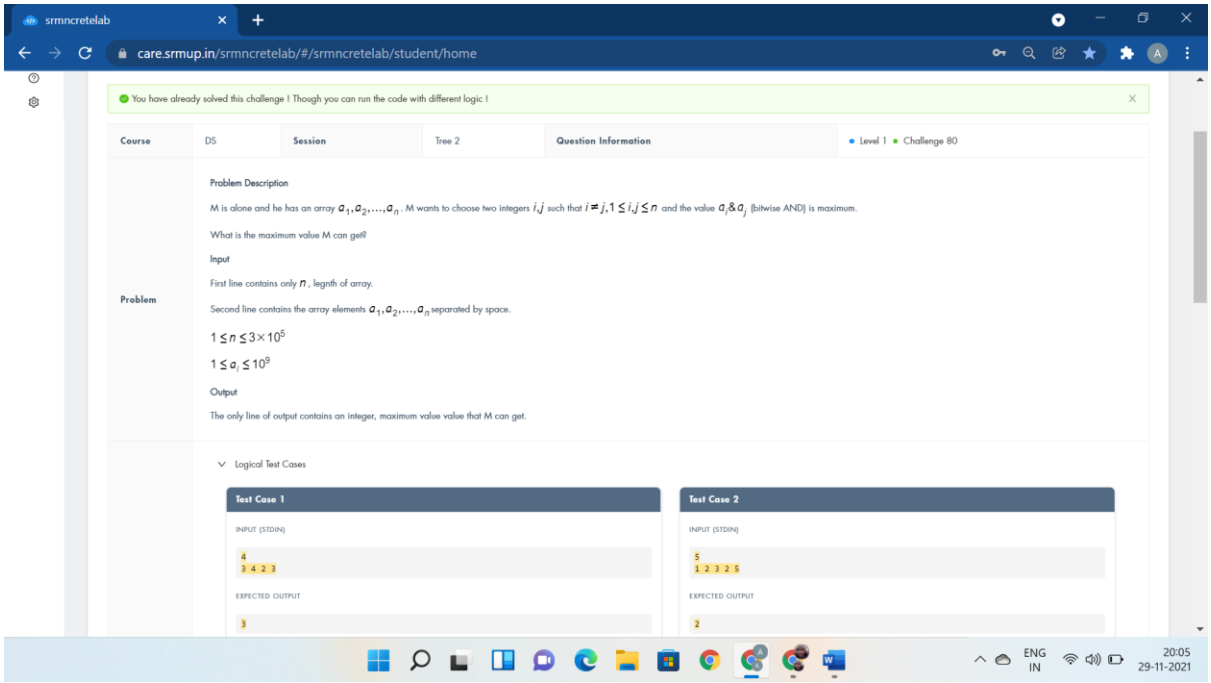
```
for(i=0;i<N;i++) {

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

    maximum=max(maximum,S[i]);

    map[S[i]]++;

}

int temp,Q;

cin>>Q;

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

    scanf("%d", &temp);

    if(temp==maximum) printf("%d\n",N);

    else {

        if(map[temp]>=2) printf("%d\n",N);

        else {

            map[temp]++;

            N++;

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

            maximum=max(maximum,temp);

        }

    }
```

```
  }

  for(auto it=map.begin();it!=map.end();it++) printf("%d ",it->first);

  for(auto it=map.rbegin();it!=map.rend();it++) {

     if(it->second>1) printf("%d ",it->first);

  }

}
```




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

using namespace std;

int dp[1000006][25];

void solve(){}

int main(){

   solve();

   int n, q; cin>>n>>q;

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

      int x, y; cin>>x>>y;
```

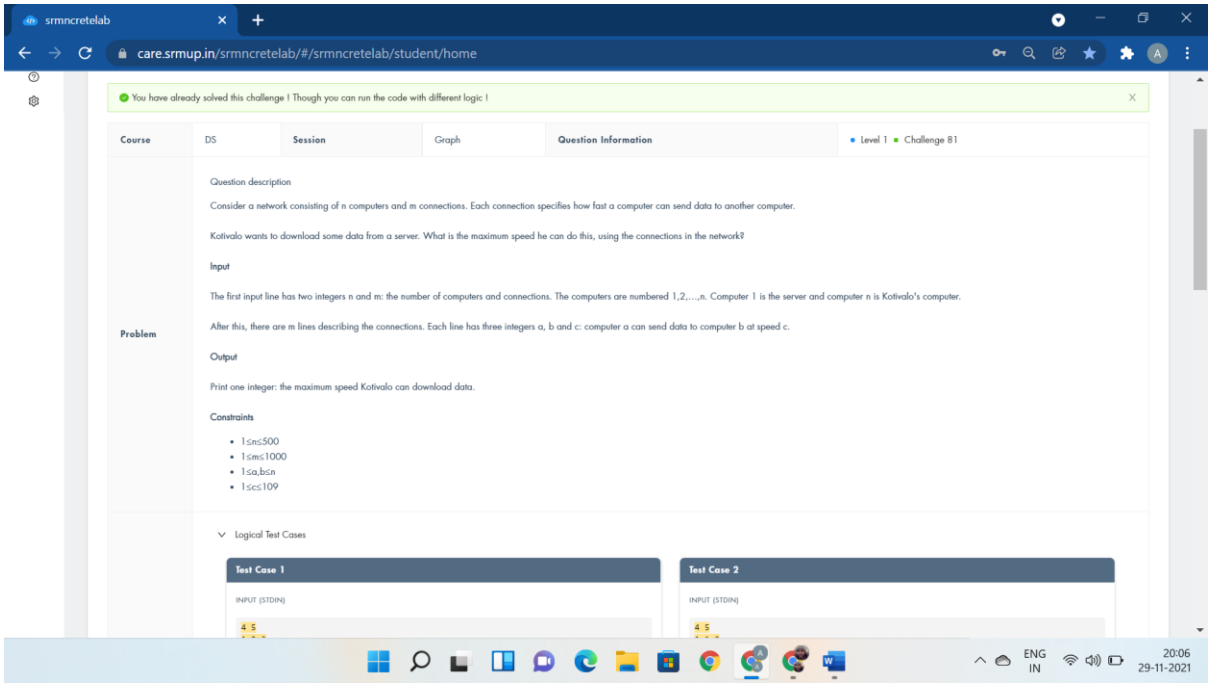
```
        dp[y][0] = max(dp[y][0], x);
    }
    for (int i = 1; i <= 1000000; i++)
        dp[i][0] = max(dp[i][0], dp[i-1][0]);

    for (int k = 1; k <= 20; k++)
        for (int i = 1; i <= 1000000; i++)
            dp[i][k] = dp[dp[i][k-1]][k-1];

    while(q--) {
        int x,y; cin>>x>>y;
        int ans = 0;
        while(y>0) {
            int z = 0;
            for (int i = 0; i <= 20; i++) {
                if (dp[y][i] < x) {
                    z = i;
                    break;
                }
            }
            if (z == 0)
                break;
            ans += (1<<(z-1));
            y = dp[y][z-1];
        }
        cout<<ans<<endl;
    }

}
```




```
# include<stdio.h>

# include<stdlib.h>

# include<math.h>

void input(long *,int);

int main()

{

   int n;


   scanf("%d",&n);

   long *ptr = (long*)malloc(n*sizeof(long));

   input(ptr,n);

            return 0;

}


void input(long *ptr, int n)

{

   int i, j;

            int m;

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

            {

                        scanf("%ld", ptr+i);

            }
```

```
for(i = 0; i < n; i++)
        {
                if (*(ptr + i) <= m)
                {
                        continue;
                }

                for (j = i + 1; j < n; j++)
                {
                        int temp = *(ptr + i) & *(ptr + j);
                        if(temp > m)
                        {
                                m = temp;
                        }
                }
        }

        printf("%d", m);
}
```

GRAPH:-




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

using namespace std;
```