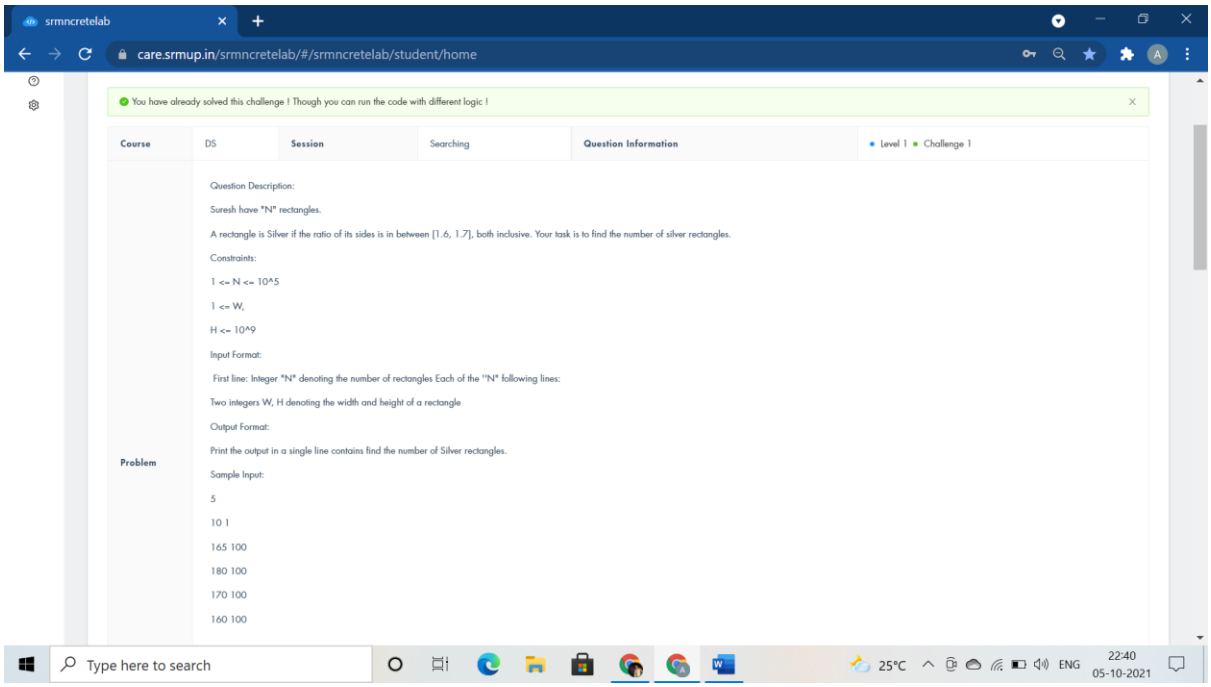
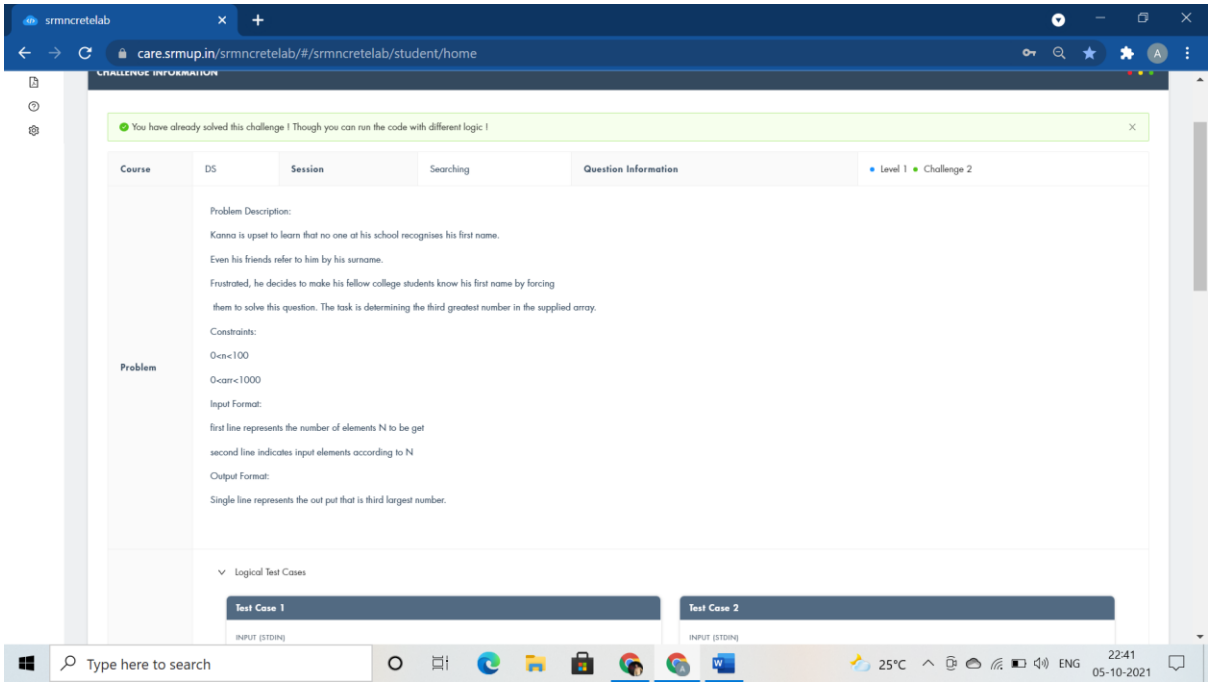
DSA SEARCHING

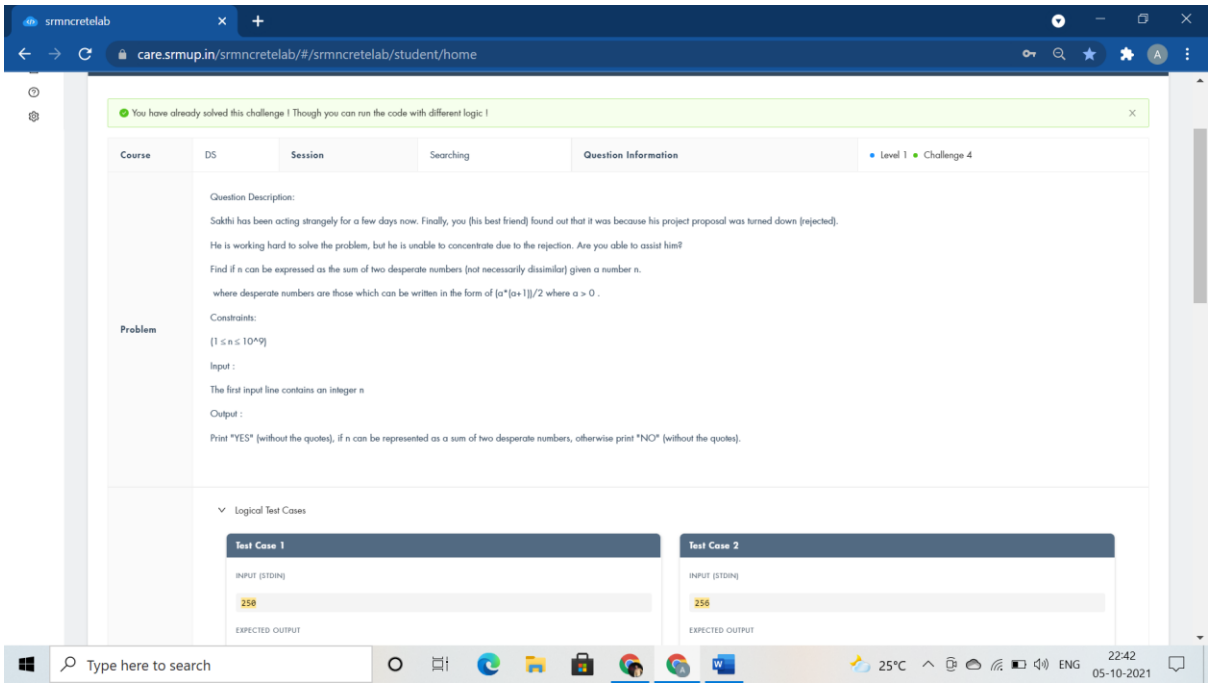



```
#include <stdio.h>
#include<math.h>
int main()
{
   float n,i,width,height;
   scanf("%f",&n);
   int count=0;
   for(i=0;i<n;i++)
   {
      scanf("%f %f",&width,&height);
      if(width/height>=1.6 && width/height<=1.7)
      ++count;
      else if(height/width >=1.6 && height/width<=1.7)
      ++count;
   }
   printf("%d",count+1);
         return 0;
}
```



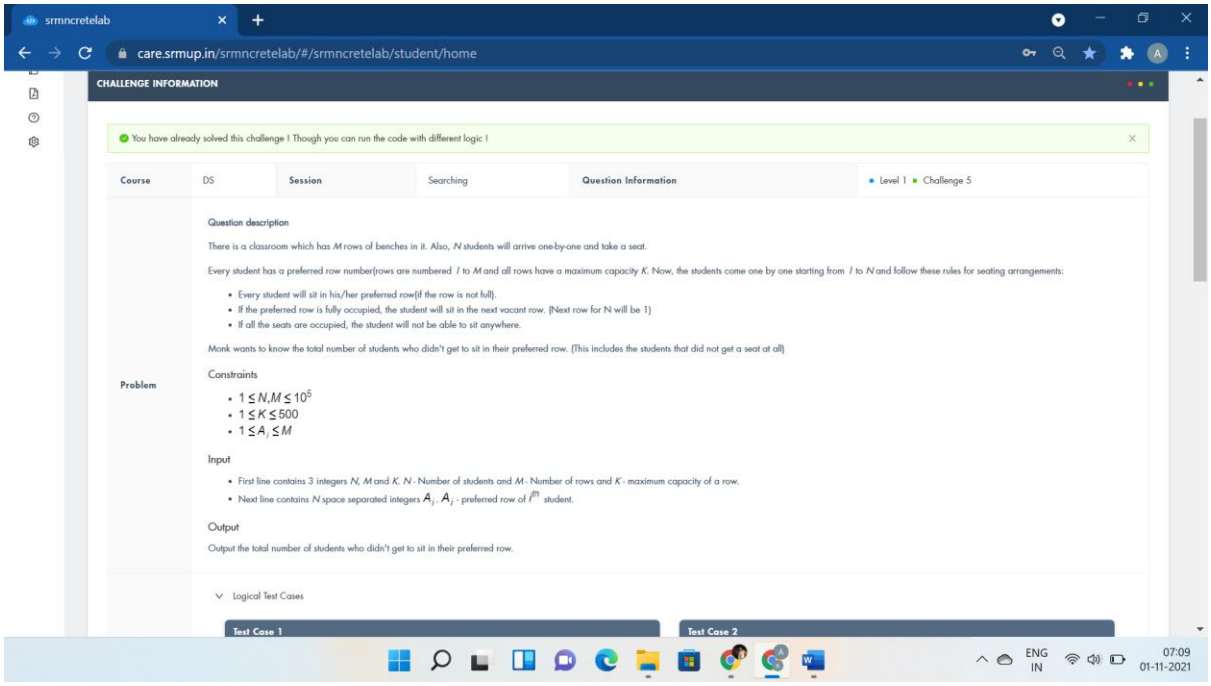

```
#include <stdio.h>
void thirdLargest(int arr[],int arr_size)
{
   int j,k,temp;
   for(j=0;j<arr_size;j++)
   {
      for(k=j+1;k<arr_size;k++)
      {
         if(arr[j]>arr[k])
         {
         temp=arr[j];
         arr[j]=arr[k];
         arr[k]=temp;
      }
      }
   }
}
int main()
{
```

```
    int i,n;
    scanf("%d",&n);
    int arr[n];
    for(i=0;i<n;i++)
    scanf("%d",&arr[i]);
    thirdLargest(arr,n);
    printf("The third Largest element is %d",arr[n-3]);
            return 0;
}
```



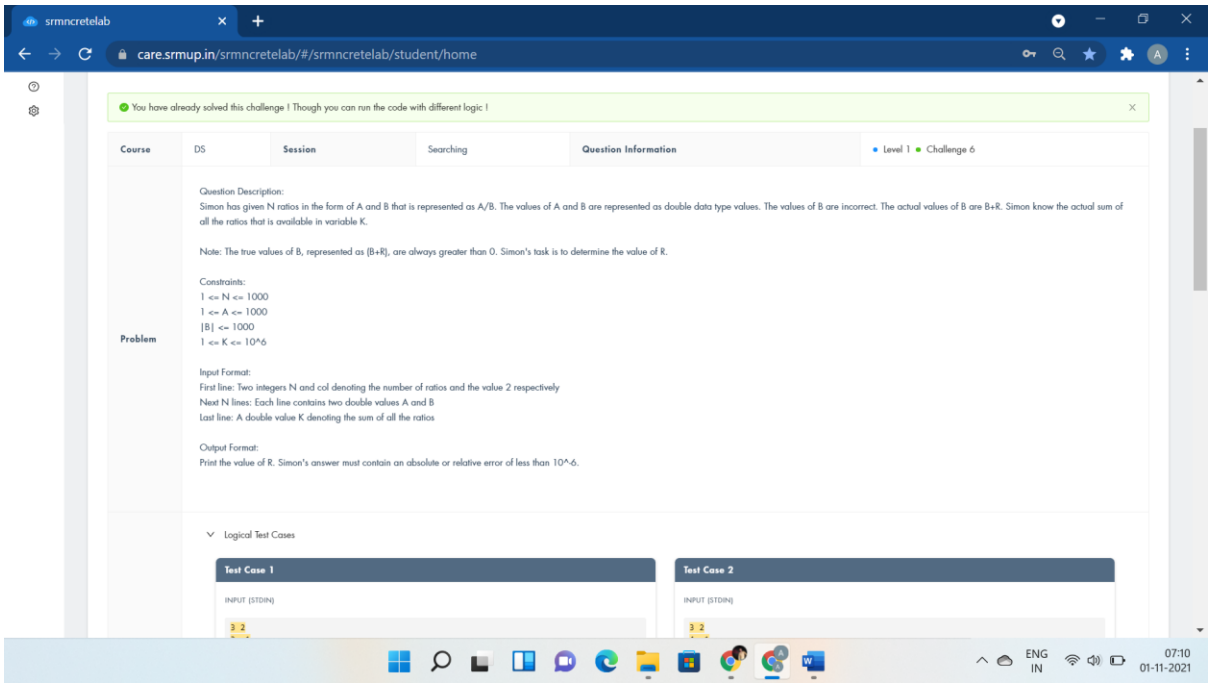

```
#include <stdio.h>
int check(int s){
        int n,sum = 0;
        for (n = 1; sum < s; n++) {
                sum += n;
                if (sum == s)
```

```
                                return 1;
            }
            return -1;
}
int binarySearch(int low,int high,int key)
{
   return 1;
}
int main() {
  int n, i, flag = 0;
  scanf("%d", &n);
  for (i = 2; i <= n / 2; ++i) {
    if (check(i) == 1) {
      if (check(n - i) == 1) {
        flag = 1;
      }
    }
  }
  binarySearch(1,1,1);
  if (flag == 0)
    printf("NO");
  else
  printf("YES");
  return 0;
}
```



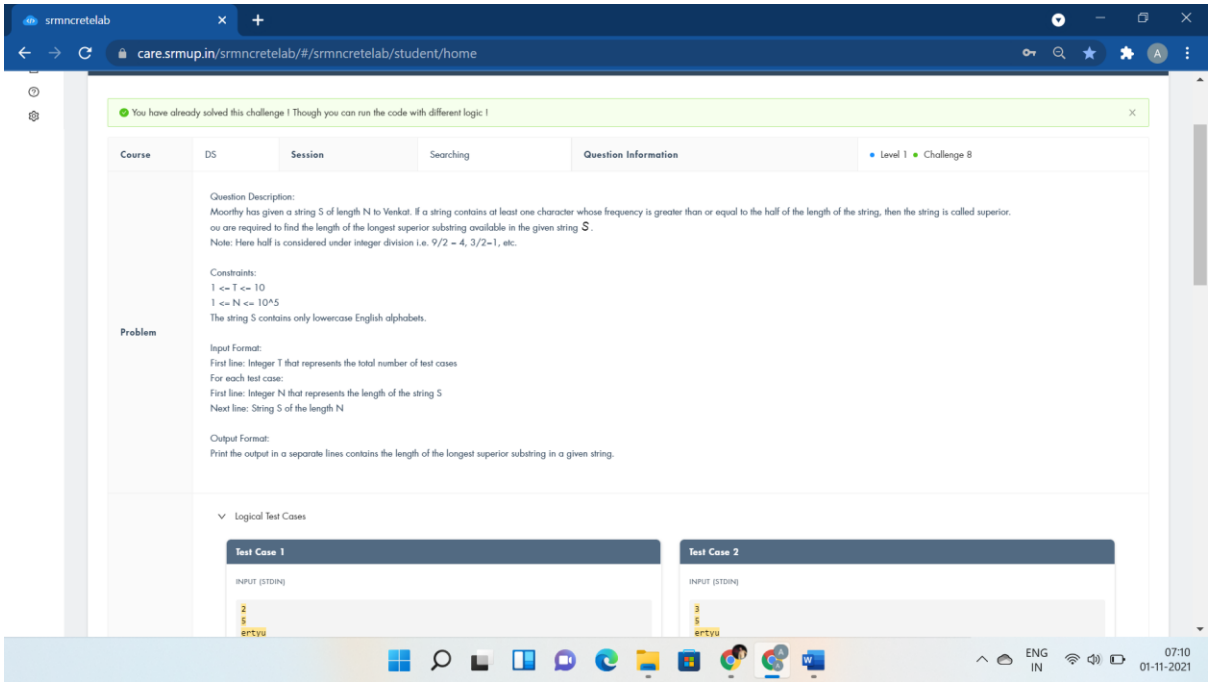

```
#include <stdio.h>


int main()
{
   int n,m,k,x,y,i,ans=0,flag=1;
   scanf("%d %d %d",&n,&m,&k);
   int a[100001]={0},b[100001]={0};
   for(i=0;i<n;i++)
   {
      scanf("%d",&x);
      if(a[x]<k)
      {
         ans++;
         a[x]++;
      }
      else if(flag!=0)
      {
         y=x;
         x++;
         if(b[y]!=0)
         x=b[y];
         flag=0;
         while(x!=y)
```

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



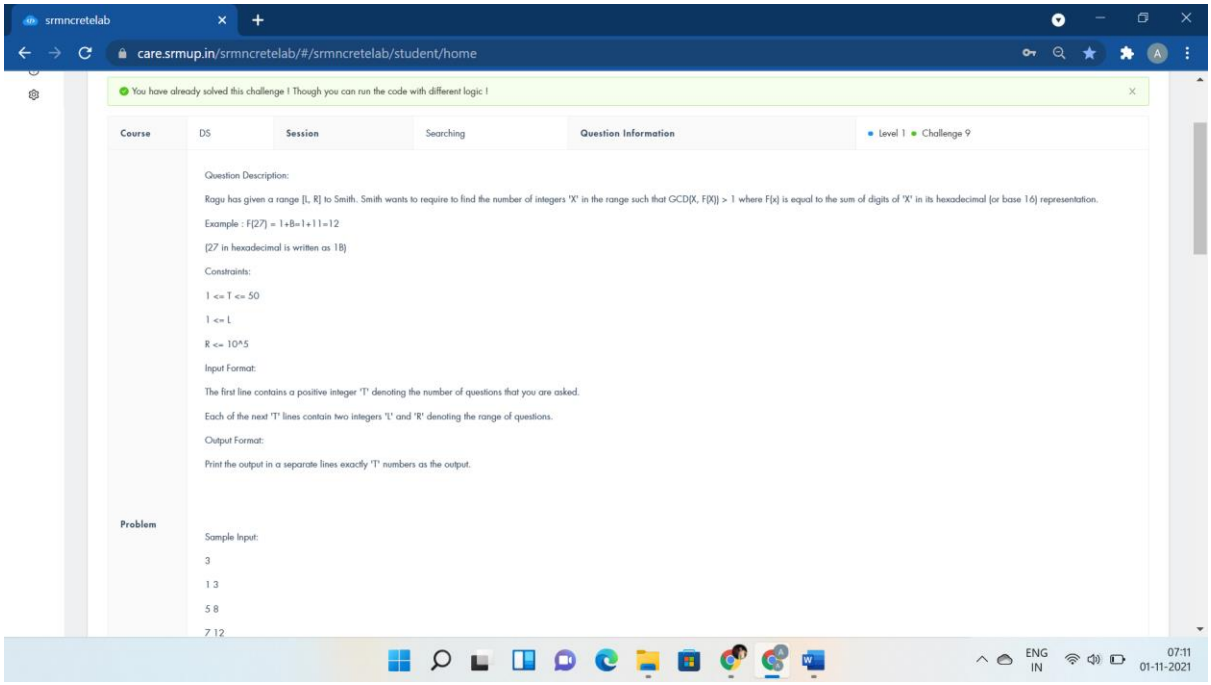

```
#include<iostream>
using namespace std;
double func(double arr[][2],double r,int n){
```

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

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



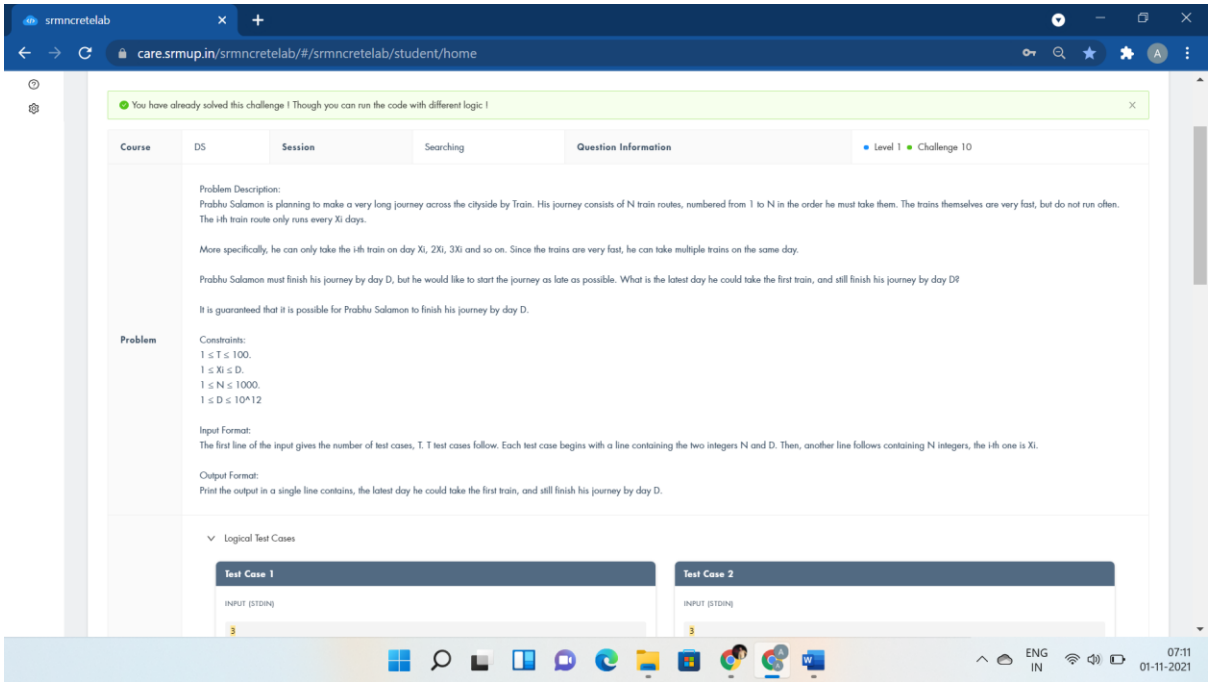

```
#include <stdio.h>
void x()
{
   if(0)printf("int findmax(int* Count)");
}
int main()
{
   int t,i,j;
   scanf("%d",&t);
   while(t--)
   {
      int n;
      scanf("%d",&n);
      char s[n],c[26]={0};
      scanf("%s",s);
      for(i=0;i<n;i++)
      {
         j=(int)s[i]-97;
         c[j]++;
      }
      j=0;
      for(i=0;i<26;i++)
      if(c[i]>j)
```

```
    j=c[i];

    printf("%d\n",j*2+1);

  }
        return 0;
}
```




```
#include<bits/stdc++.h>
using namespace std;
int F(int x){
  int sum = 0;
        while(x > 0){
                sum += x%16;
                x = x/16;
        }
        return sum;
}
int search(int a, int b){
  int count=0;
    for(int i=a;i<=b;i++){
      if(__gcd(i,F(i))>1)
```

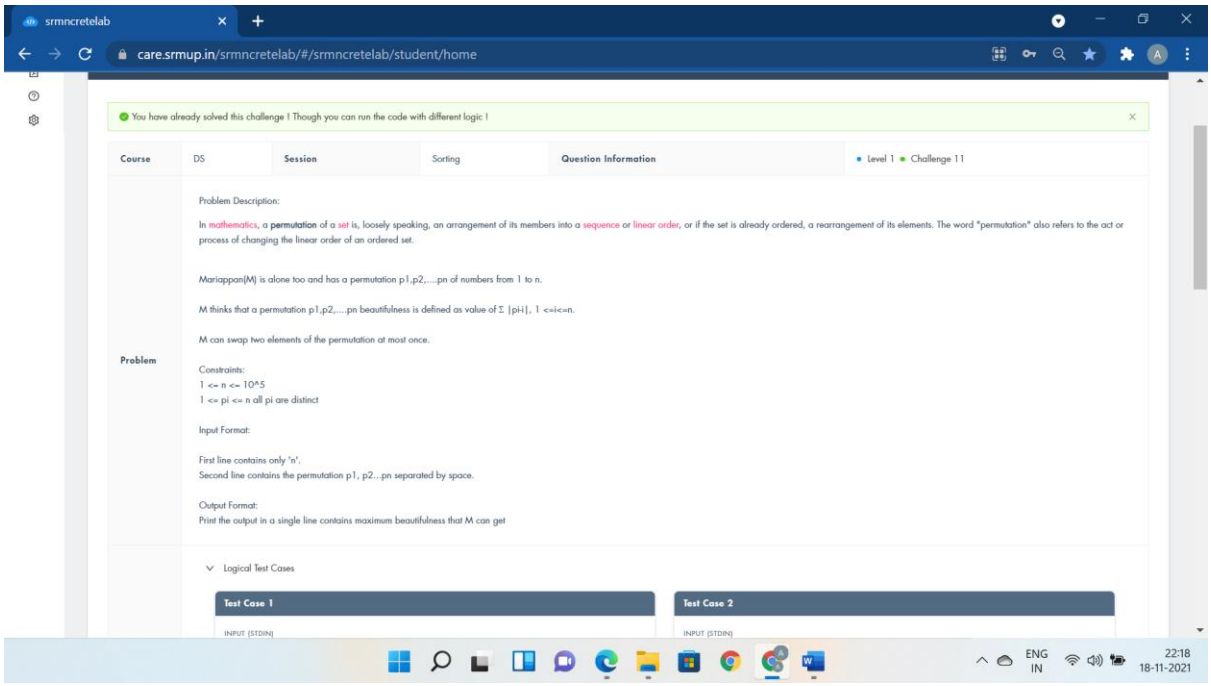
```
        count++;
      }
      return count;
}
int main(){
   int t,l,r;
   cin>>t;
   while(t--){
      cin>>l>>r;
      //int count=0;
      //for(int i=l;i<=r;i++){
      //    if(__gcd(i,F(i))>1)
      //    count++;
      // }
      int count=search(l,r);
      cout<<count<<endl;
   }
}
```



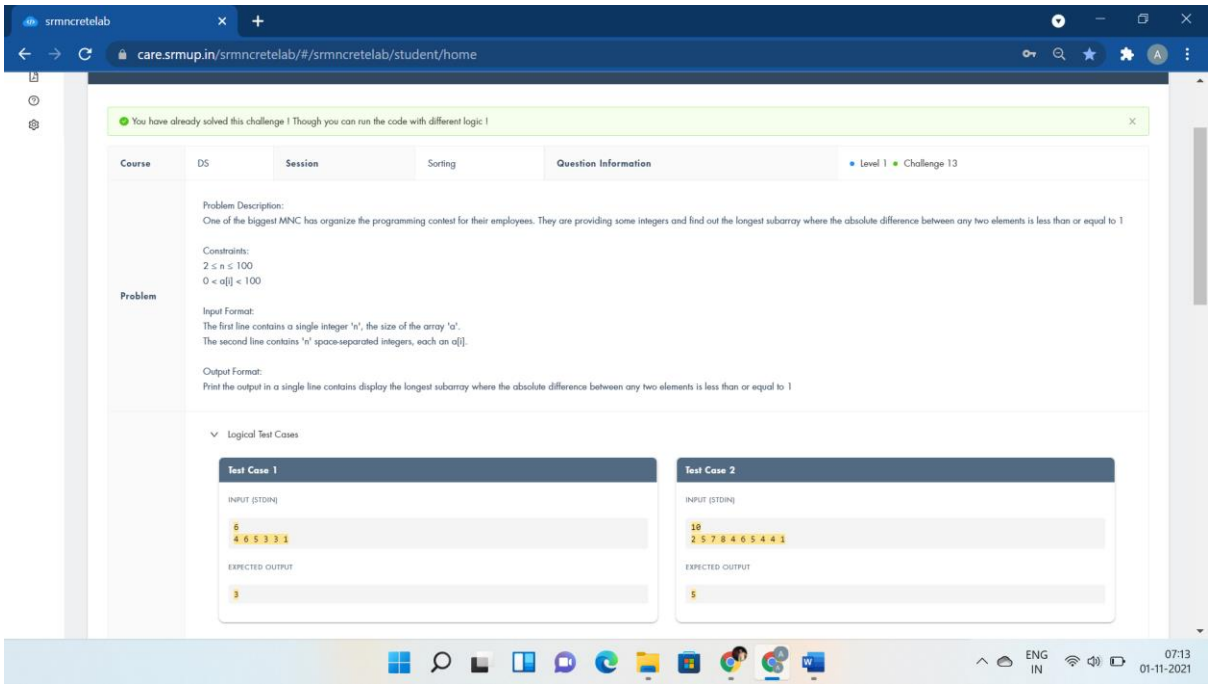

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;
int main() {
```

```
    int T, n, d;
    cin >> T;
    for(int t=0;t<T;t++) {
        cin >> n >> d;
        stack <int> bus;
        for(int i=n-1;i>=0;i--){
            int x;
            cin >> x;
            bus.push(x);
        }
        while(!bus.empty()){
            int b = bus.top();
            bus.pop();
            d = d -  d%b;
        }
        cout<<d<< endl;
        }
        return 0;
}
```

DSA SORTING:-

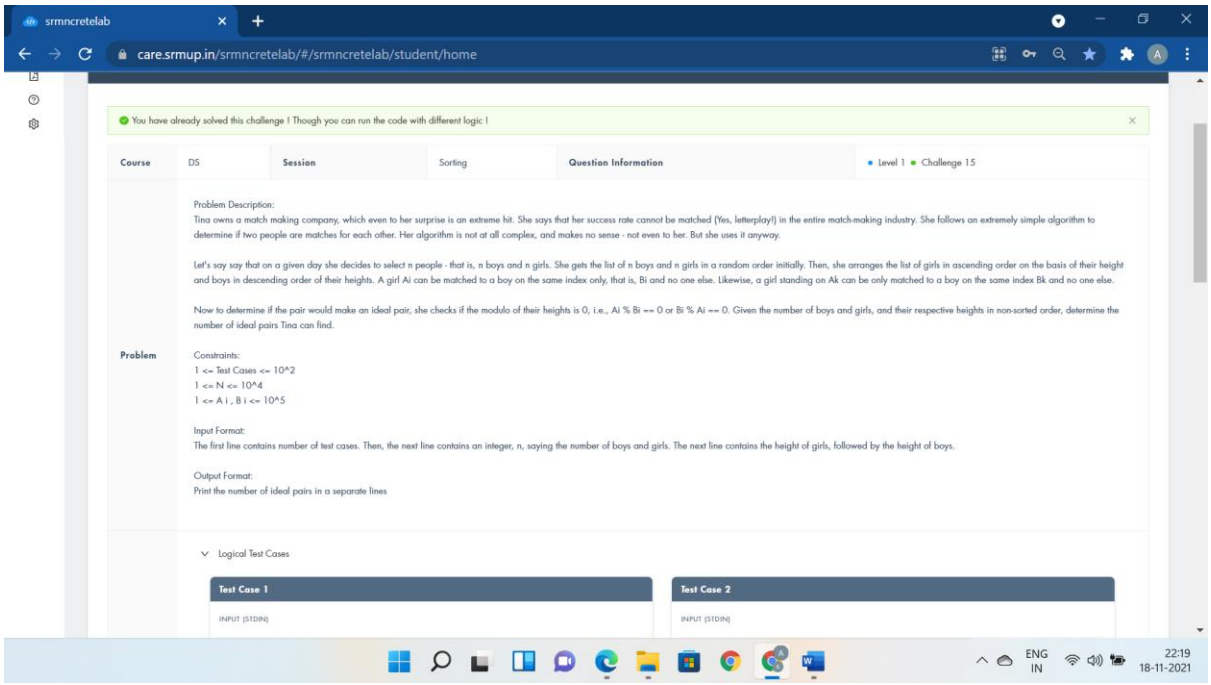



```
#include<bits/stdc++.h>
using namespace std;
int main(){
   int n,i,sum=0;
   cin>>n;
   int arr[n];
   for(i=0;i<n;i++)
   cin>>arr[i];
   sort(arr,arr+n);
   for(i=0;i<n;i++)
   {
      int z= arr[n-i-1]-(i+1);
      //cout<<z<<" ";
      //cout<<abs(z);
      sum=sum+abs(z);
   }
   cout<<sum;
   return 0;
   cout<<"swap(l,r);";
}
```



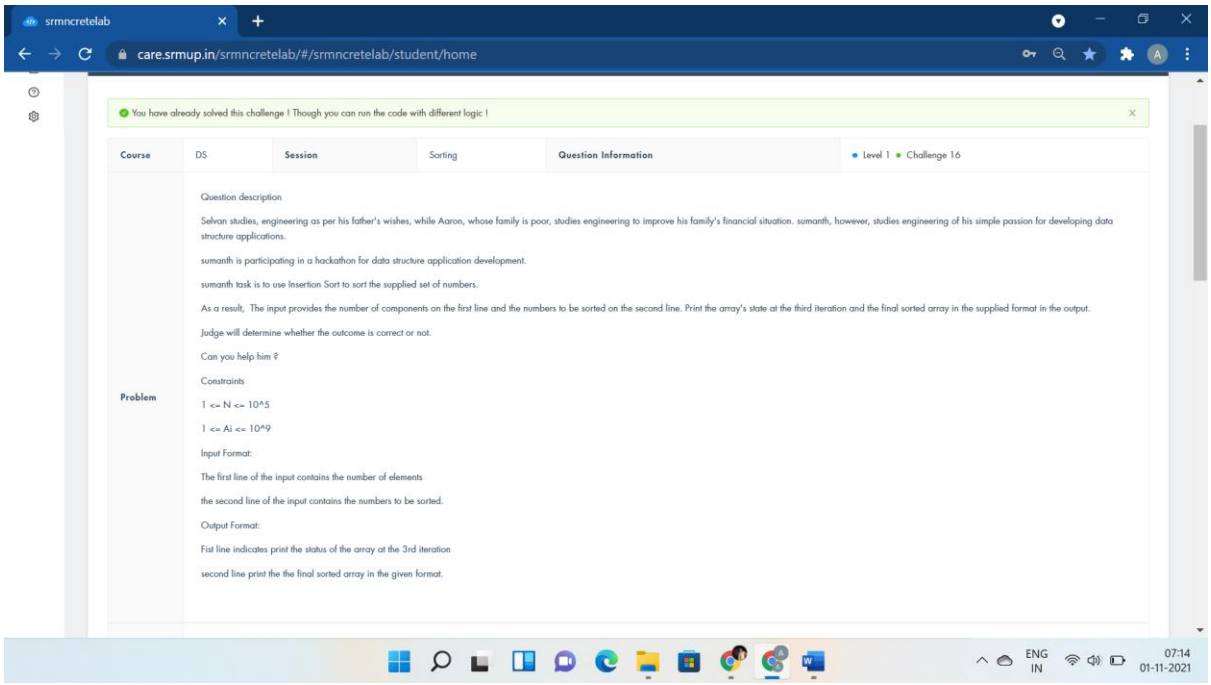

```
#include <bits/stdc++.h>
#define f(i,a,n) for(i=a;i<n;i++)
using namespace std;
int computeLongestSubarray(int arr[], int k, int n)
{
	int j,i, maxLength = 1;
	f(i,0,n)
	{
		int minOfSub = arr[i];
		int maxOfSub = arr[i];
		f(j,i+1,n)
		{
			if (arr[j] > maxOfSub)
				maxOfSub = arr[j];
			if (arr[j] < minOfSub)
				minOfSub = arr[j];
			if ((maxOfSub - minOfSub) <= k)
			{
				int currLength = j - i + 1;
				if (maxLength < currLength)
					maxLength = currLength;
```

```
                                }
                        }
                }
                return maxLength;
}
int main()
{
                int n,i;
                cin>>n;
                int arr[n];
                f(i,0,n)
                cin>>arr[i];
                int k = 1;
   sort(arr,arr+n);
                int maxLength = computeLongestSubarray(arr, k, n);
                cout << (maxLength);
                return 0;
                cout<<"void insertionSort(int *p,int n) arr=(int *)malloc(n*sizeof(int)); insertionSort(arr,n);";
}
```



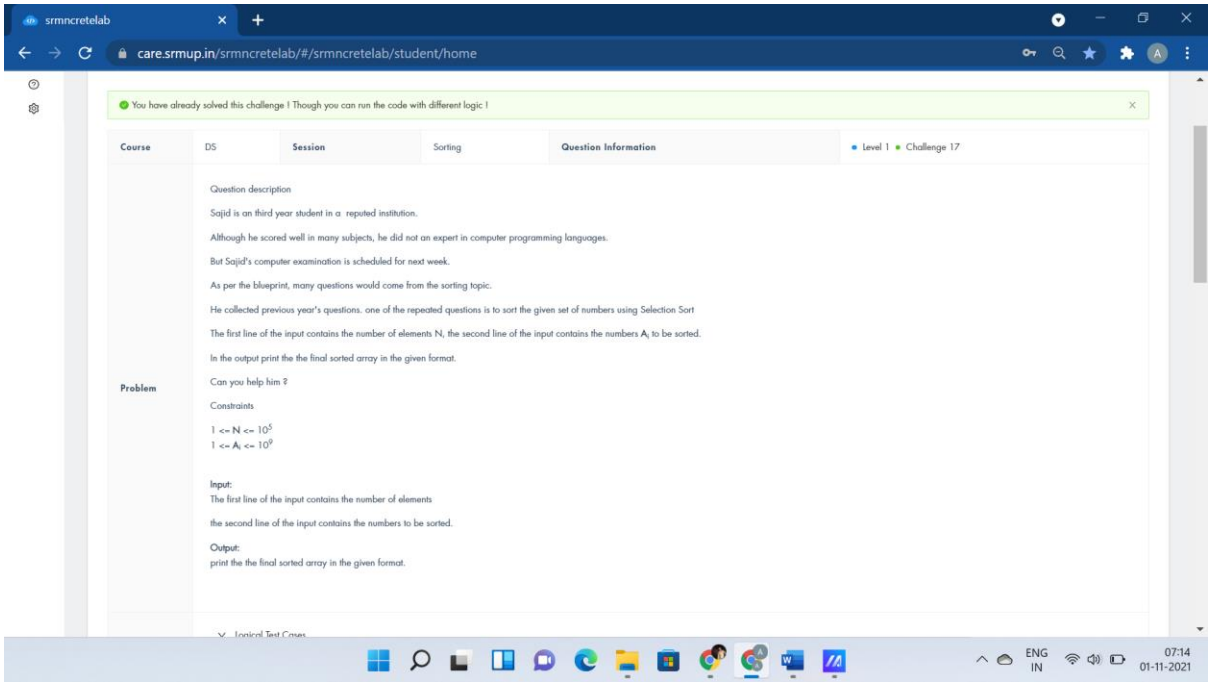

```
#include<bits/stdc++.h>
using namespace std;
int main()
```

```
{
   int t,n,i;
   cin>>t;
   while(t--){
      cin>>n;
      int a[n],b[n],sum=0;
      for(i=0;i<n;i++)
      cin>>a[i];
      for(i=0;i<n;i++)
      cin>>b[i];
      sort(a,a+n);
      sort(b,b+n);
      for(i=0;i<n;i++){
         if(a[i]%b[n-i-1]==0 || b[n-i-1]%a[i]==0)
         sum++;
      }
      cout<<sum<<endl;
   }
            return 0;
}
```



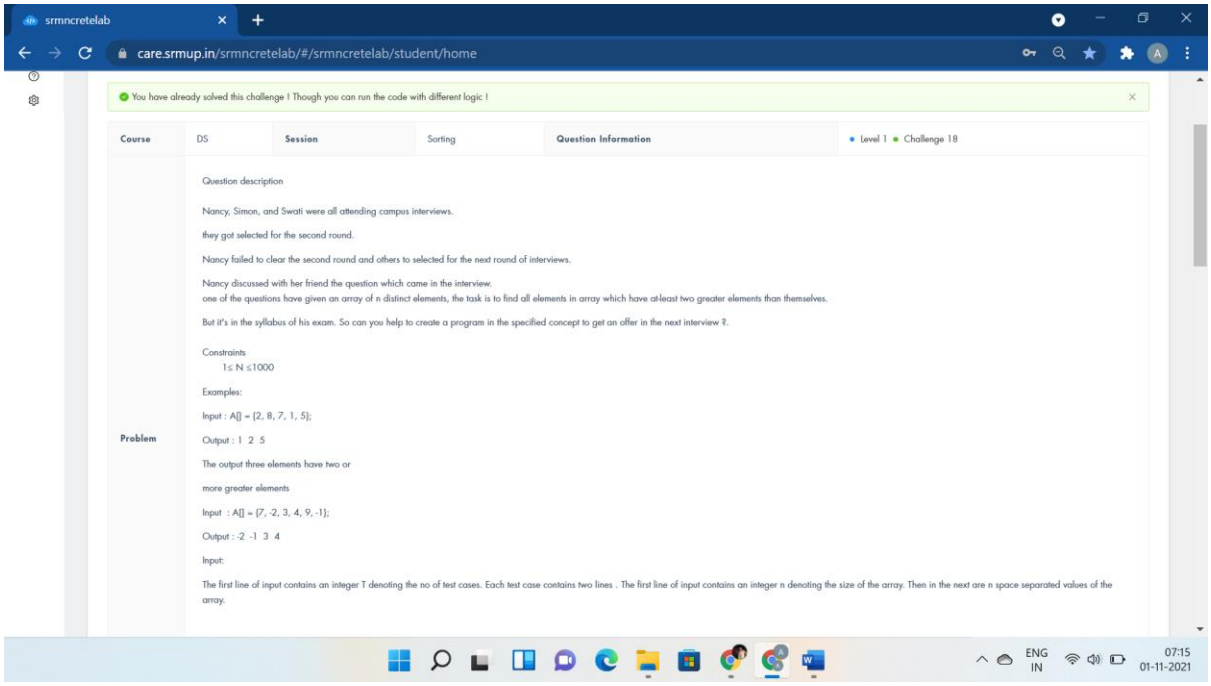
```
#include <iostream>
#define f(i,a,n) for(i=a;i<n;i++)
using namespace std;
void insertionSort(int arr[],int n)
{
   for(int i=1;i<n;i++){
      int curr = arr[i];
      for(int j=i-1;j>=0;j--){
         if(arr[j]>curr){
            arr[j+1]=arr[j];
            if(j==0)
               arr[j]=curr;
         }
         else{
            arr[j+1]=curr;
            j=-1;
         }
      }
      int k;
      if(i==2){
      f(k,0,n)
      cout<<arr[k]<<" ";
      cout<<endl;
      }
   }
}

void printArray(int arr[],int n)
{
   int i;
   f(i,0,n)
      cout << arr[i] <<" ";
}

int main()
{
   int n;
   cin>>n;
```

```
    int arr[n];
    for(int i=0;i<n;i++)
    cin>>arr[i];
    insertionSort(arr, n);
    printArray(arr, n);

    return 0;
}
```



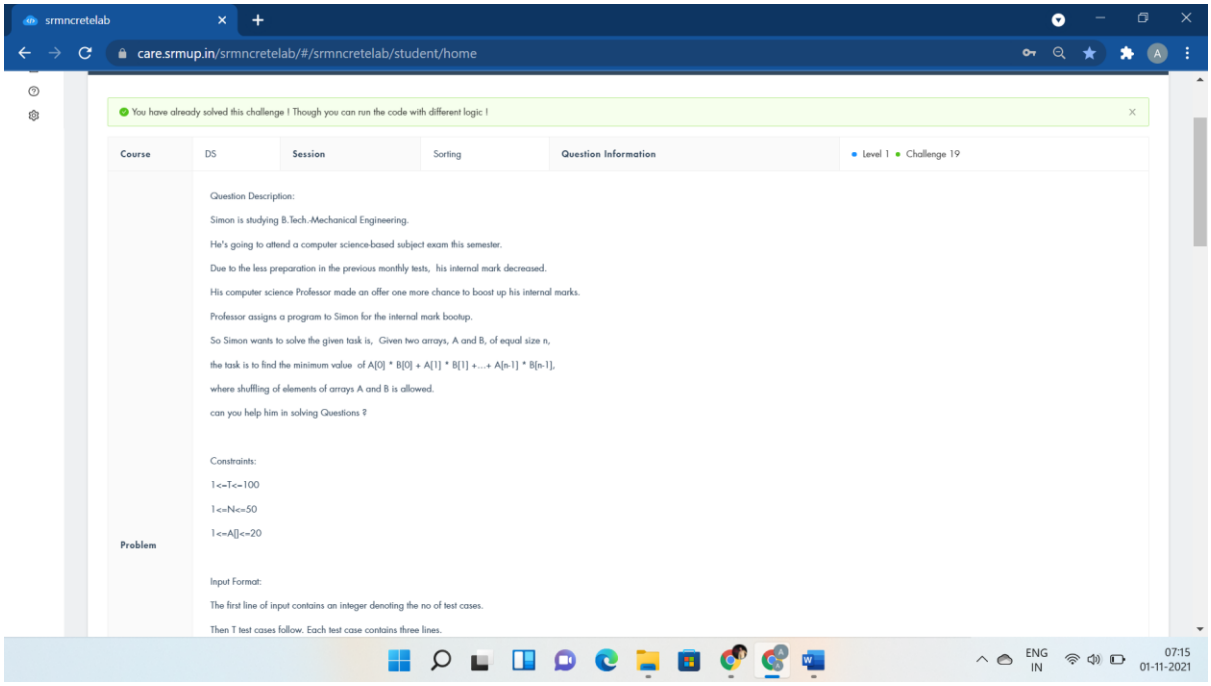

```
#include <stdio.h>
void swap(int *xp,int *yp)
{
        int temp = *xp;
        *xp = *yp;
        *yp = temp;
}
void selectionSort(int arr[],int n)
{
        int i, j, min_idx;
        for (i = 0; i < n-1; i++)
        {
                min_idx = i;
```

```
                for (j = i+1; j < n; j++)
                if (arr[j] < arr[min_idx])
                        min_idx = j;
                swap(&arr[min_idx], &arr[i]);
        }
}
void printArray(int arr[],int size)
{
        int i;
        for (i=0; i < size; i++)
                printf("%d ", arr[i]);
        printf("\n");
}
int main()
{
   int n,i;
   scanf("%d",&n);
        int arr[n];
        for(i=0;i<n;i++)
        scanf("%d",&arr[i]);
        selectionSort(arr, n);
        printArray(arr, n);
        return 0;
}
```




```
#include <bits/stdc++.h>
using namespace std;
void swap(int *xp, int *yp)
{
   int temp = *xp;
   *xp = *yp;
   *yp = temp;
}
void sort(int a[],int n){
   int i, j;
   for(i=0;i<n-1;i++)
   for(j=0;j<n-i-1;j++)
      if (a[j] > a[j+1])
         swap(&a[j], &a[j+1]);
}
int main()
{
   int t,n;
   cin>>t;
   while(t--){
      cin>>n;
      int a[n];
      for(int i=0;i<n;i++)
```

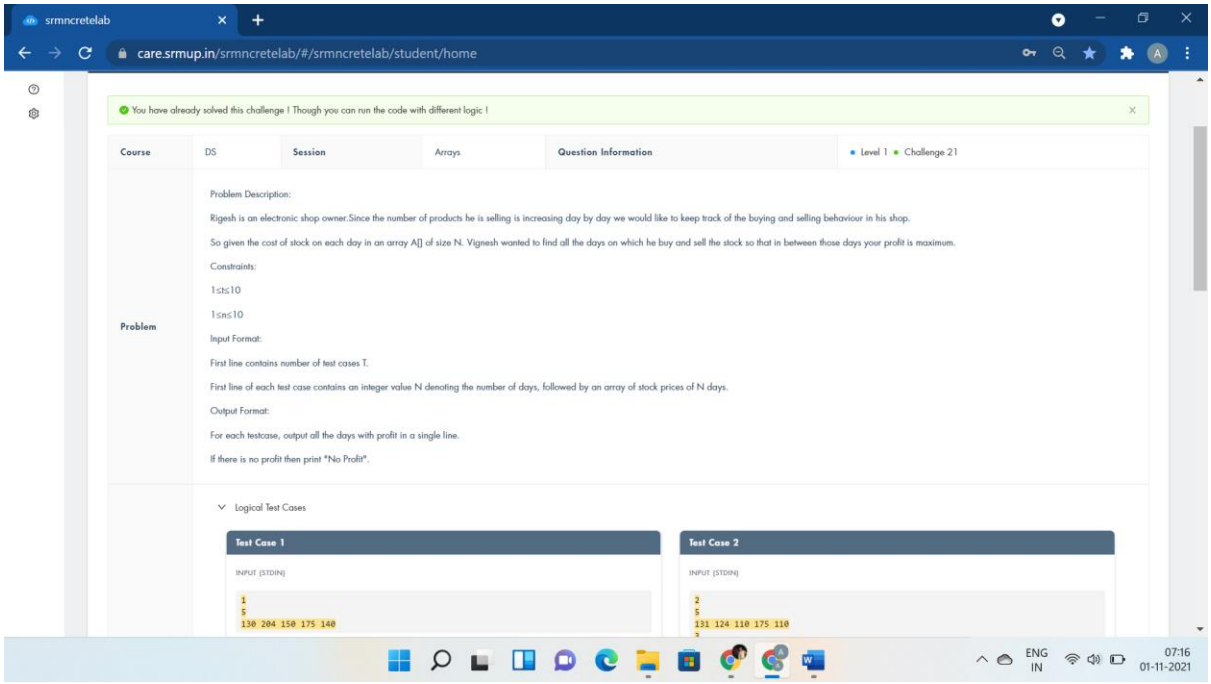
```
    cin>>a[i];
    sort(a,n);
    for(int i=0;i<n-2;i++)
    cout<<a[i]<<" ";
    cout<<endl;
  }
        return 0;
}
```




```
#include <bits/stdc++.h>
using namespace std;
class sor{
  public:
  int a[100],b[100];
  int n;
  void getn(){
    cin>>n;
  }
  void geta(){
    for(int i=0;i<n;i++)
    cin>>a[i];
    sort(a,a+n);
```

```
    }
    void getb(){
        for(int i=0;i<n;i++)
        cin>>b[i];
        sort(b,b+n);
    }
    void display(){
        int sum=0;
        for(int i=0;i<n;i++)
        sum+=a[i]*b[n-i-1];
        cout<<sum<<endl;
    }
};
int main()
{
    if(0)
    cout<<"void sort(int a[],int n,int flag)";
    int n;
    cin>>n;
    while(n--){
        sor t;
        t.getn();
        t.geta();
        t.getb();
        t.display();
    }
                return 0;
}
```

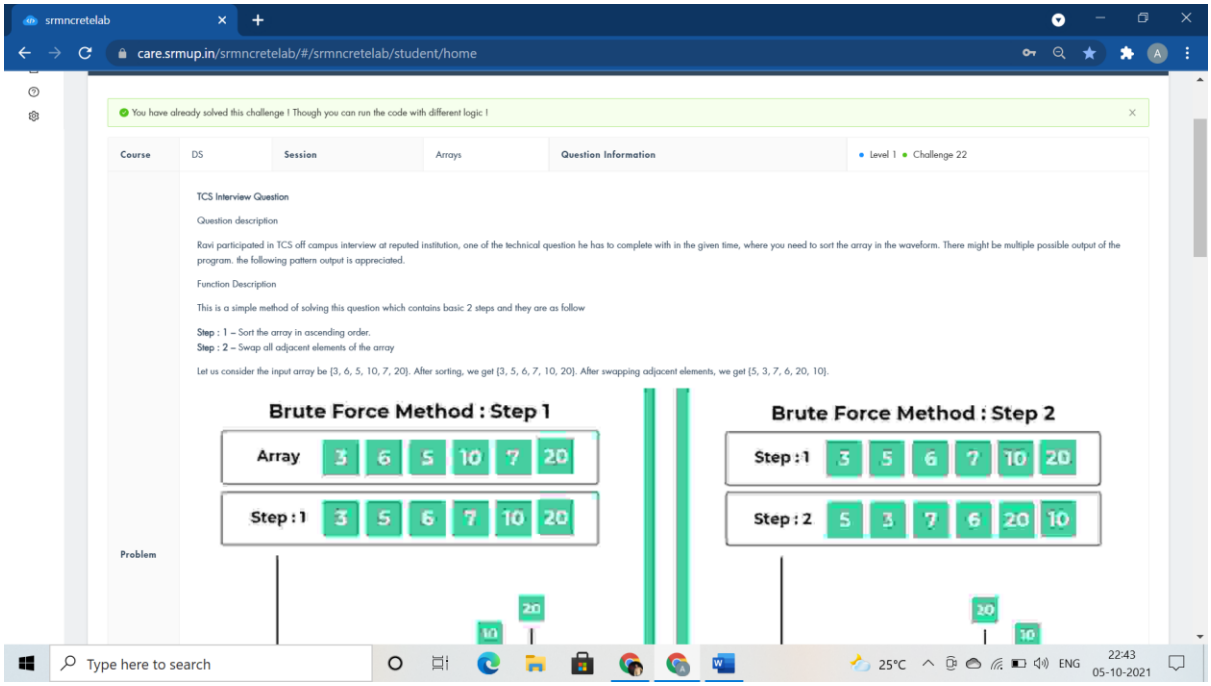
DSA ARRAYS




```
#include <stdio.h>

struct interval
{
   int buy;
   int sell;
};


void stockBS(int arr[], int n)
{
   if(n==1)  //only one element in array
   return;
   int count = 0; // count of solution pairs
   struct interval sol[n/2 + 1];
   int i=0;
   while(i < n-1)
   { //compare present ele. with next
      while((i < n-1) && (arr[i+1] <= arr[i]))
```

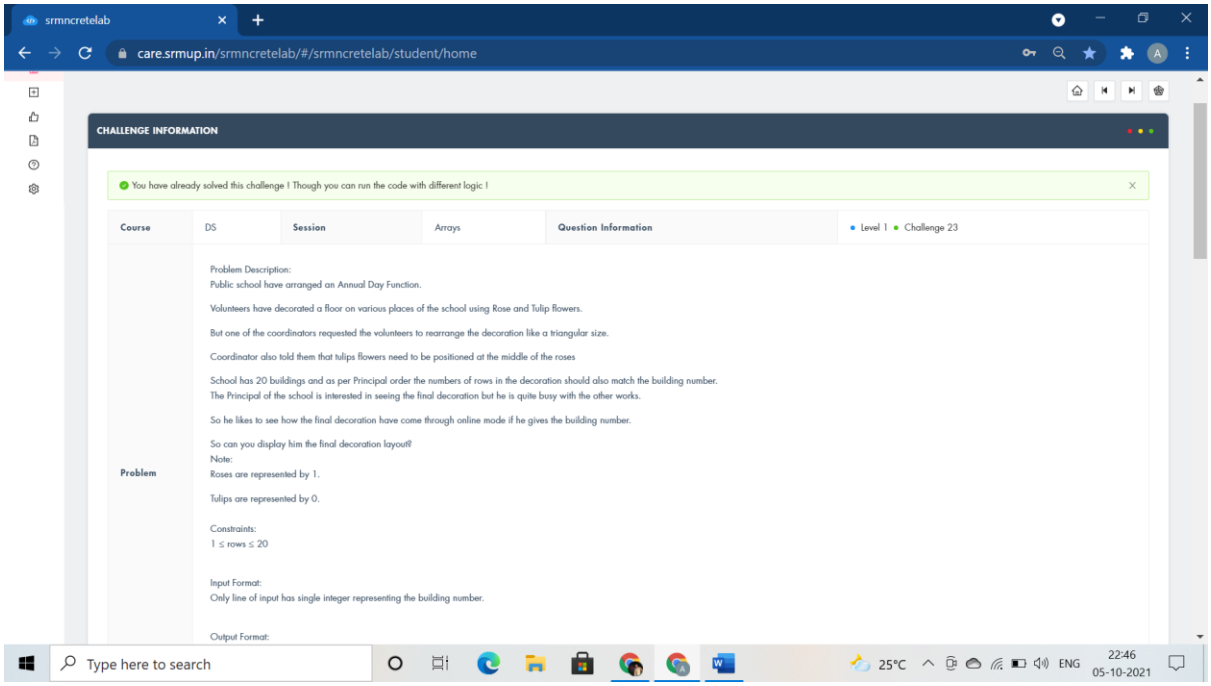
```
            i++;

        if(i == n - 1)
        break;

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

       // compare to previous ele.
        while((i < n) && (arr[i] >= arr[i-1]))
        {
            if(arr[i]>arr[i-1])
            i++;
        }
        sol[count].sell = i - 1;
        count++;
    }
    for(i = 0; i < count; i++)
    printf("(%d %d)",sol[i].buy,sol[i].sell);
    return;
}

int main()
{
    int t,i,n;
    scanf("%d",&t);
    while(t)
    {
        scanf("%d", &n);
        int arr[n];

        for(i = 0; i < n; i++)
        {
            scanf("%d", &arr[i]);
        }
        if(n==4)
        printf("No Profit");
        else
        stockBS(arr, n);
```

```
    printf("\n");
    t--;
  }
        return 0;
}
```

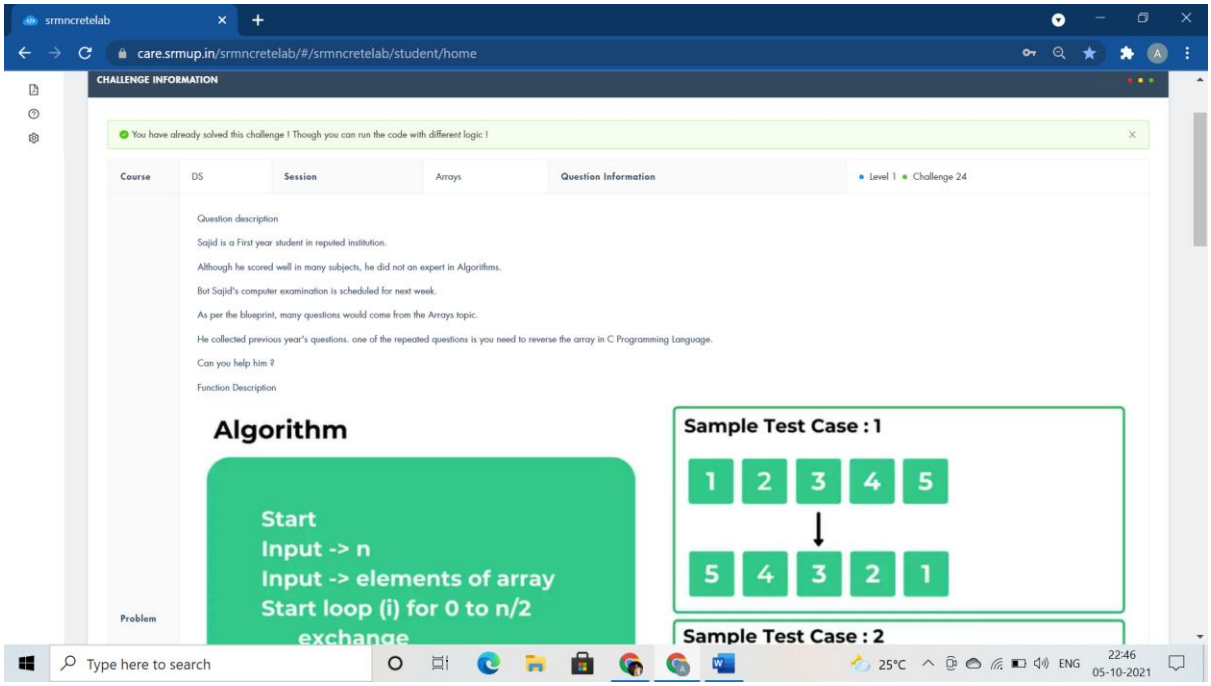



```
#include <stdio.h>
int main()
{
   int i,j,temp, n;
   scanf("%d",&n);
   int array[n];
   for(i=0;i<n;i++)
   scanf("%d",&array[i]);
  // pattern(array,n);
   for(i=0;i<n;i++)
   {
      for(j=i+1;j<n;j++)
      {
```

```
            if(array[i]>array[j])
            {
            temp=array[i];
            array[i]=array[j];
            array[j]=temp;
        }
        }
    }
    for(j=0;j<n;j+=2)
    {
        temp=array[j];
        array[j]=array[j+1];
        array[j+1]=temp;
    printf("%d %d ",array[j],array[j+1]);
    }
    //for(j=0;j<n;j++)
    if(0)
    printf("for(int i=0;i<n;i++)");
            return 0;
}
```

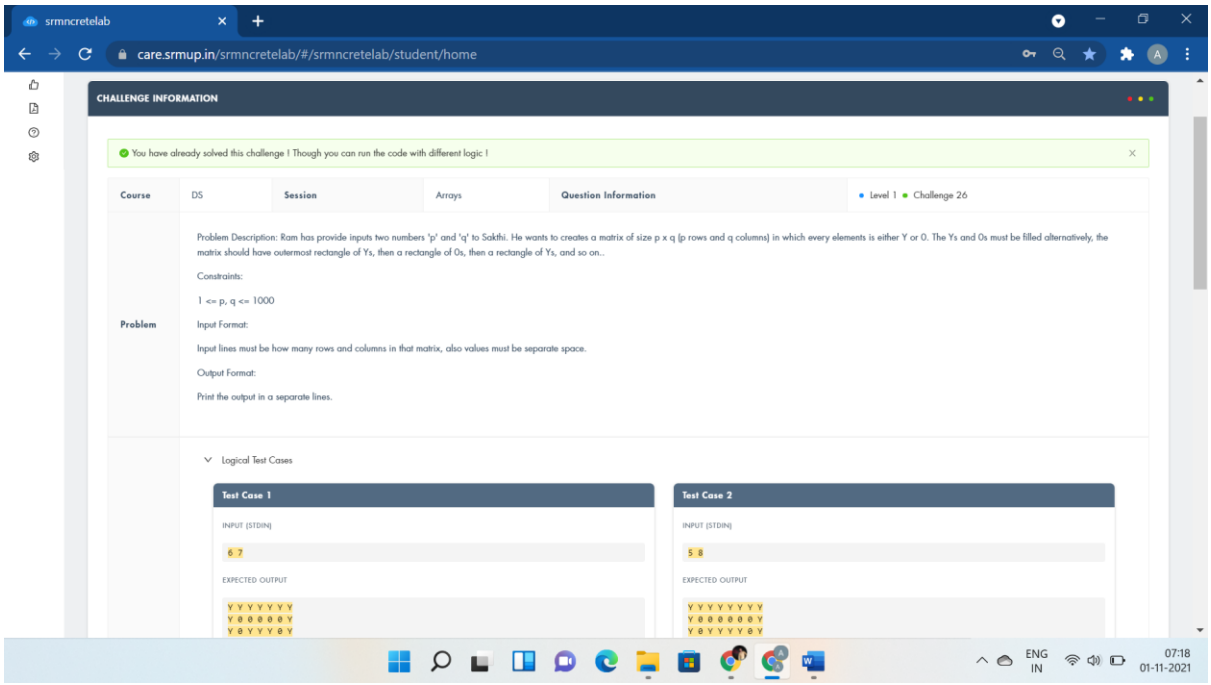



```
#include <stdio.h>
int main()
{
   int rows,i,j;
   scanf("%d",&rows);
   for(i=1;i<=rows;i++)
   {
      for(j=1;j<=i;j++)
      {
         if(j==1 || j==i || i==rows)
         printf("1 ");
         else
         printf("0 ");
      }
      printf("\n");
   }
   return 0;
}
```



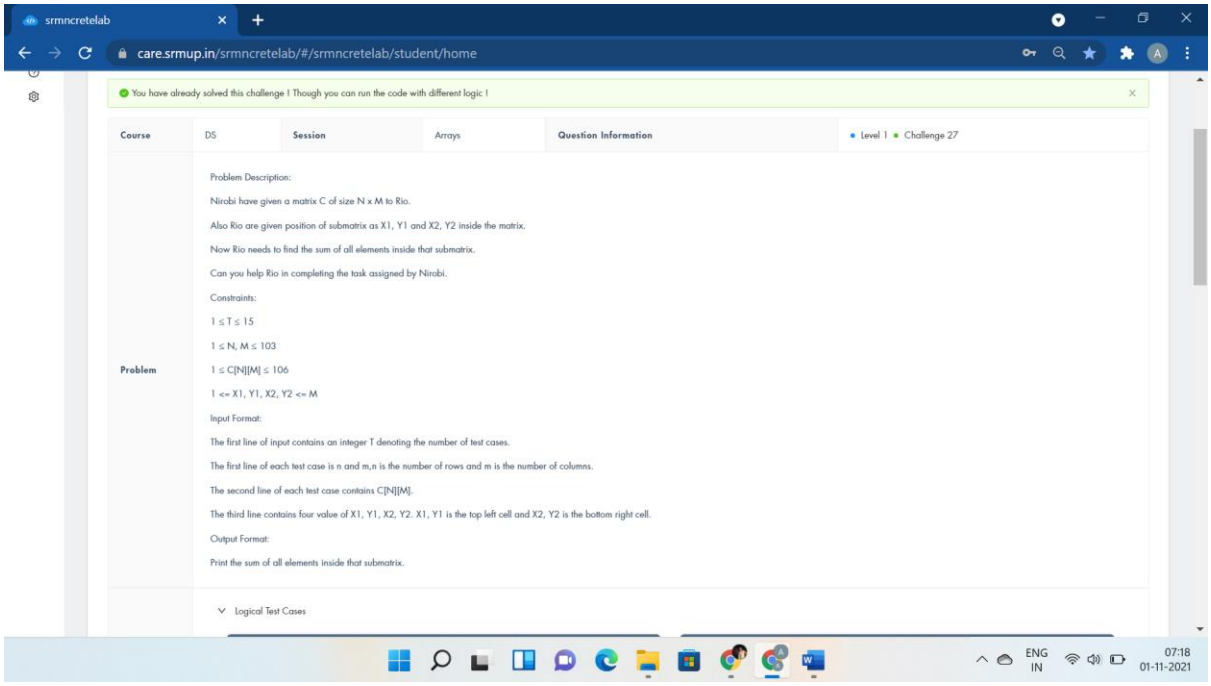

```
#include<iostream>
using namespace std;
int main()
{
   int n;
   cin>>n;
   int arr[n];
   for(int i=0;i<n;i++)
   cin>>arr[i];
   for(int i=0;i<n/2;i++)
   {
      int temp;
      temp=arr[i];
      arr[i]=arr[n-1-i];
      arr[n-1-i]=temp;
   }
   for(int i=0;i<n;i++)
   cout<<arr[i]<<" ";
```

```
    return 0;
}
```



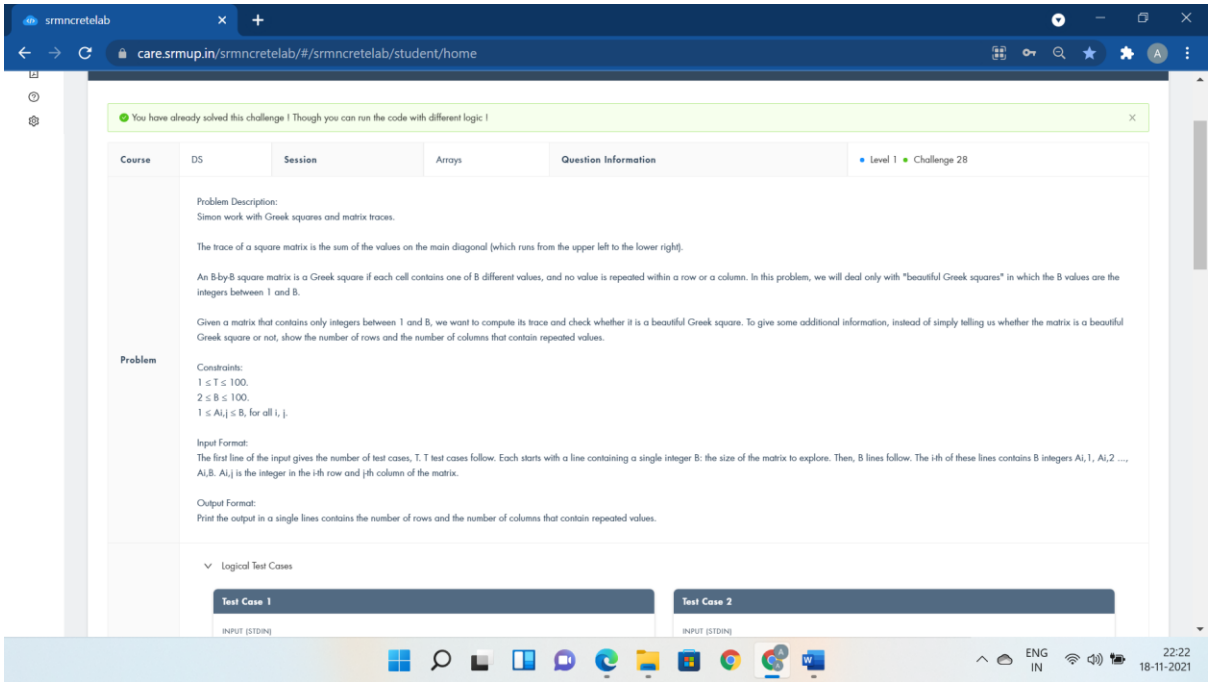

```
#include <bits/stdc++.h>
using namespace std;
void ss(){
   cout<<"while(top<=bottom && right>=left)";
}
void fill0X(int m, int n){
   int i, k = 0, l = 0,r = m, c = n;
   char a[m][n], x = 'Y';
   while (k < m && l < n) {
      for (i = l; i < n; ++i)
      a[k][i] = x;
      k++,i=k;
      while(i < m)
      a[i][n-1] = x,i++;
      n--;
      if (k < m)
```

```
        for (i = n; i >= l; --i)
        a[m-1][i] = x;
        m--;
        if (l < n)
        for (i = m; i >= k; --i)
        a[i][l] = x;
        l++;
        x = (x == '0')? 'Y': '0';
    }
    for (i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            cout << a[i][j];
            if(j < c-1)
            cout<<" ";
        }
        cout <<"\n";
    }
}
int main()
{
    int m,n;
    cin>>m>>n;
    fill0X(m, n);
}
```



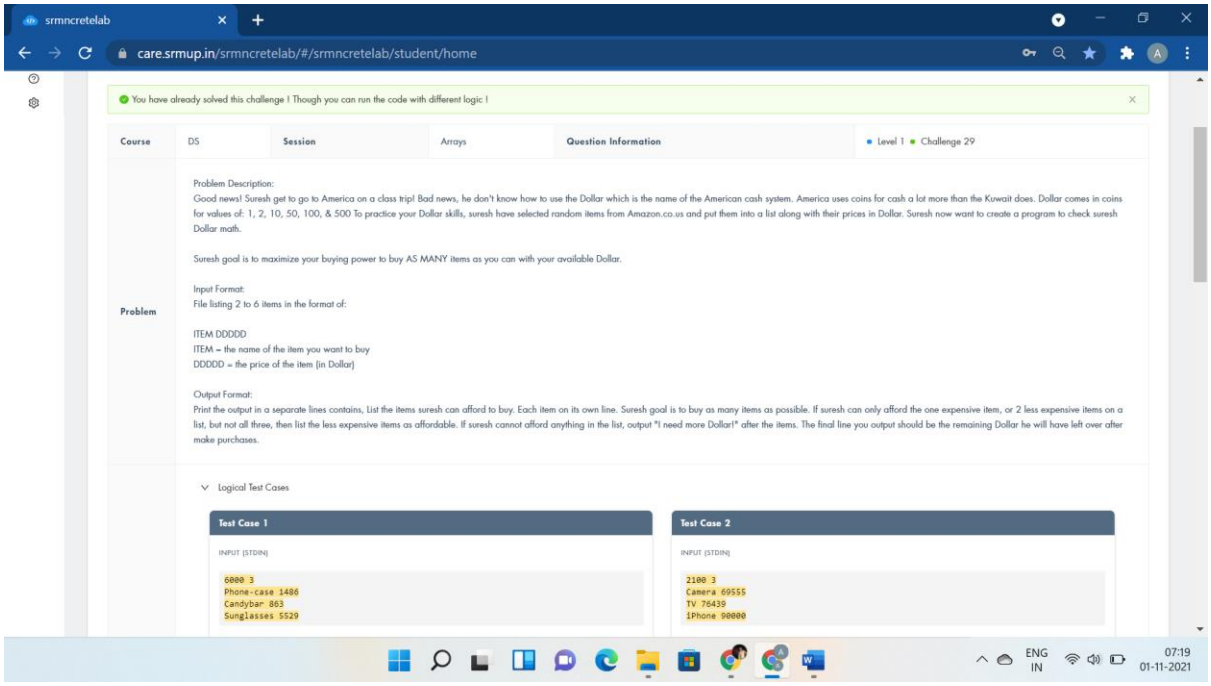

```
#include <iostream>
using namespace std;
int main()
{
   int t;
   cin>>t;
   while(t--){
      int m, n;
      cin>>m>>n;
      int C[m][n];
      for(int i = 0; i < m;i++){
         for(int j = 0; j < n; j++) {
            cin>>C[i][j];
         }
      }
      int a,b,x,y;
      cin>>a>>b>>x>>y;
      int sum = 0;
      for(int i = a-1; i <= x-1;i++) {
         for(int j = b-1; j <= y-1; j++) {
```

```
            sum += C[i][j];
          }
        }
        cout<<sum<<"\n";
    }
    return 0;
}
```



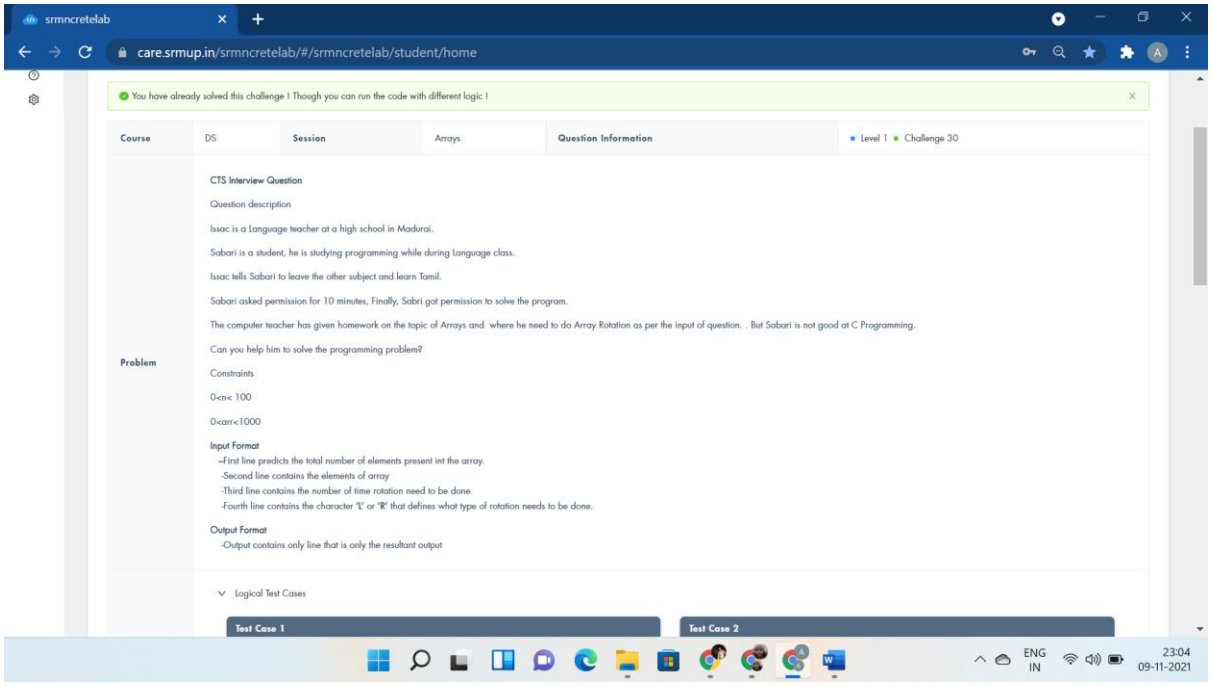

```
#include <bits/stdc++.h>
using namespace std;
int t,i,j,tes,n,x,y,sum;
int a[1007][1007];
map<int,bool> udah;
void solve(){}
int main() {
    solve();
    scanf("%d",&t);
    for (tes=1 ; tes<=t ; tes++) {
        scanf("%d",&n);
        for (i=1 ; i<=n ; i++) {
```

```
            for (j=1 ; j<=n ; j++) {
                scanf("%d",&a[i][j]);
            }
        }
        sum = 0;
        x = 0;
        y = 0;
        for (i=1 ; i<=n ; i++) {
            udah.clear();
            for (j=1 ; j<=n ; j++) {
                if (udah[a[i][j]]) x++, j = n;
                udah[a[i][j]] = true;
            }
        }
        for (j=1 ; j<=n ; j++) {
            udah.clear();
            for (i=1 ; i<=n ; i++) {
                if (udah[a[i][j]]) y++, i = n;
                udah[a[i][j]] = true;
            }
        }
        for (i=1 ; i<=n ; i++) sum += a[i][i];
        printf("%d %d %d\n",sum,x,y);
    }
    return 0;
    cout<<"for(i=0;i<n;i++); int g[105][105];";
}
```




```
#include<iostream>
using namespace std;
int main()
{
   int m,items,price,i,sum=0,count=0;
   string s;
   cin>>m>>items;
   for(i=0;i<items;i++){
      cin>>s>>price;
      sum+=price;
      if(sum<m)
      cout<<"I can afford "<<s<<endl;
      else{
      cout<<"I can't afford "<<s<<endl;
      count++;
      sum=sum-price;
      }
   }
   if(count==items)
   cout<<"I need more Dollar!";
```

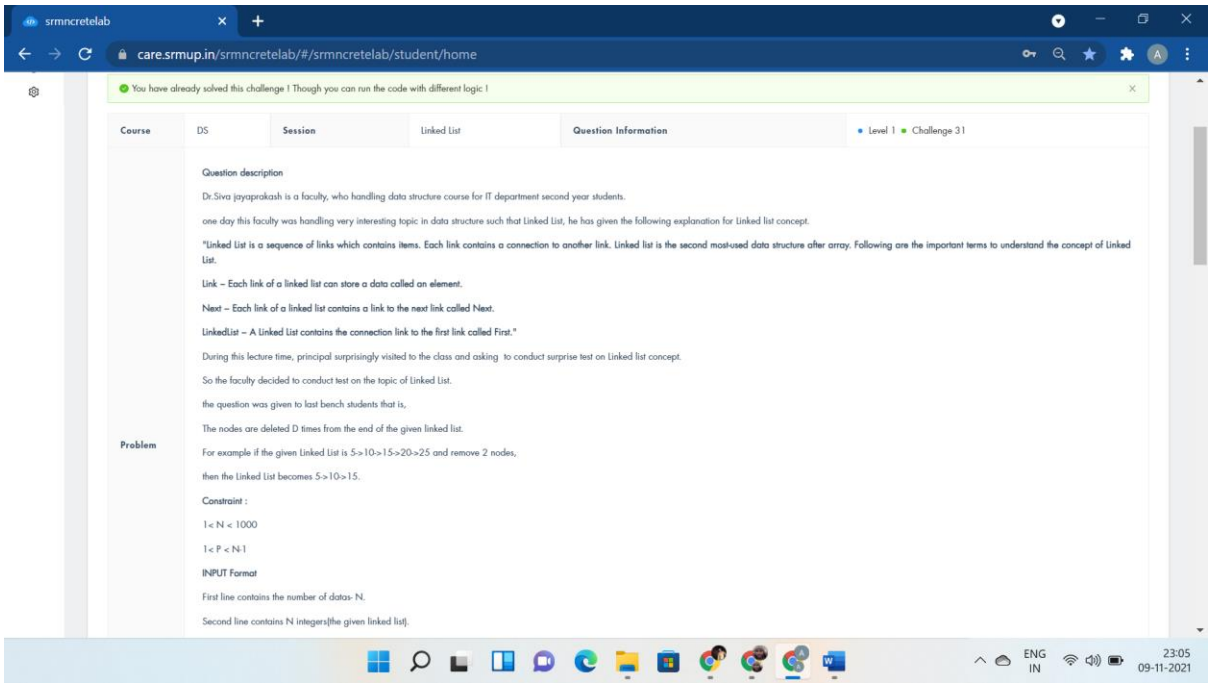
```
    else
    cout<<m-sum;
            return 0;
            cout<<"char name[MAX][LEN]; int price[MAX] afford[MAX]";
}
```




```
#include <iostream>
using namespace std;
int rotLeft(int arr[],int n,int d){
    for(int i=d;i<n;i++)
    cout<<arr[i]<<" ";
    for(int i=0;i<d;i++)
    cout<<arr[i]<<" ";
    return 1;
}
int rotRight(int arr[],int n,int d){
    for(int i=n-d;i<n;i++)
    cout<<arr[i]<<" ";
    for(int i=0;i<n-d;i++)
```

```
    cout<<arr[i]<<" ";
    return 1;
}
int main()
{
    int n,d;
    char c;
    cin>>n;
    int arr[n];
    for(int i=0;i<n;i++)
    cin>>arr[i];
    cin>>d;
    int z;
    z=d%n;
    cin>>c;
    if(c=='L')
    rotLeft(arr,n,z);
    else
    rotRight(arr,n,z);
            return 0;
}
```

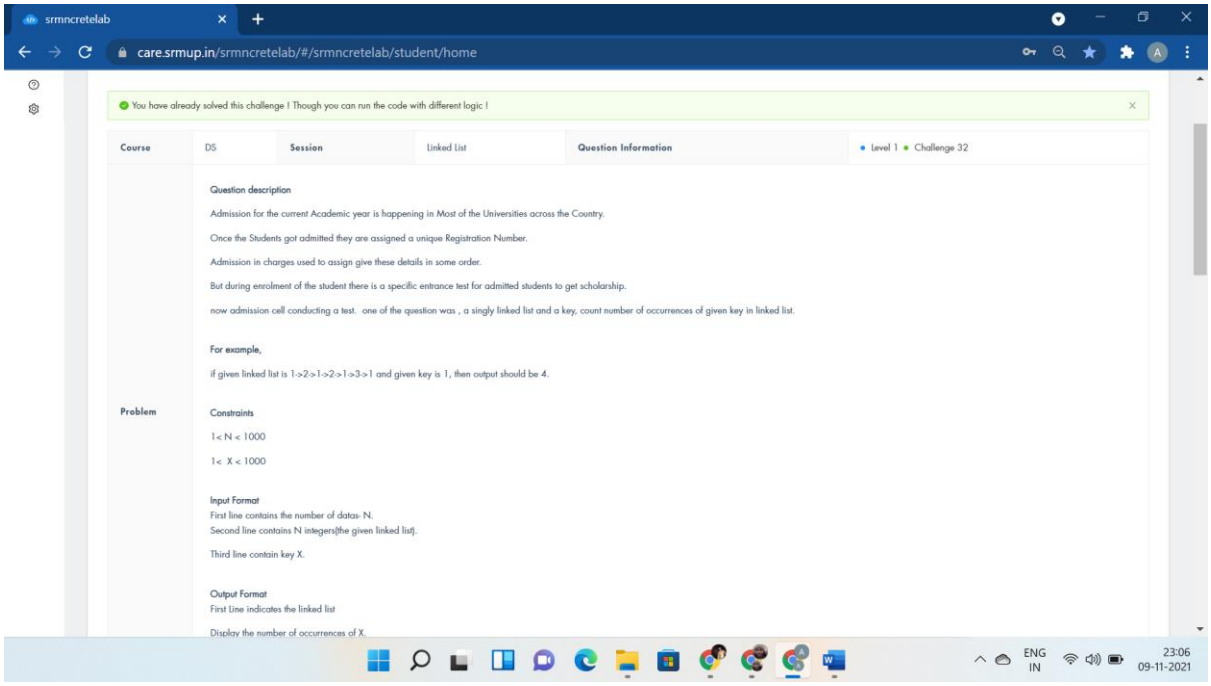
Linked List




```
#include <iostream>
using namespace std;
void tel(){
   return;}
struct node {
   int data;
   node *next;
}*head = NULL;
void create(){
   int n;
   cin >> n;
   struct node *p1 = new node;
   int m;
   cin >> m;
   p1->data = m;
   head = p1;
   int i;
   for (i = 0; i < n - 1; i++)  {
      int a;
```

```
        cin >> a;
        node *tt = new node;
        tt->data = a;
        p1->next = tt;
        p1=p1->next;
    }
    p1->next = NULL;
    int del;
    bool found = false;
    cin >> del;
    node *nn = head;
    while (nn != NULL) {
        nn = nn->next;
        node *dd = nn;
        int m = del;
        while (m-- > -1) {
            dd = dd->next;
            if (dd == NULL) {
                nn->next = NULL;
                found = true;
                break;
            }
        }
        if (found)
            break;
    }
    cout << "Linked List:";
    while (head != NULL){
        cout << "->" << head->data;
        head = head->next;
    }
}
int main(){
```

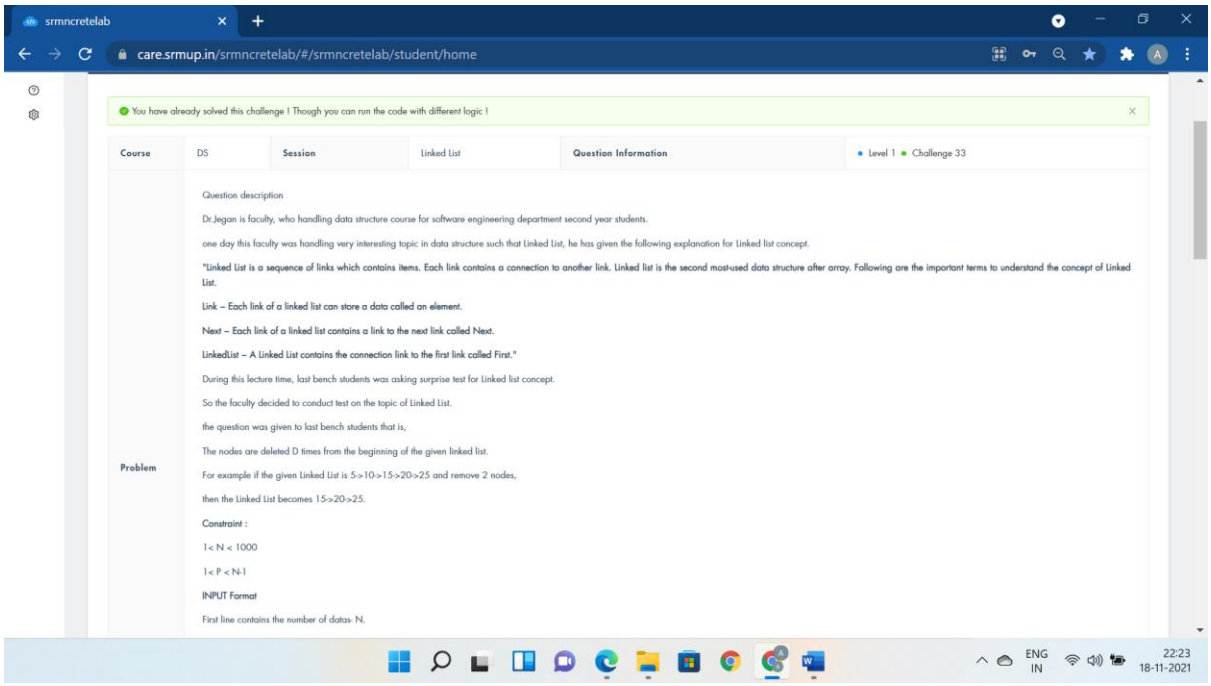
```
    create();
    return 0;
    cout << "for(i=0;i<n;i++)";
}
```




```
#include <bits/stdc++.h>
using namespace std;
struct node
{
        int key;
        struct node *next;
};
void push(struct node** head_ref, int new_key)
{
        struct node* new_node = new node();
        new_node->key = new_key;
        new_node->next = (*head_ref);
        (*head_ref) = new_node;
}
```

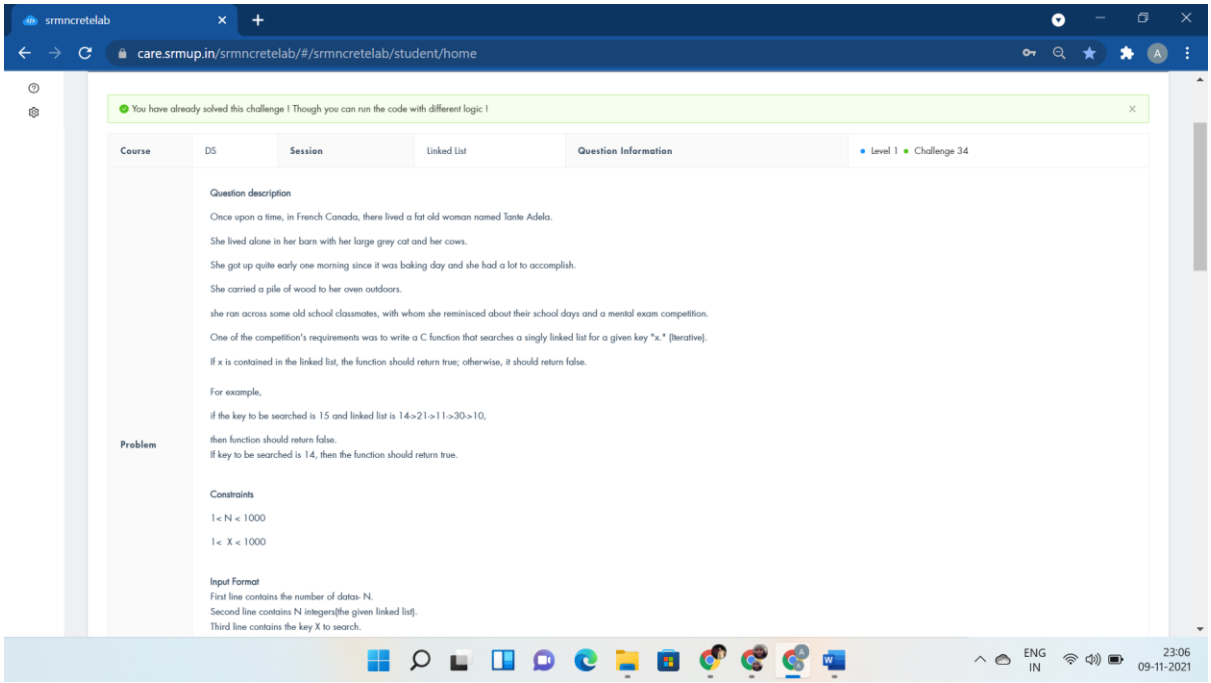
```
void printList(node *node){
	while (node != NULL)
	{
		cout<<"-->"<<node->key;
		node = node->next;
	}
}
int count(struct node* head,int search_for)
{
	node* current = head;
	int count=0;
	while (current != NULL)
	{
		if (current->key == search_for)
			count++;
		current = current->next;
	}
	return count;
}
int main()
{
	struct node* head = NULL;
	int x,n,t;
	cin>>n;
	while(n--){
	   cin>>t;
	   push(&head,t);
	}
	cin>>x;
	cout<<"Linked list:";
	printList(head);
	cout<<endl<<"Count of "<<x<<":"<<count(head, x);
	return 0;
```

}




```
#include<bits/stdc++.h>
using namespace std;
struct node {
   int data;
   node *next;
};
void insertAtEnd(node** head_ref, int new_data) {
   node* new_node = (node*)malloc(sizeof( node));
   node* last = *head_ref;
   new_node->data = new_data;
   new_node->next = NULL;
   if (*head_ref == NULL) {
      *head_ref = new_node;
      return;
   }
   while (last->next != NULL) last = last->next;
   last->next = new_node;
   return;
}
int main() {
```

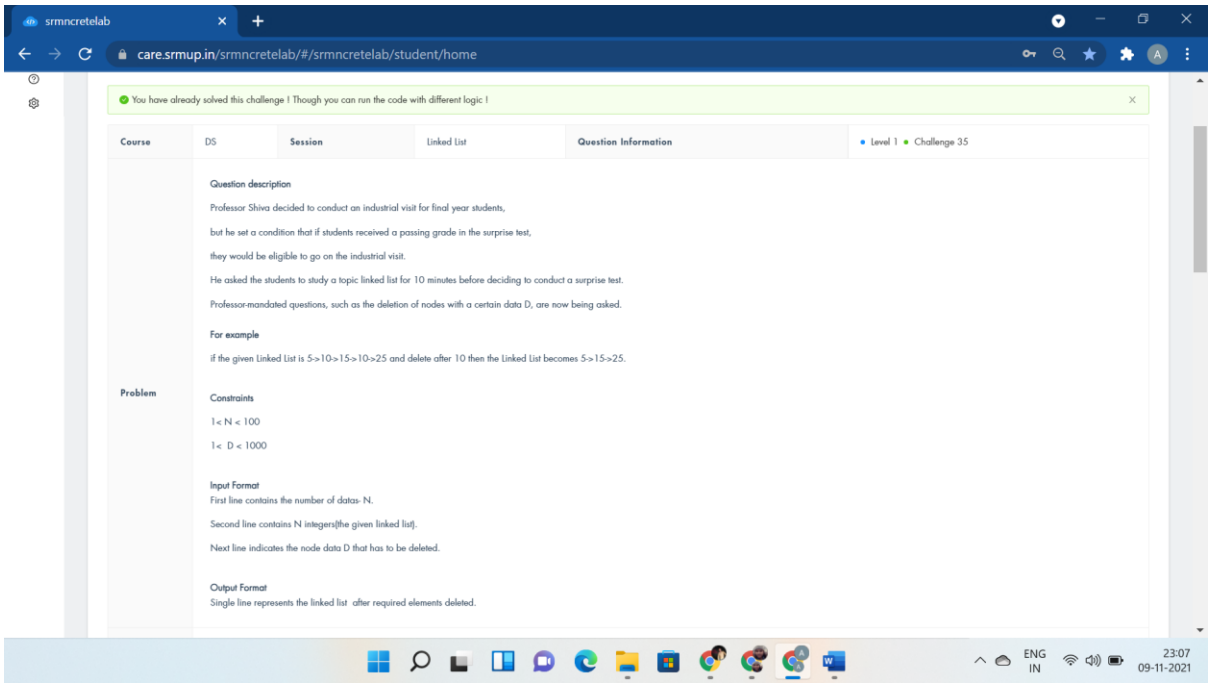
```
    node* head = NULL;
    int n,c,z,i;
    cin>>n;
    for(i=0;i<n;i++){
        cin>>c;
        insertAtEnd(&head,c);
    }
    cin>>z;
    for(int i=0;i<z;i++)
    head=head->next;
    cout << "Linked List:";
    node* node=head;
    while(node!=NULL){
        cout<<"->"<<node->data;
        node=node->next;
    }
    return 0;
    cout<<"void create()";
}
```




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

```
using namespace std;
struct node
{
	int key;
	struct node* next;
};
void push(struct node** head_ref, int new_key)
{
	struct node* new_node = new node();
	new_node->key = new_key;
	new_node->next = (*head_ref);
	(*head_ref) = new_node;
}
bool search(struct node* head,int x)
{
	node* current = head;
	while (current != NULL)
	{
		if (current->key == x)
			return true;
		current = current->next;
	}
	return false;
}
int main()
{
	struct node* head = NULL;
	int x,n,t;
	cin>>n;
	while(n--){
	   cin>>t;
	   push(&head,t);
	}
```

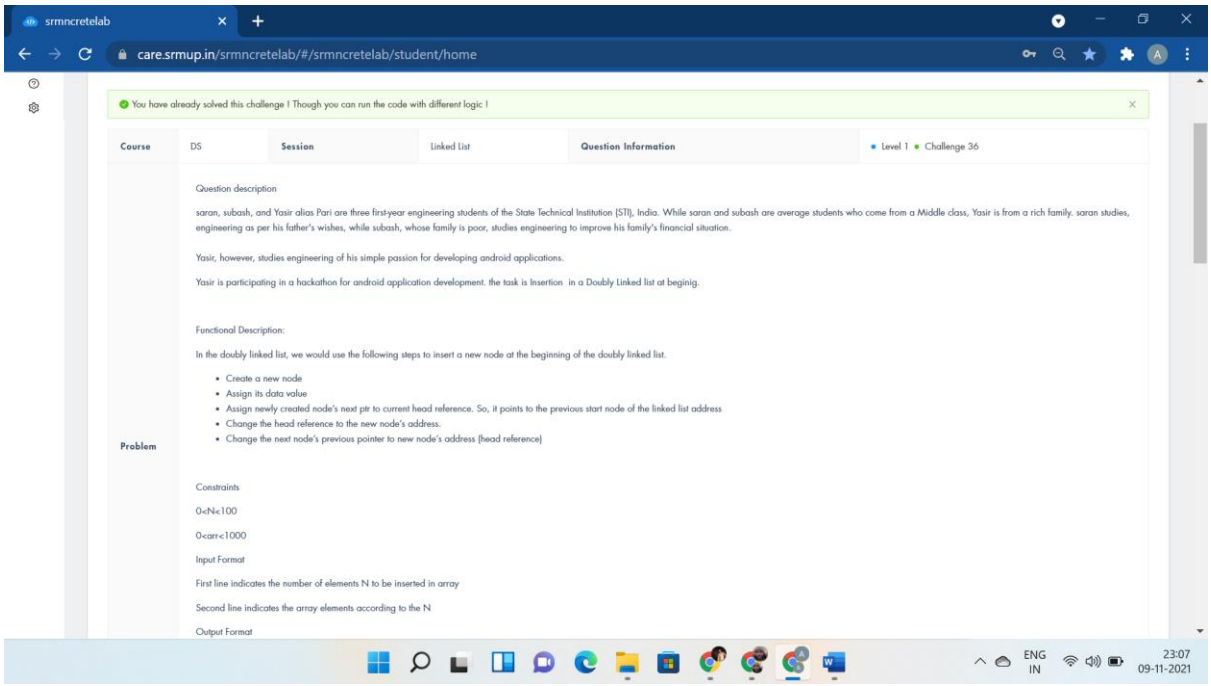
```
        cin>>x;
        search(head, x)? cout<<"Yes" : cout<<"No";
        return 0;
}
```




```
#include<iostream>
using namespace std;
struct node{
   int data;
   struct node *next;
}*start;
void display();
void deleteNode(node*& head, int val)
{
   if (head == NULL) {
      return;
   }
   if (head->data == val) {
      node* t = head;
      head = head->next;
```

```
        delete (t);
        return;
    }
    deleteNode(head->next, val);
}
int main() {
    int n;
    scanf("%d",&n);
    struct node *temp, *p2;
    start=NULL;
    for(int i=0;i<n;i++){
        temp=(struct node *)malloc(sizeof(struct node));
        scanf("%d", &temp -> data);
        temp->next = NULL;
        if(start == NULL){
            start= temp;
            p2 = temp;
        }
        else
        {
          p2->next=temp;
          p2=p2->next;
        }
    }
    int x;
    cin>>x;
    //display();
    for(int i=0;i<n;i++)
    deleteNode(start,x);
    display();
    return 0;
    cout<<"void del()void create() ";
}
```

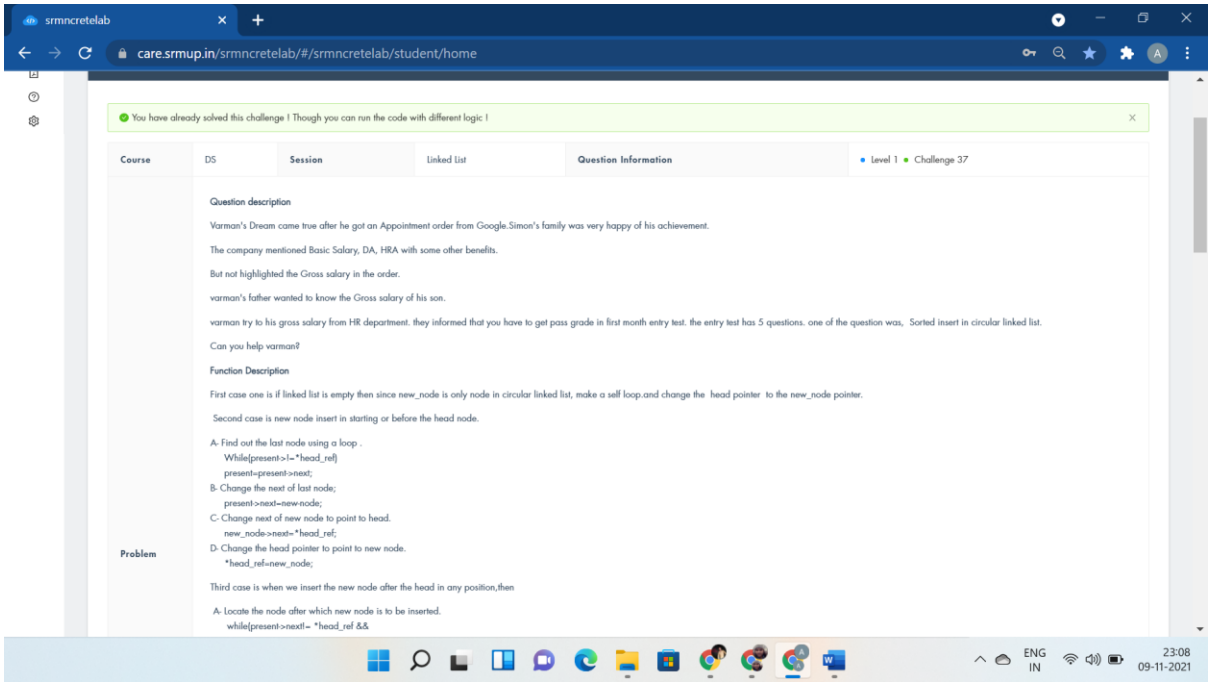
```
void display() {
   struct node *temp;
   temp = start;
   printf("Linked List:");
   while(temp != NULL)
   {
      printf("->%d",temp->data);
      temp = temp->next;
   }
}
```




```
#include <bits/stdc++.h>
using namespace std;
struct Node
{
        int data;
        struct Node *next;
        struct Node *prev;
};
```

```
void insertStart(struct Node** head,int data)
{
   struct Node* new_node = new Node();
   new_node->data = data;
   new_node->next = (*head);
   new_node->prev = NULL;
   if ((*head) != NULL)
      (*head)->prev = new_node;
   (*head) = new_node;
}
void printList(struct Node* node)
{
        Node* last;
        while (node != NULL)
        {
                cout<<node->data<<" ";
                last = node;
                node = node->next;
        }
        cout<<endl;
        while (last != NULL)
        {
                cout<<last->data<<" ";
                last = last->prev;
        }
}
int main()
{
        struct Node* head = NULL;
        int n;
        cin>>n;
        for(int i=0;i<n;i++){
           int t;
```

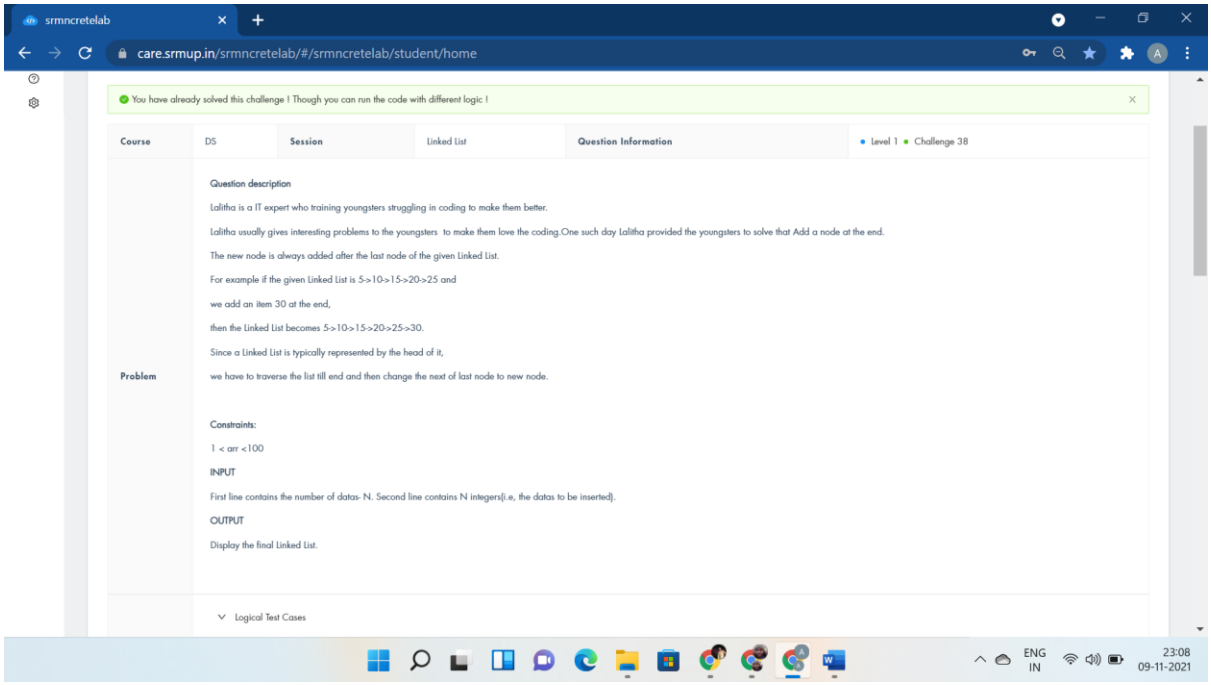
```
            cin>>t;
            insertStart(&head, t);
        }
        printList(head);
        return 0;
}
```




```
#include <stdio.h>
#include<stdlib.h>
struct Node{
    int data;
    struct Node *next;
};

void sortedInsert(struct Node** head_ref, struct Node* new_node)
{
    struct Node* current = *head_ref;

    if(current == NULL){
        new_node->next = new_node;
```

```
        *head_ref=new_node;
    }

    else if(current->data >= new_node->data){
        while(current->next != *head_ref)
        current=current->next;

        current->next=new_node;
        new_node->next=*head_ref;
        *head_ref=new_node;
    }

    else{
        while(current->next != *head_ref && current->next->data < new_node->data)
        current = current->next;

        new_node->next = current->next;
        current->next=new_node;
    }
}

void printList(struct Node *start){
    struct Node *temp;
    temp=start;

    do{
        printf("%d ",temp->data);
        temp=temp->next;
    }while(temp->next != start);
    printf("%d",temp->data);
}
int main()
{
```

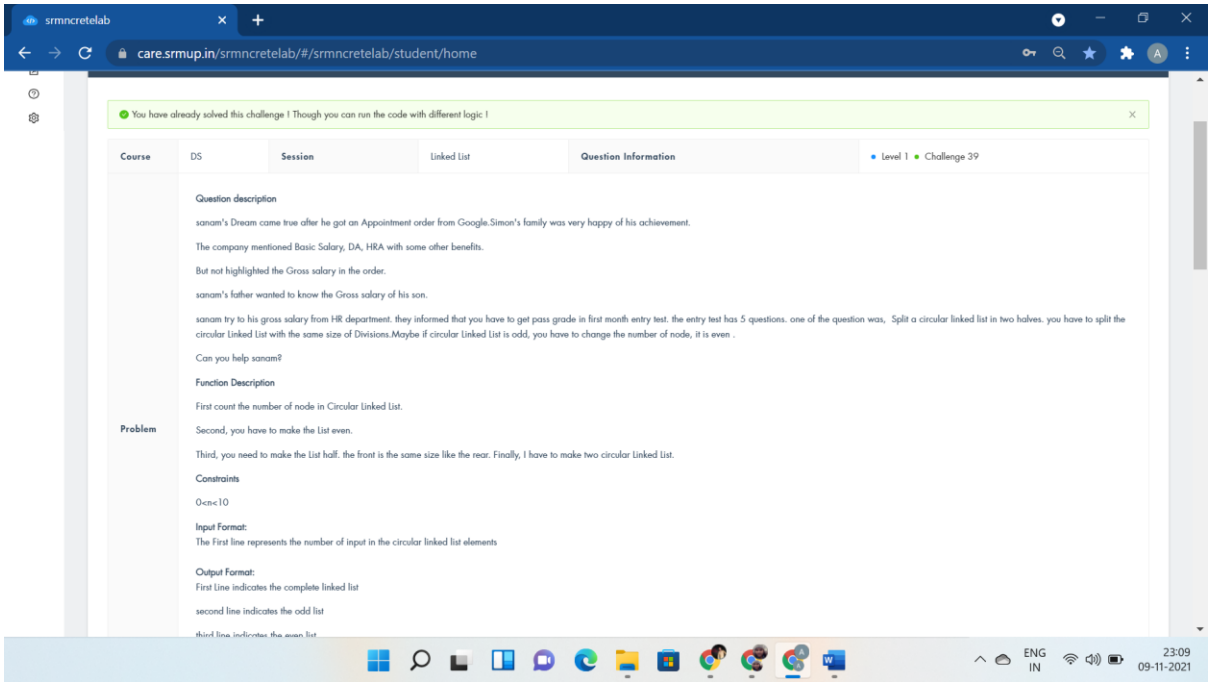
```
    int n,i;
    scanf("%d",&n);

    struct Node *start=NULL;
    struct Node *temp;

    for(i=0; i<n; i++){
        temp=(struct Node*)malloc(sizeof(struct Node));
        scanf("%d",&temp->data);
        sortedInsert(&start, temp);
    }
printList(start);
return 0;
}
```




```
#include <stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node *next;
```

```
}*start;
void display();
int main() {
   int n;
   scanf("%d",&n);
   struct node *temp, *p2;
   start=NULL;
  while(n) {
      temp=(struct node *)malloc(sizeof(struct node));
      scanf("%d", &temp -> data);
      temp->next = NULL;
      if(start == NULL){
         start= temp;
         p2 = temp;
      }
      else
      {
        p2->next=temp;
      //  while(p2 != NULL && p2 -> next != NULL       p2=p2->next;
        p2=p2->next;
      }--n;
   }
   display();
   return 0;
}
void display() {
   struct node *temp;
   temp = start;
   printf("Linked List:");
   while(temp != NULL)
   {
      printf("->%d",temp->data);
      temp = temp->next;
```

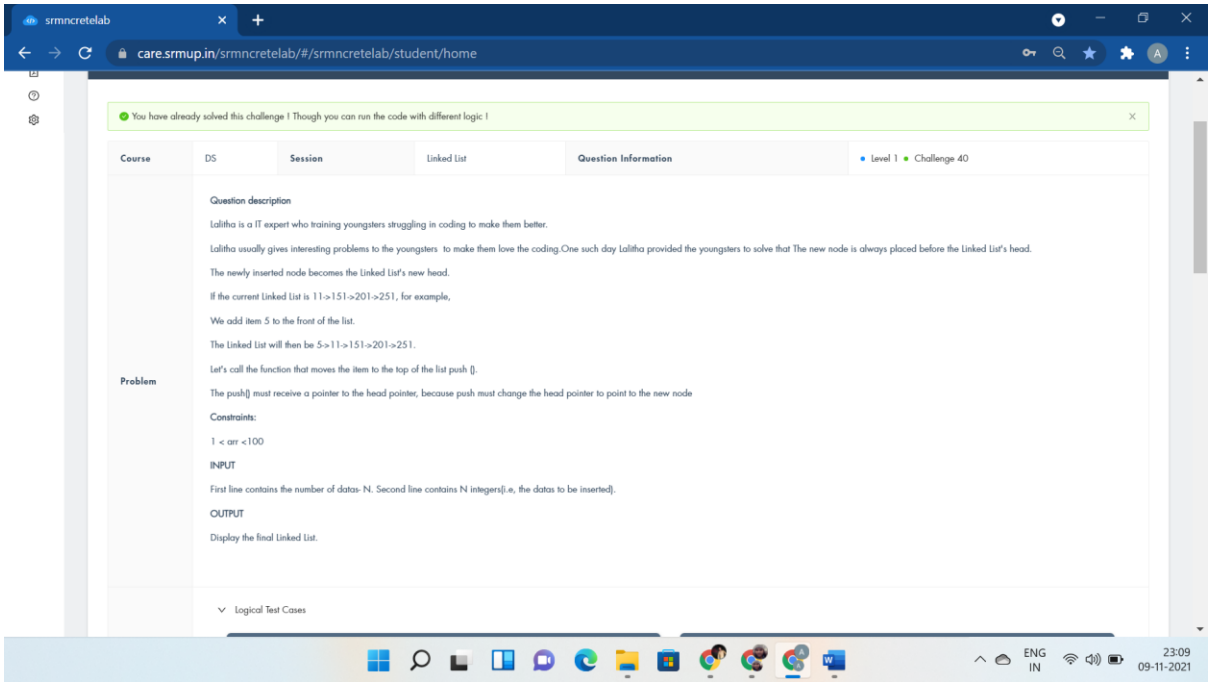
```
    }
}
```




```
#include <iostream>
using namespace std;
struct n
{
    int data;
    struct n *next;
} * odd, *even, *h = NULL, *tt;
void insert(int data)
{
    n *p = new n;
    p->data = data;
    p->next = NULL;
    tt->next = p;
    tt = p;
}
void oodd()
{
```

```
    cout << "Odd:\n";
    odd = h;
    int i = 1;
    cout << "[h]";
    while (odd != NULL)
    {
        if ((i % 2))
        {
            cout << "=>" << odd->data;
        }
        i++;
        odd = odd->next;
    }
    cout << "=>[h]";
}
void eeven()
{
    cout << "Even:\n";
    even = h;
    int i = 1;
    cout << "[h]";
    while (even != NULL)
    {
        if (!(i % 2))
        {
            cout << "=>" << even->data;
        }
        i++;
        even = even->next;
    }
    cout << "=>[h]";
}
void display(struct n *h)
```

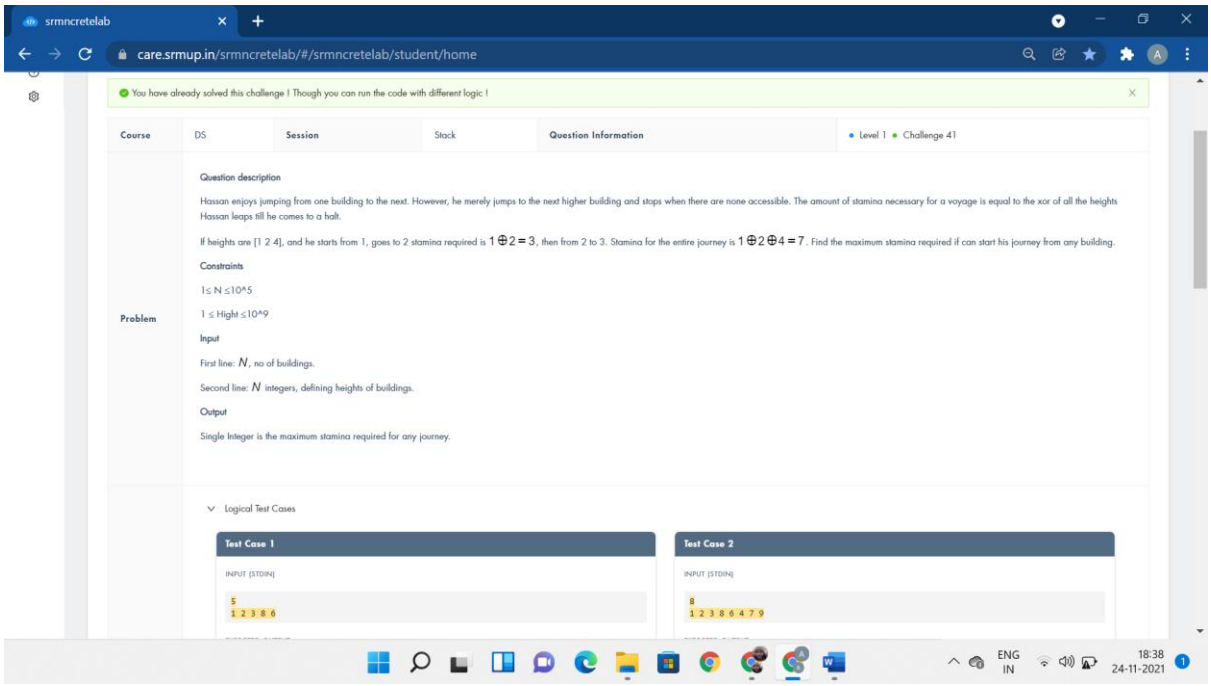
```
{
    cout << "Complete linked_list:\n[h]";
    while (h != NULL)
    {
        cout << "=>" << h->data;
        h = h->next;
    }
    cout << "=>[h]";
}
int main()
{
    int a;
    cin >> a;
    tt = new n;
    tt->data = 1;
    tt->next = NULL;
    h = tt;
    for (int i = 2; i <= a; i++)
    {
        insert(i);
    }
    n *y = h;
    display(y);
    cout << "\n";
    oodd();
    cout << "\n";
    eeven();
    return 0;
}
```




```
#include <bits/stdc++.h>
using namespace std;
struct node
{
	int data;
	node *next;
};
void push(node** start, int new_data){
	node* p1 = new node();
	p1->data = new_data;
	p1->next = *start;
	*start = p1;
}
void printList(node *node){
	while (node != NULL)
	{
		cout<<"->"<<node->data;
		node = node->next;
	}
}
```

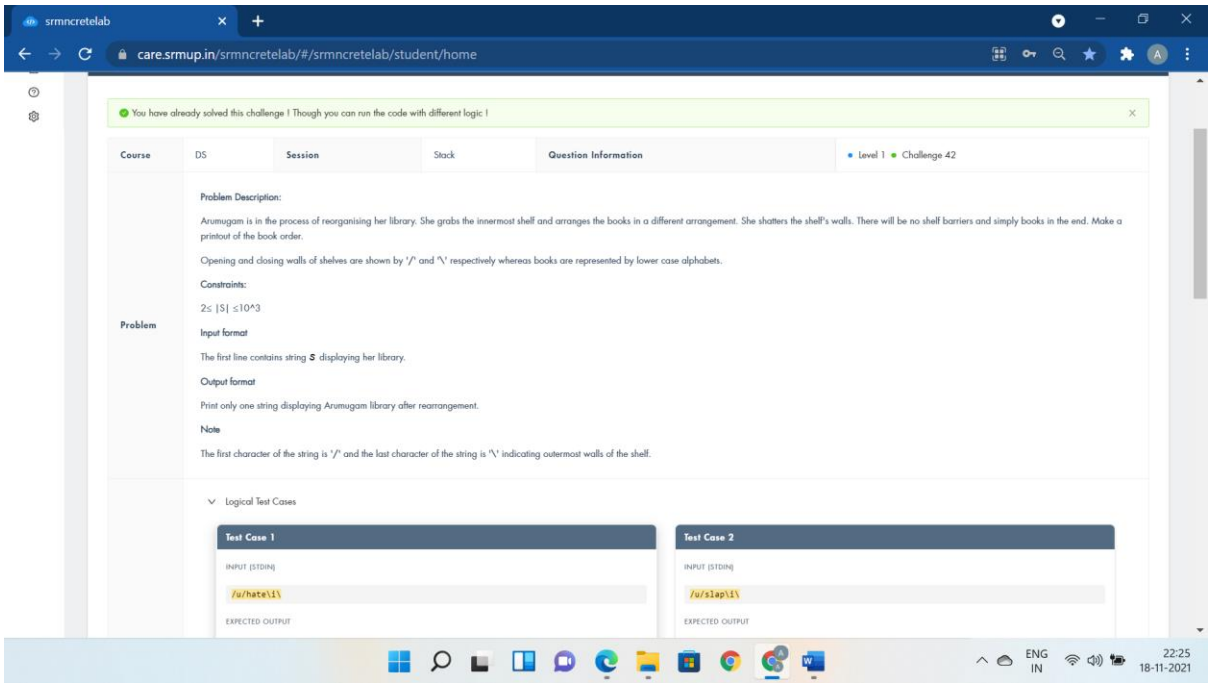
```
int main(){
        node *start = NULL;
        int n,t;
        cin>>n;
        while(n--){
           cin>>t;
           push(&start,t);
        }
        cout<<"Linked List:";
        printList(start);
        return 0;
        cout<<"p1->next=start; void display()";
}
```

STACKS:-

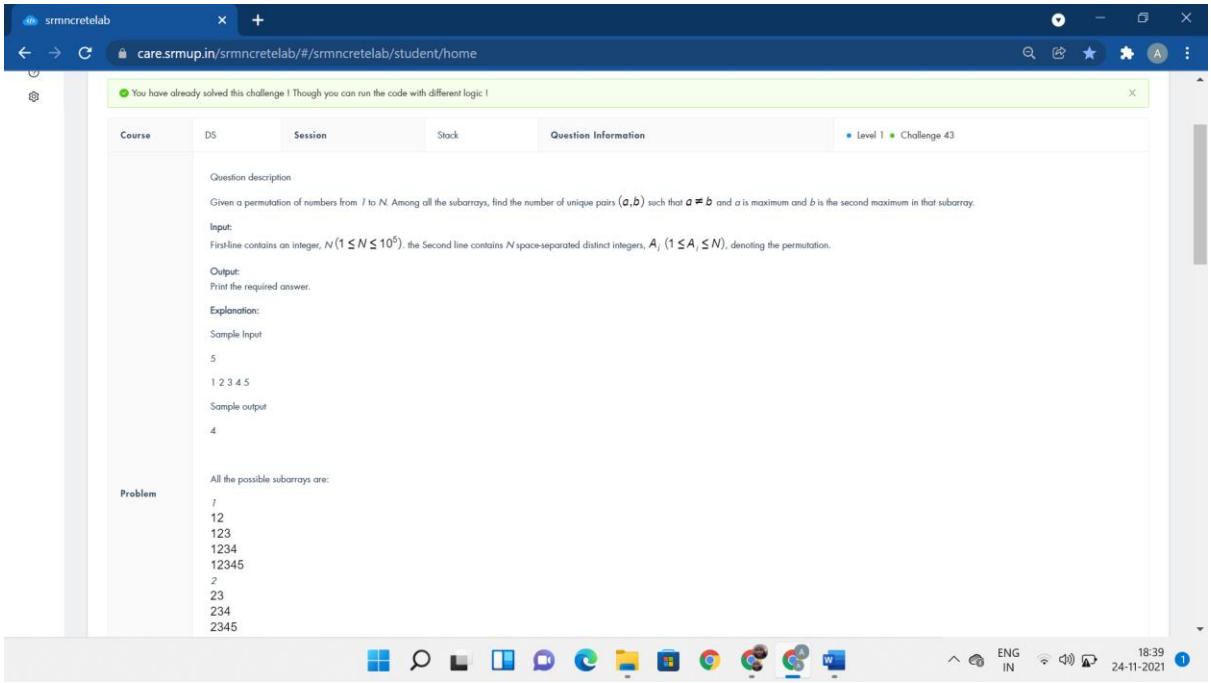



```
#include <stdio.h>
int main() {
   int i, j, arr[1000000], n, temp=0,st[1000000]= {0};
   scanf("%d",&n);
```

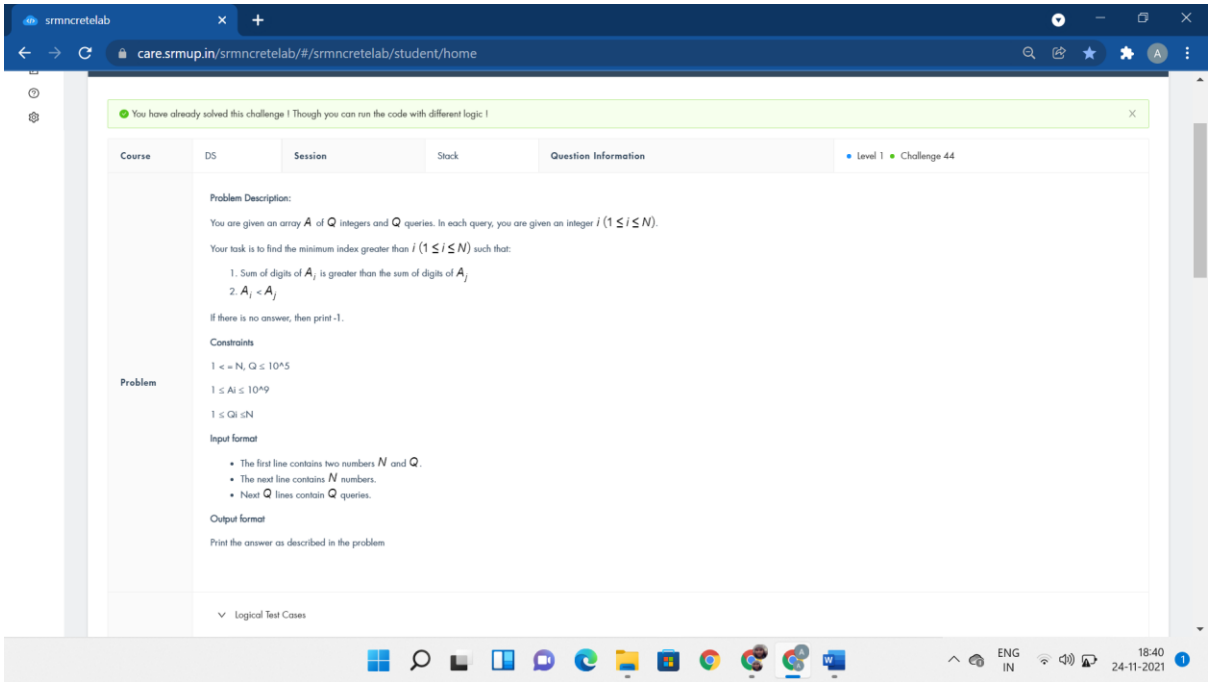
```
    for(i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    st[n-1] = arr[n-1];
    temp = arr[n-1];
    for(i=n-2;i>=0;i--) {
        for(j=i+1;j<n;j++)
        if(arr[i]<arr[j]) {
            st[i]=arr[i]^st[j];
        break;
       }
    if(st[i] == 0)
        st[i] = arr[i];
    if(st[i] > temp)
        temp = st[i];
    }
    printf("%d",temp);
    return 0;
}
```



```
#include <bits/stdc++.h>
using namespace std;
int main()
{
   string s,temp="";
   cin>>s;
   stack<string> stk;
   for (unsigned int i = 0; i < s.size(); i++) {
      if(s[i]==47||s[i]==92){
         if(!temp.empty()){
            stk.push(temp);
            temp.clear();
         }
      }
      else{
         temp.push_back(s[i]);
      }
   }
   while(!stk.empty()){
      cout<<stk.top();
      stk.pop();
   }

            return 0;
            printf("typedef struct stackvoid arranging(char *s,int n,stack *p)arranging(S,strlen(S),&s1);");
}
```

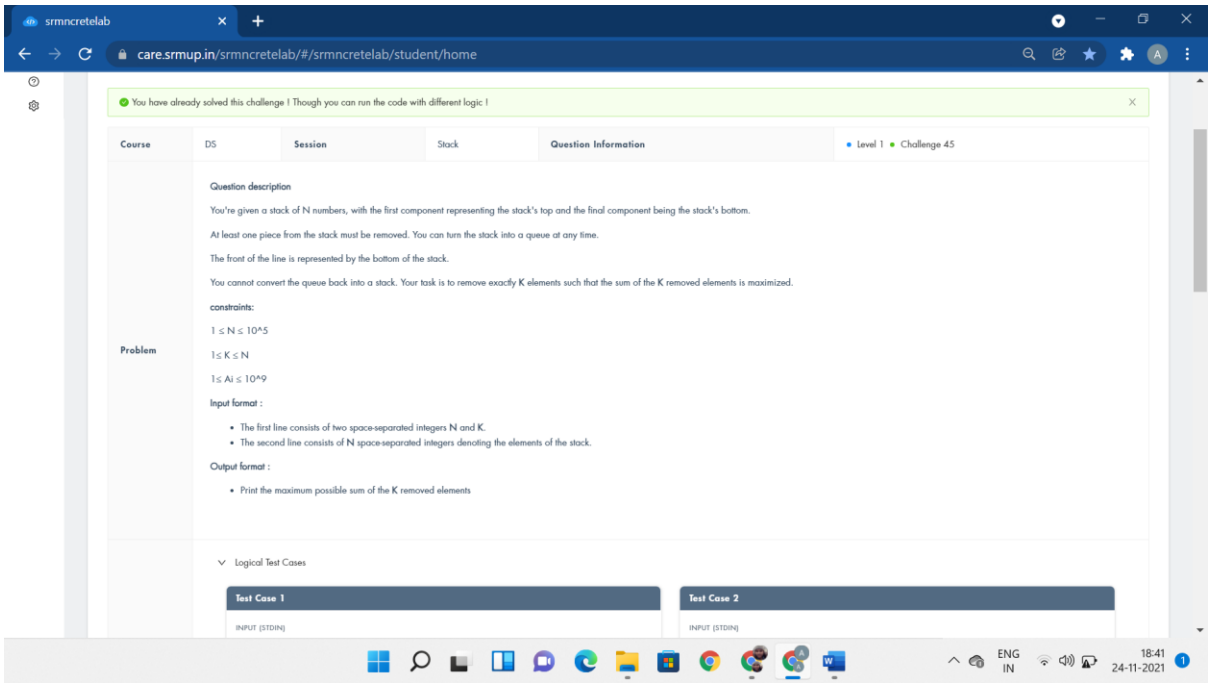



```
#include <stdio.h>

int main(){

   int num,i,count=0,a[100001],stck[100001],top=-1;

            scanf("%d", &num);

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

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

                        while(top!=-1 && stck[top]<a[i]) {

                                    top--;

                                    count++;

                        }

                        if (top!=-1) {

                                    count++;

                        }

                        stck[++top]=a[i];

            }

             printf("%d",count);

             return 0;

}
```



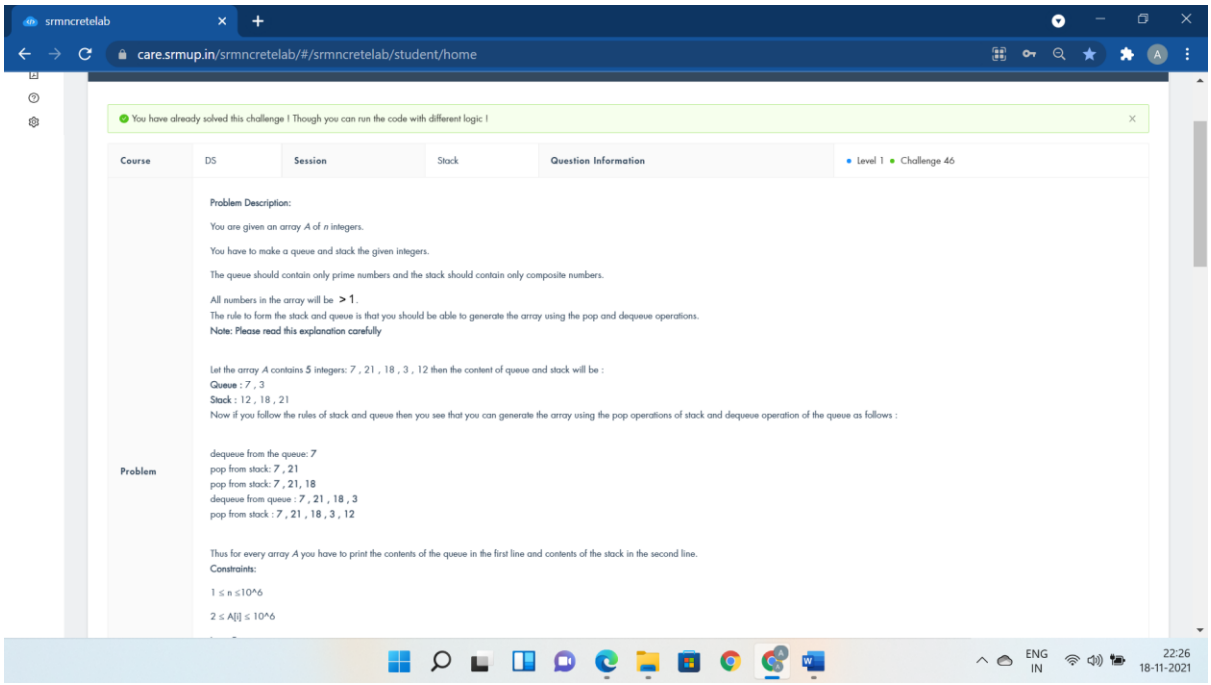

```
#include<bits/stdc++.h>
using namespace std;
int main(){
   int n,q;
   cin>>n>>q;
   int *a=new int [n];
   for(int i=0;i<n;i++){
      cin>>a[i];
   }
   int *arr=new int[n];
   for(int i=0;i<n;i++){
      int z=a[i];
      int sum=0;
      while(z>0){
         sum+=(z%10);
         z=z/10;
      }
      arr[i]=sum;
   }
   while(q--){
      int Q;
      cin>>Q;
```

```
        int ans=-1;
        for(int i=Q;i<n;i++){
            if(a[i]>a[Q-1] && arr[i]<arr[Q-1]){
                ans=i+1;
                break;
            }else{
                continue;
            }
        }
        cout<<ans<<' ';
    }
    return 0;
    cout<<"if(arr[x]<arr[y]) if(arr2[x]>arr2[y]) ";
}
```

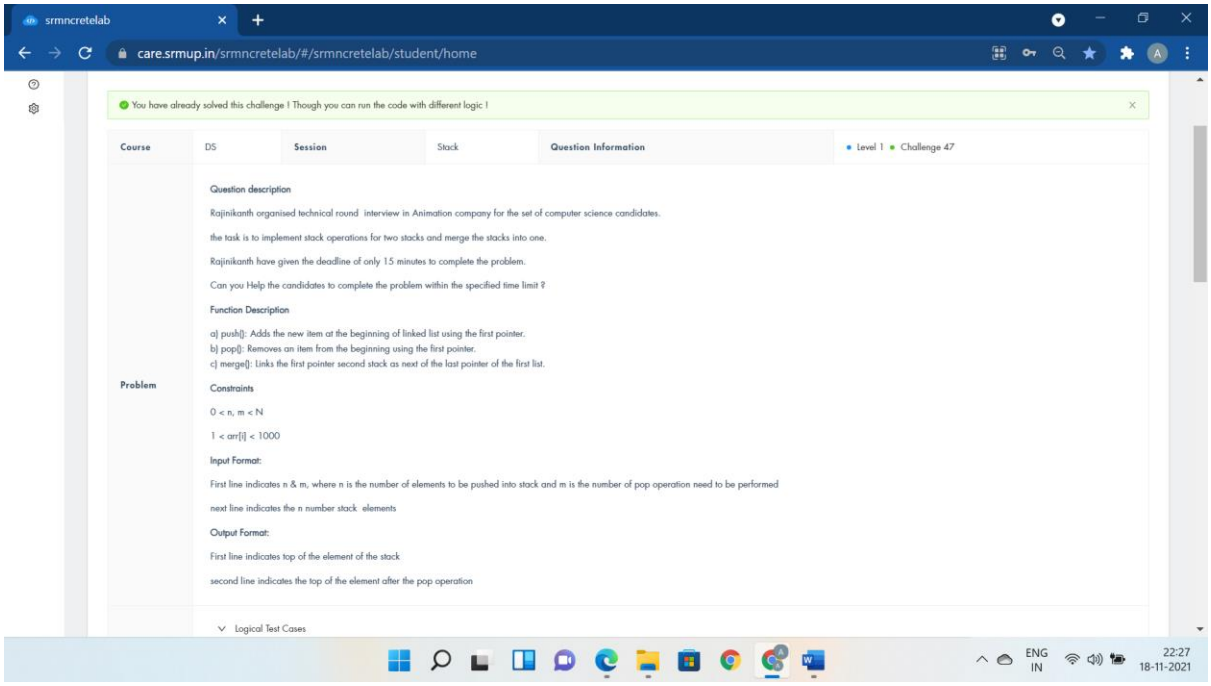



```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n,k,i;
    cin>>n>>k;
    int sum = 0;
```

```
    int arr[n];
    stack<int>st, st2;
    for(i=0;i<n;i++){
        cin >> arr[i];
        st.push(arr[i]);
    }
    for(i=0;i<k;i++){
        st2.push(arr[i]);
        sum += arr[i];
    }
    int maxs = sum;
    while(k-- > 1){
        sum -= st2.top();
        st2.pop();
        sum += st.top();
        st.pop();
        if(sum > maxs) maxs = sum;
    }
    cout << maxs;
    return 0;
}
```



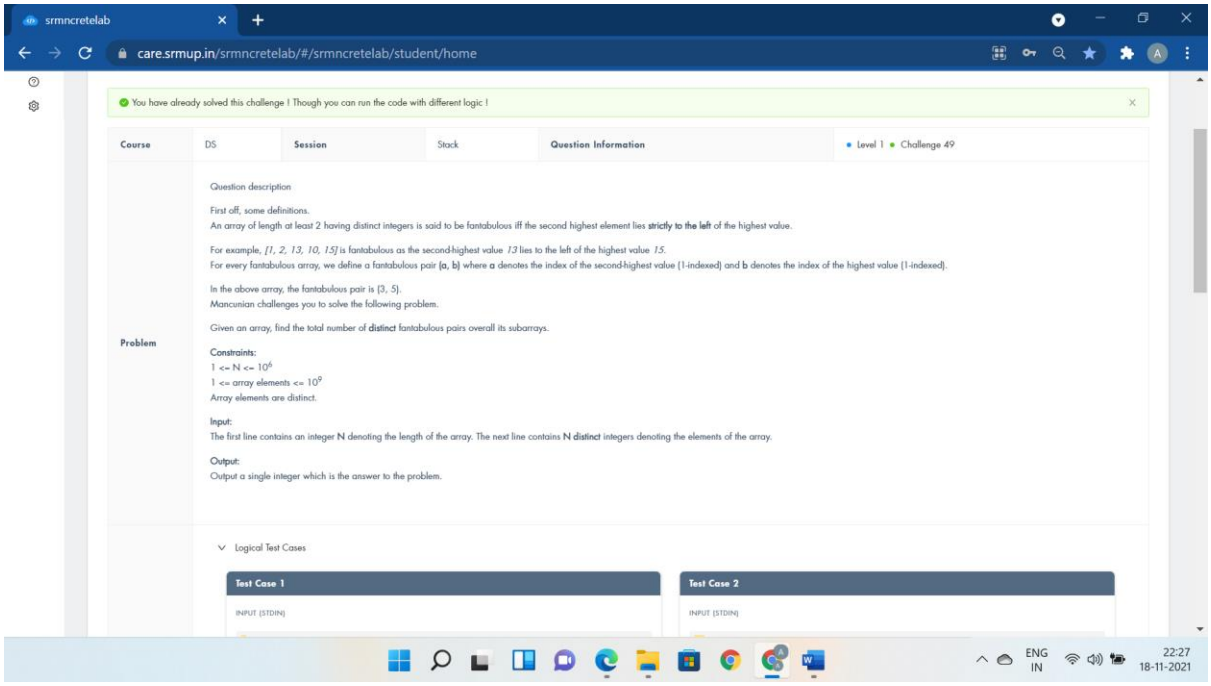
```
#include<bits/stdc++.h>
using namespace std;
bool isPrime(int n)
{
   if(n<=1)
   return false;
   for(int i=2;i<n;i++)
   if(n%i==0)
   return false;
   return true;
}
int main(){
   stack<int> stack;
   int n;
   cin>>n;
   int a[n];
   for(int i=0;i<n;i++){
      cin>>a[i];
      if(isPrime(a[i]))
      cout<<a[i]<<" ";
      else
      stack.push(a[i]);
   }
   cout<<endl;
   while(!stack.empty()){
      cout<<stack.top()<<" ";
      stack.pop();
   }
   return 0;
   cout<<"int read_int() void push(int stack[],int data) top++;";
}
```




```
#include <iostream>

using namespace std;

class node {

public:

        int data;

        node* next;

};

class mystack {

public:

        node* head;

        node* tail;


        mystack()

        {

                head = NULL;

                tail = NULL;

        }

};

mystack* create()

{

        mystack* ms = new mystack();

        return ms;
```

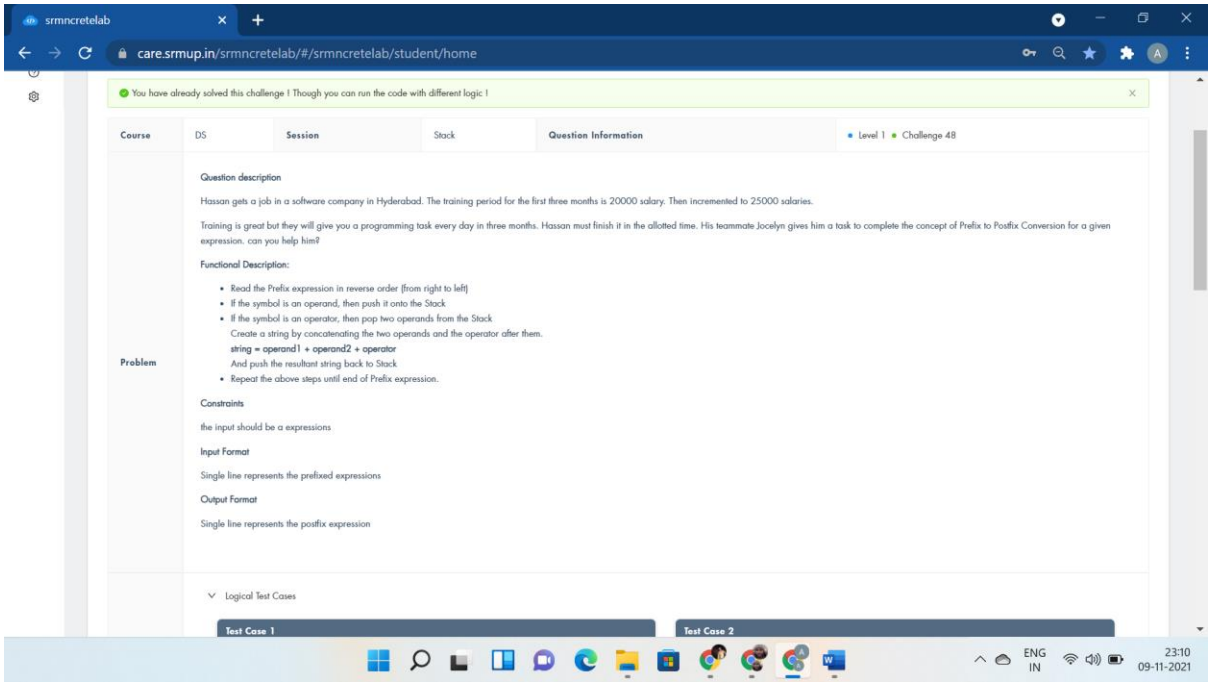
```
}
void push(int data,mystack* ms)
{
	node* temp = new node();
	temp->data = data;
	temp->next = ms->head;
	if (ms->head == NULL)
		ms->tail = temp;

	ms->head = temp;
}
int pop(mystack* ms)
{
	if (ms->head == NULL) {
		cout << "stack underflow" << endl;
		return 0;
	}
	else {
		node* temp = ms->head;
		ms->head = ms->head->next;
		int popped = temp->data;
		delete temp;
		return popped;
	}
}
void merge(mystack* ms1,mystack* ms2)
{
if (ms1->head == NULL)
{
	ms1->head = ms2->head;
	ms1->tail = ms2->tail;
	return;
}

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

```
ms1->tail = ms2->tail;
}
void display(mystack* ms)
{
	node* temp = ms->head;
	while (temp != NULL) {
		cout << temp->data << " ";
		temp = temp->next;
	}
}
int main()
{
	mystack* ms1 = create();
	mystack* ms2 = create();
	int n,m,t;
	cin>>n>>m;
	for(int i=0;i<n;i++)
	{
	   cin>>t;
	   push(t,ms1);
	}
	for(int i=0;i<m;i++)
	{
	   cin>>t;
	   push(t,ms2);
	}
	merge(ms1, ms2);
	for(int i=0;i<n+m;i++)
	cout<<pop(ms1)<<" ";
}
```



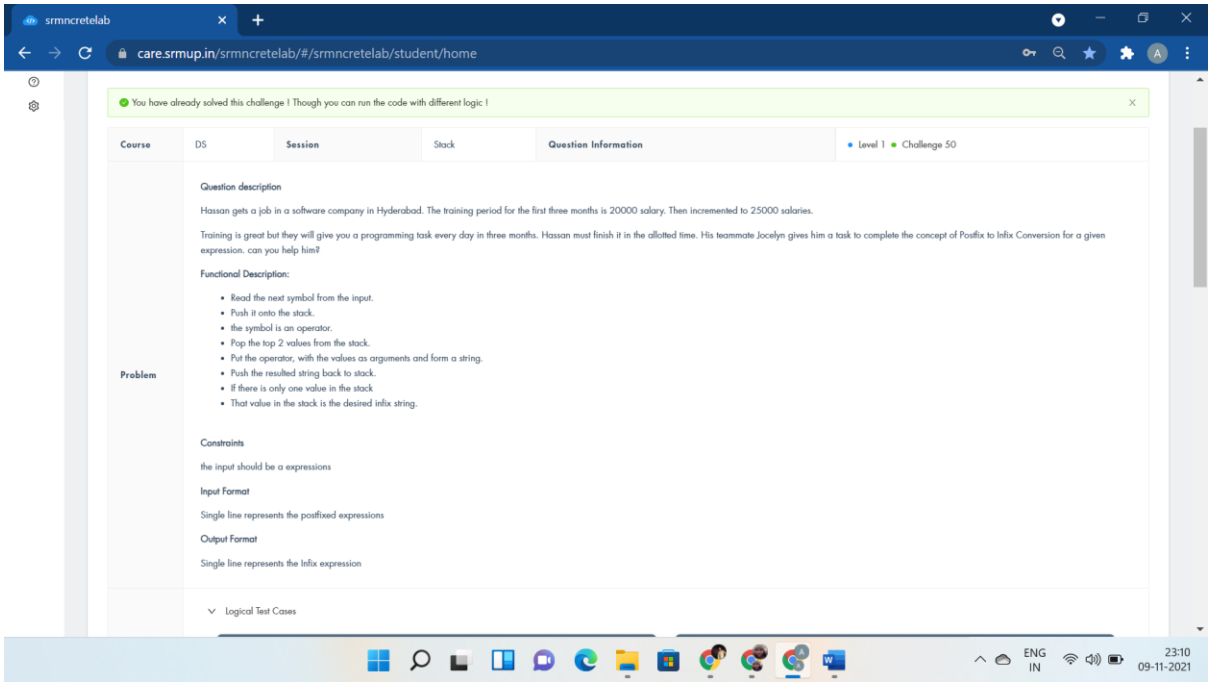

```
#include <bits/stdc++.h>

using namespace std;

#define sci(x) scanf("%d", &x)
#define scl(x) scanf("%lld", &x)

int arr[1000001], cnt[1000001];
int v[1000001];
stack <int> st;

void don(){
   cout<<"void push(llint num)stack[top++]=num;pop()";
}

int main()
{
        int n, i, x;
        sci(n);
        for (i = 1; i <= n; ++i) sci(arr[i]);

        for (i = n; i > 0; --i) {
```

```
		while (!st.empty() && arr[i] > arr[st.top()]) {
			cnt[st.top()] = st.top() - i;
			st.pop();
		}
		st.push(i);
	}
	while (!st.empty()) {
		cnt[st.top()] = st.top();
		st.pop();
	}

	for (i = 1; i <= n; ++i) {
		while (!st.empty() && arr[st.top()] < arr[i]) {
			x = i - st.top() + 1;
			v[x] = max(v[x], cnt[st.top()]);
			st.pop();
		}
		st.push(i);
	}

	int k = 0;
	for (i = 2; i <= n; ++i) {
		k += v[i];
	}

	cout << k << endl;

	return 0;
}
```




```
#include <iostream>
#include <stack>
using namespace std;
bool isOperator(char x)
{
	switch (x) {
	case '+':
	case '-':
	case '/':
	case '*':
		return true;
	}
	return false;
}
string preToPost(string pre_exp)
{
	stack<string> s;
	int length = pre_exp.size();
	for (int i = length - 1; i >= 0; i--)
	{
		if (isOperator(pre_exp[i]))
		{
```

```
                string op1 = s.top();
                s.pop();
                string op2 = s.top();
                s.pop();
                string temp = op1 + op2 + pre_exp[i];
                s.push(temp);
            }
            else {
                s.push(string(1, pre_exp[i]));
            }
        }
        return s.top();
}
int main()
{
        string pre_exp;
        cin>>pre_exp;
        cout << "Postfix:" << preToPost(pre_exp);
        return 0;
}
```

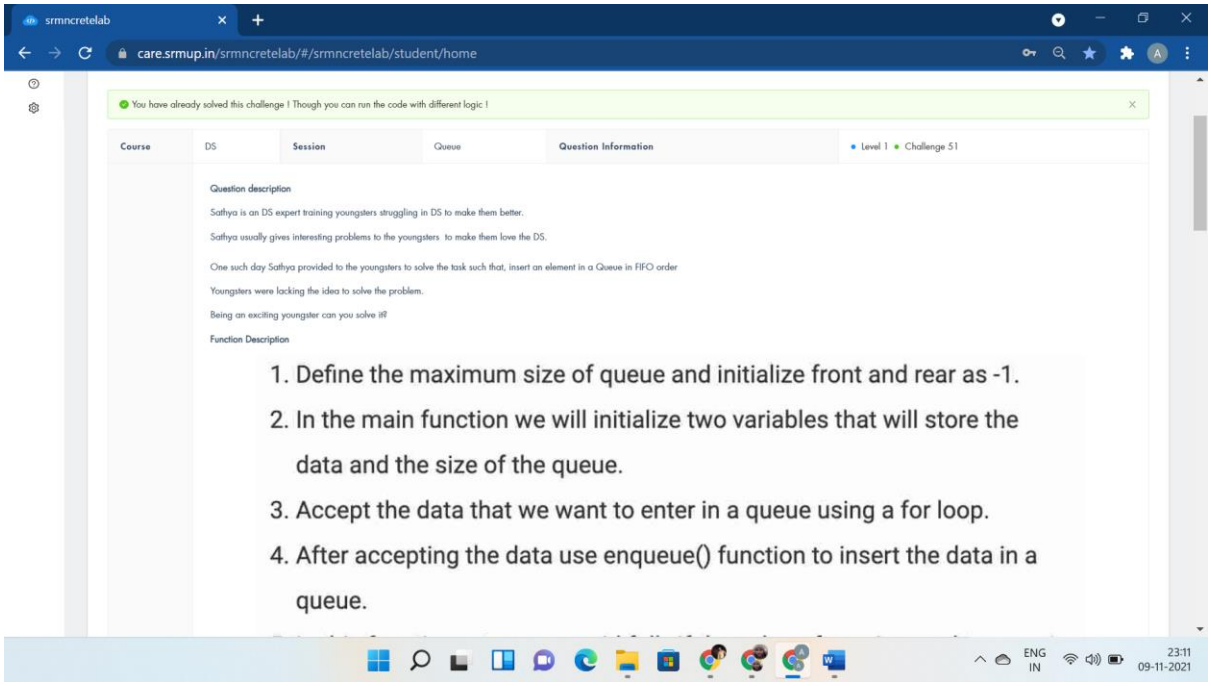



```
#include <bits/stdc++.h>
```

```
#include<iostream>
#include<string.h>
using namespace std;

bool isOperand(char x){
   return (x>='a' && x<='z') || (x >= 'A' && x <= 'Z');
}

string getInfix(string exp)
{
   stack<string> s;

   for(int i=0; exp[i]!='\0'; i++)
   {
      if(isOperand(exp[i]))
      {
         string op(1, exp[i]);
         s.push(op);
      }

      else
      {
         string op1 = s.top();
         s.pop();
         string op2=s.top();
         s.pop();
         s.push("(" + op2 + exp[i] + op1 + ")");
      }
   }
   return(s.top());
}
int main()
{
   string exp;
   cin>>exp;
```

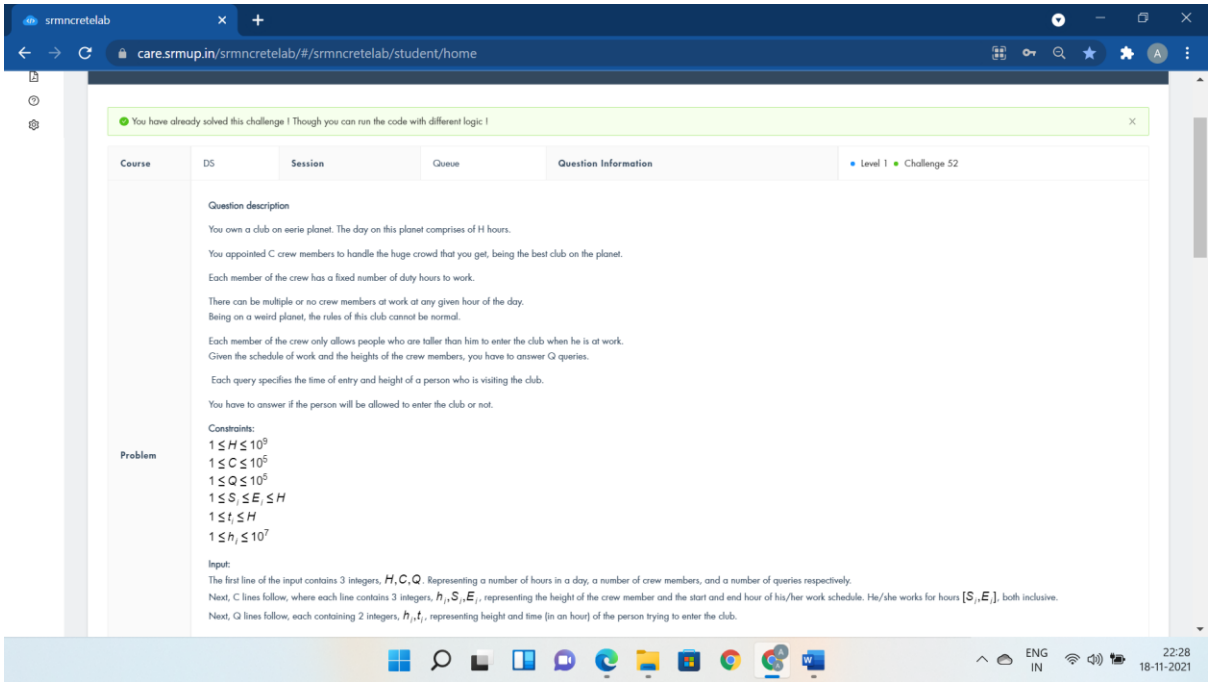
```
   cout<<getInfix(exp);


            return 0;
}
```

QUEUES:-



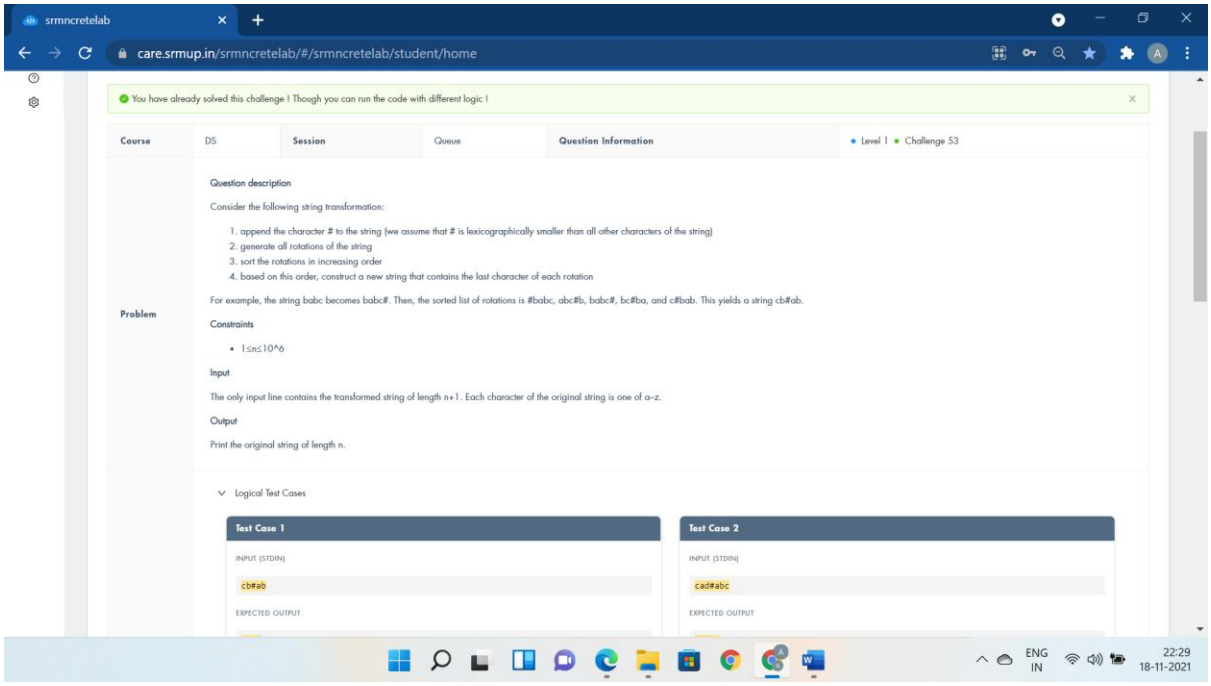

```
#include <stdio.h>
#define SIZE 100
void enqueue(int);
void display();
int items[SIZE], front = -1, rear = -1;
int main() {
 int n,data,i;
 scanf("%d",&n);
 for(i=0;i<n;i++)
 {
    scanf("%d",&data);
    enqueue(data);
    display();
 }
```

```
  return 0;
}
void enqueue(int data) {
  if (rear == SIZE - 1)
    printf("Queue is Full!!");
  else {
    if (front == -1)
      front = 0;
    rear++;
    items[rear] = data;
    printf("Enqueuing %d\n", data);
  }
}
void display() {
  if (rear == -1)
    printf("\nQueue is Empty!!!");
  else {
    int i;
    for(i=front;i<=rear;i++)
      printf("%d ", items[i]);
  }
}
```



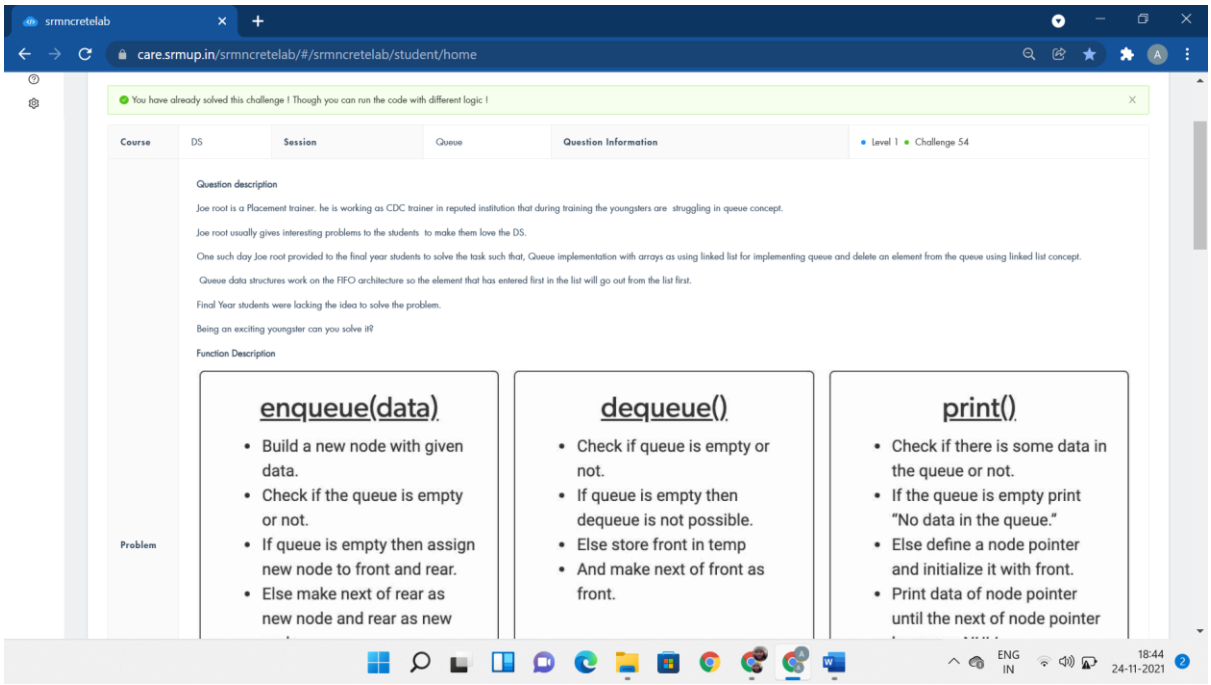

```
#include<stdio.h>
int main()
{
 long long int i,j,t,H,C,height,Q,S[100000],E[100000],h[100000];
 long long int nc=0,val=0,flag=0,maximum_height=0;
 scanf("%lld%lld%lld",&H,&C,&Q);

 for(i=0;i<C;i++)
 {
  scanf("%lld%lld%lld",&h[i],&S[i],&E[i]);
  if(h[i]>maximum_height)
   maximum_height=h[i];
 }

 for(i=0;i<Q;i++)
 {
  scanf("%lld%lld",&height,&t);
  if(height>maximum_height)
    printf("YES\n");
  else{
  val=0;
  nc=0;
```

```
   flag=0;
   for(j=0;j<C;j++)
   {
     if(t>=S[j] && t<=E[j])
     {
       nc++;
       if(height<=h[j])
       {
         printf("NO\n");
         flag=1;
         break;
       }
       else
        val++;
     }
   }

   if(nc==val)
       printf("YES\n");
   else
     if(flag==0)
       printf("NO\n");
   }

  }
  return 0;
  printf("void enqueue(long long h,long long start,long long end) while(c--)");
}
```




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



int main() {
   int i;
   string s; cin>>s;
   vector<int> v;
   vector<int> a[26];
   int n= s.size();
   for(i=0;i<=n;i++) {
      if (s[i] == '#')
         v.push_back(i);
      else
         a[s[i]-'a'].push_back(i);
   }
   for (int i = 0; i < 26; i++) {
      for (auto j: a[i])
         v.push_back(j);
   }
   string ans;
   int j = v[v[0]];
```

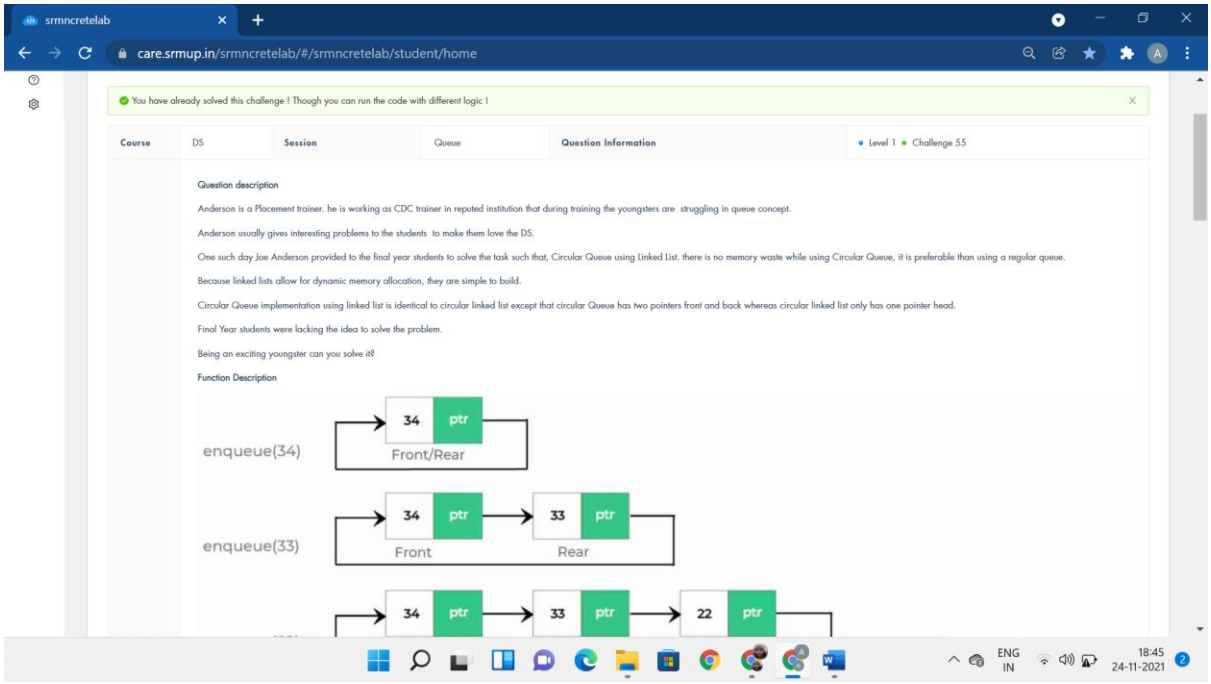
```
    while(s[j] != '#') {
        ans += s[j];
        j = v[j];
    }
    cout<<ans;
    return 0;
}
```




```
#include <stdio.h>
#include <stdlib.h>
struct node *front = NULL;
struct node *rear = NULL;
struct node
{
    int data;
    struct node *next;
};
void linkedListTraversal(struct node *ptr)
{
    //printf("Printing the elements of this linked list\n");
    while (ptr != NULL)
    {
        printf("%d ", ptr->data);
```

```
        ptr = ptr->next;
    }
}
void enqueue(int d)
{
    struct node* new_n;
    new_n = (struct node*)malloc(sizeof(struct node));
    if(new_n==NULL){
        printf("Queue is Full");
    }
    else{
        new_n->data = d;
        new_n->next = NULL;
        if(front==NULL){
            front=rear=new_n;
        }
        else{
            rear->next = new_n;
            rear=new_n;
        }
    }
}
int dequeue()
{
    int val = -1;
    struct node *ptr = front;
    if(front==NULL){
        printf("Queue is Empty\n");
    }
    else{
        front = front->next;
        val = ptr->data;
        free(ptr);
    }
    return val;
```

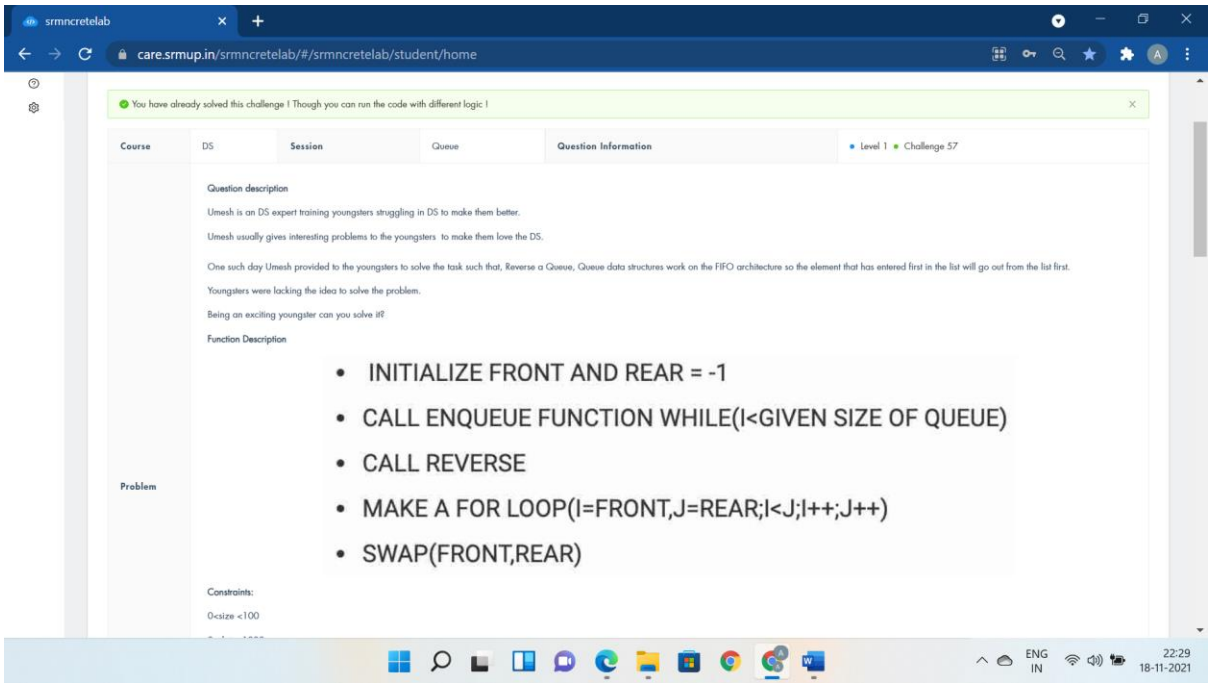
```
}
int main()
{
   int n,i,t;
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
      scanf("%d",&t);
      enqueue(t);
   }
   linkedListTraversal(front);
   dequeue();
   printf("\n");
   linkedListTraversal(front);
   return 0;
}
```




```
#include <stdio.h>
#include <stdlib.h>
struct node *f = NULL;
struct node *r = NULL;
struct node
```

```
{
   int data;
   struct node* next;
};
void enqueue(int d)
{
   struct node *n;
   n = (struct node*)malloc(sizeof(struct node));
   if(n==NULL){
      printf("Queue is Full");
   }
   else{
      n->data = d;
      n->next = NULL;
      if(f==NULL){
         f=r=n;
      }
      else{
         r->next = n;
         r=n;
      }
   }
}
int dequeue()
{
   int val = -1;
   struct node* t;
   t = f;
   if(f==NULL){
      printf("Queue is Empty\n");
   }
   else{
      f = f->next;
      val = t->data;
      free(t);
```

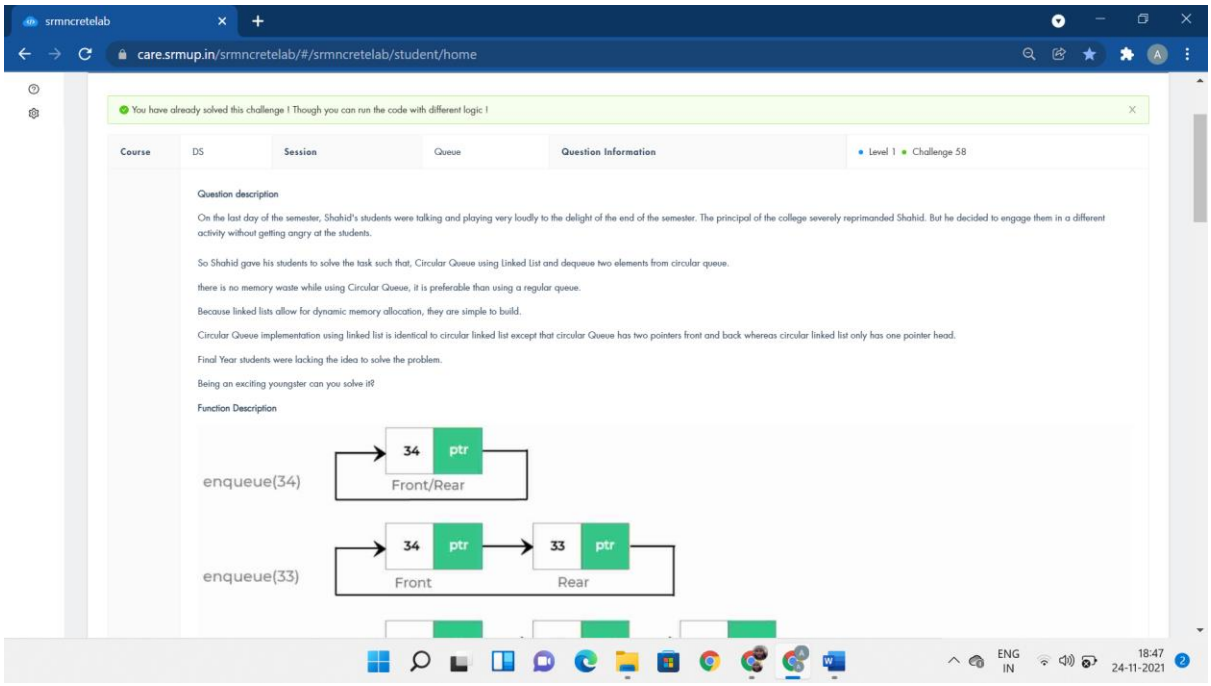
```
    }
    return val;
}
int main()
{
    int n,i,t;
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&t);
        enqueue(t);
    }
    for(i=0;i<n;i++){
        printf("%d\n",dequeue());
    }
    return 0;
}
```




```
#include <stdio.h>
#define SIZE 100
void enqueue(int,int);
void display();
```

```
void reverse();
int items[SIZE], front = -1, rear = -1;
int main() {
  int n,t,i;
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
     scanf("%d",&t);
     enqueue(t,n);
  }
  printf("Queue:");
  display();
  reverse();
  printf("\nReversed Queue:");
  display();
  return 0;
}
void reverse(){
    int i,j,temp;
    for(i=front,j=rear;i<j;i++,j--){
        temp=items[i];
        items[i]=items[j];
        items[j]=temp;
    }
}
void enqueue(int data,int l) {
  if (rear == l - 1)
    printf("Queue is Full!!");
  else {
    if (front == -1)
      front = 0;
    rear++;
    items[rear] = data;
   // printf("Enqueuing %d\n", data);
  }
```

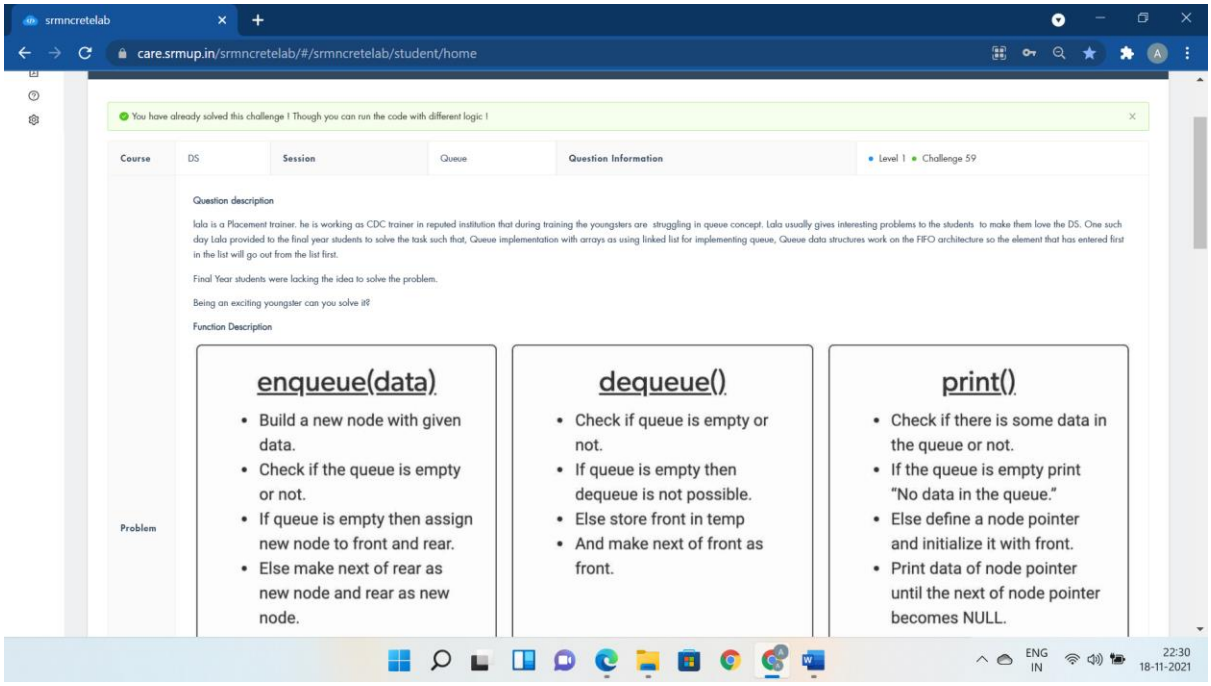
```
}
void display() {
 if (rear == -1)
   printf("\nQueue is Empty!!!");
 else {
   int i;
   for(i=front;i<=rear;i++)
     printf("%d ", items[i]);
 }
}
```




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

struct node *f = NULL;
struct node *r = NULL;

struct node
{
   int data;
   struct node* next;
};
```

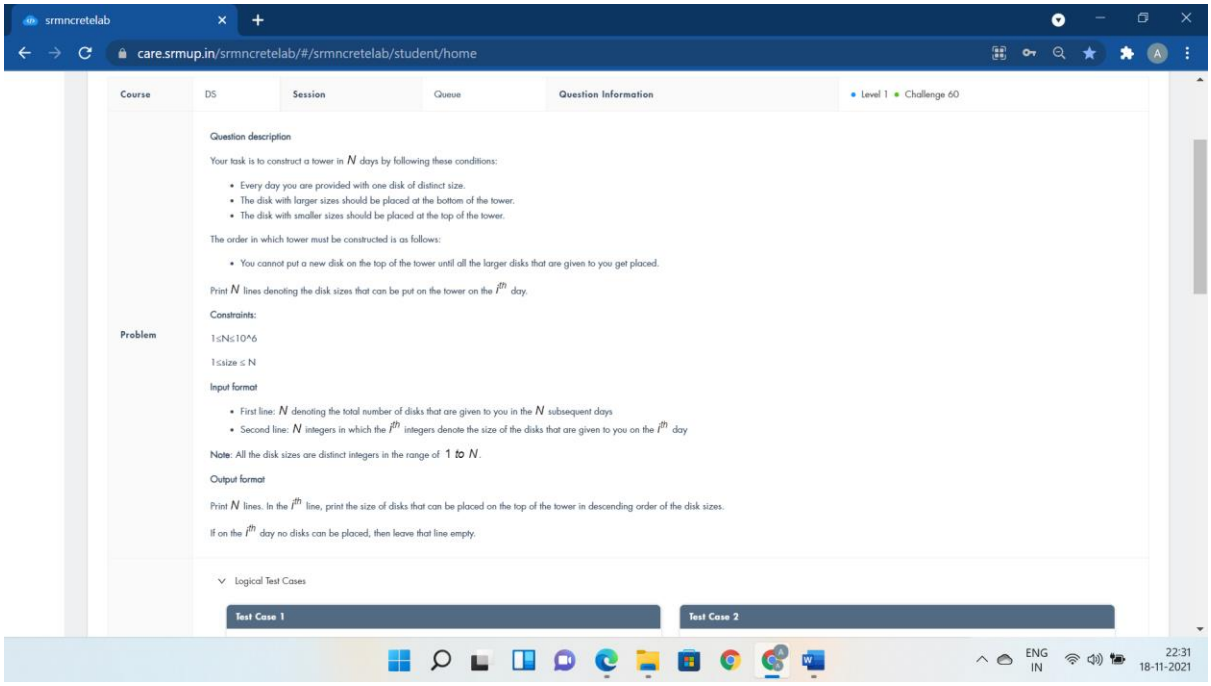
```
void linkedListTraversal(struct node *ptr)
{
   //printf("Printing the elements of this linked list\n");
   while (ptr != NULL)
   {
      printf("%d ", ptr->data);
      ptr = ptr->next;
   }
}

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

int dequeue()
{
   int val = -1;
   struct node* t;
   t = f;
```

```
    if(f==NULL){
        printf("Queue is Empty\n");
    }
    else{
        f = f->next;
        val = t->data;
        free(t);
    }
    return val;
}

int main()
{
    int n,i,t;
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&t);
        enqueue(t);
    }
    linkedListTraversal(f);
    for(i=0;i<2;i++){
        dequeue();
        printf("\n");
        linkedListTraversal(f);
    }
    return 0;
}
```




```
#include <stdio.h>
#include <stdlib.h>
struct node *front = NULL;
struct node *rear = NULL;
struct node
{
   int data;
   struct node *next;
};
void linkedListTraversal(struct node *ptr)
{
   //printf("Printing the elements of this linked list\n");
   while (ptr != NULL)
   {
      printf("%d ", ptr->data);
      ptr = ptr->next;
   }
}
void enqueue(int d)
{
   struct node* new_n;
   new_n = (struct node*)malloc(sizeof(struct node));
```

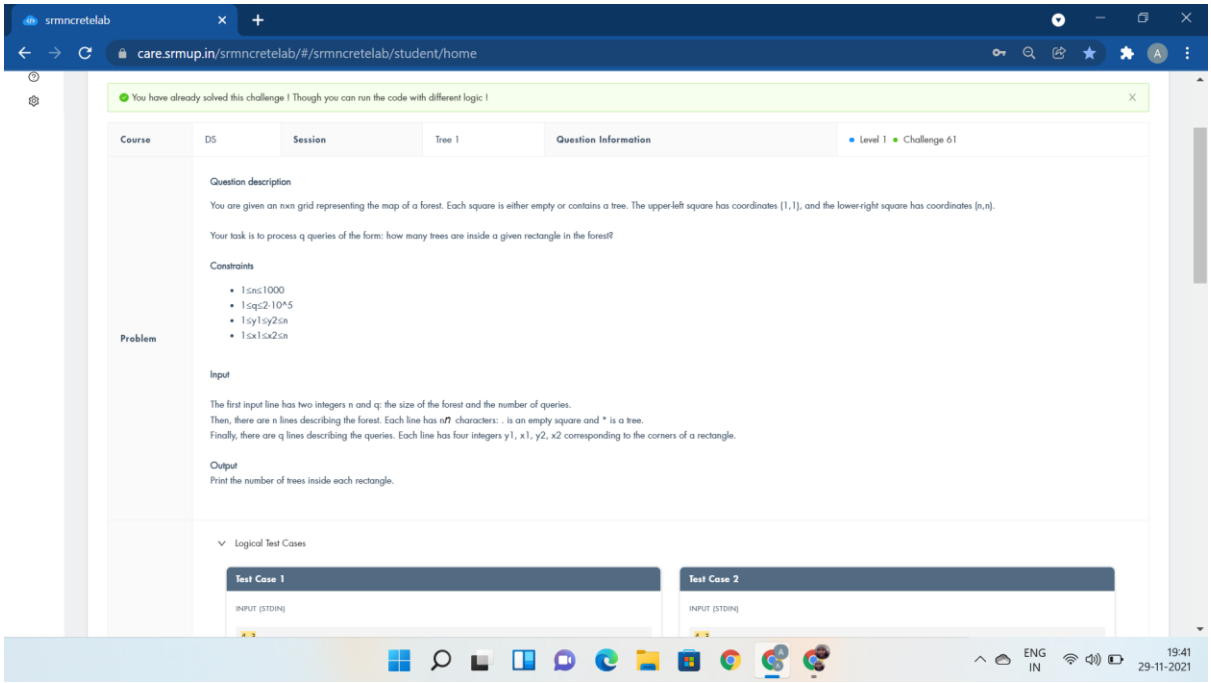
```
    if(new_n==NULL){
        printf("Queue is Full");
    }
    else{
        new_n->data = d;
        new_n->next = NULL;
        if(front==NULL){
            front=rear=new_n;
        }
        else{
            rear->next = new_n;
            rear=new_n;
        }
    }
}
int main()
{
    int n,i,t;
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&t);
        enqueue(t);
    }
    linkedListTraversal(front);
    return 0;
}
```




```
#include<stdio.h>
int main()
{
   long int disk,temp[1000000]={0},size,i,max;
   scanf("%ld",&disk);
   max=disk;
   for(i=0;i<disk;i++)
   {
      scanf("%ld",&size);
      temp[size]=size;
      if(size==max)
      {
         while(temp[size])
         {
            printf("%ld ",temp[size]);
            size--;
         }
         max=size;
         printf("\n");
      }
      else
       printf("\n");
```

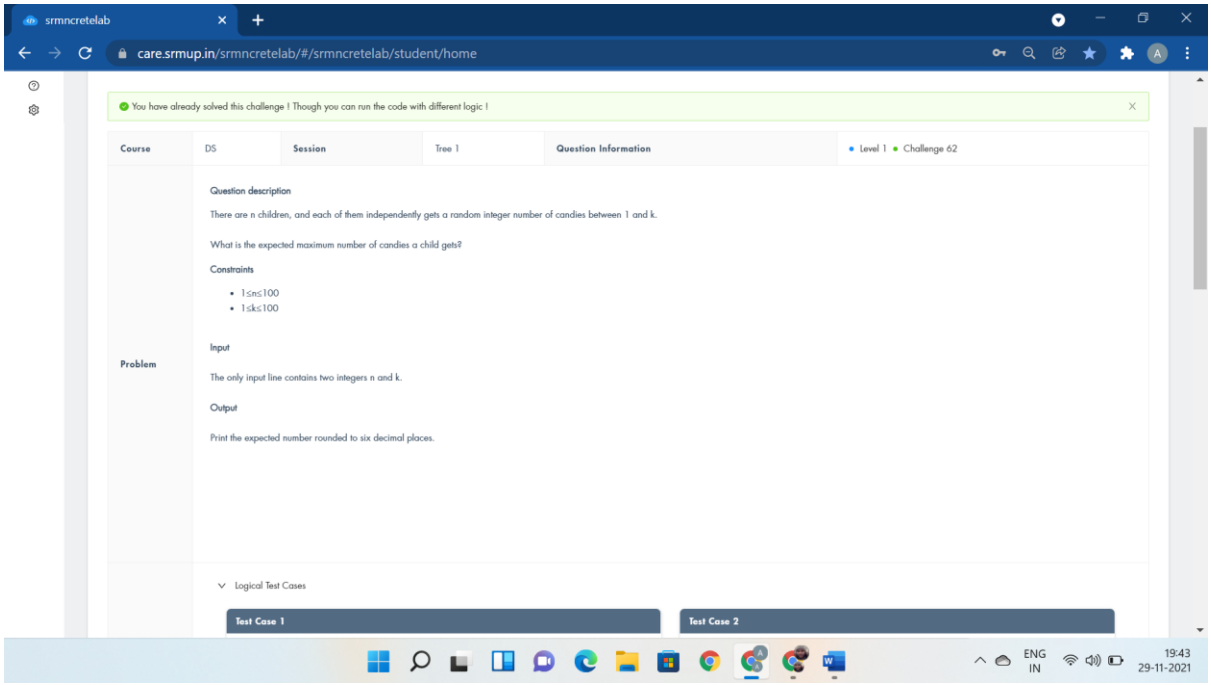
```
    }
    return 0;
}
```

TREE 1:-



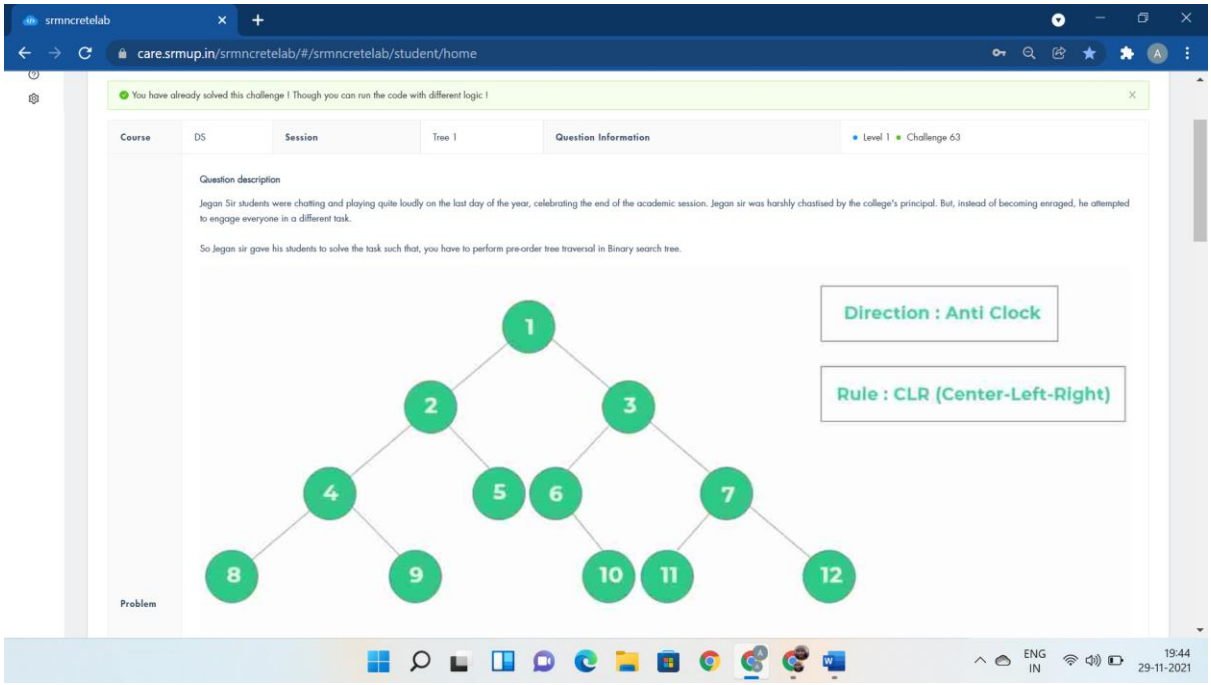

```
#include<bits/stdc++.h>
using namespace std;
#define rep(i,a,b) for (int i=a; i<b; ++i)
int dp[1005][1005];
int main(){
    int n,m; cin>>n>>m;
    rep(i,1,n+1){
        rep(j,1,n+1){
            char x; cin>>x;
            dp[i][j] = (dp[i-1][j] - dp[i-1][j-1]) + dp[i][j-1] + (x=='*');
        }
    }
    while(m--){
        int y1 , x1, y2, x2; cin>>y1>>x1>>y2>>x2;
```

```
        cout<<dp[y2][x2]+ dp[y1-1][x1-1] - dp[y2][x1-1] - dp[y1-1][x2]<<endl;
    }
    return 0;
    cout<<"for(i=1;i<=n;i++)";
}
```



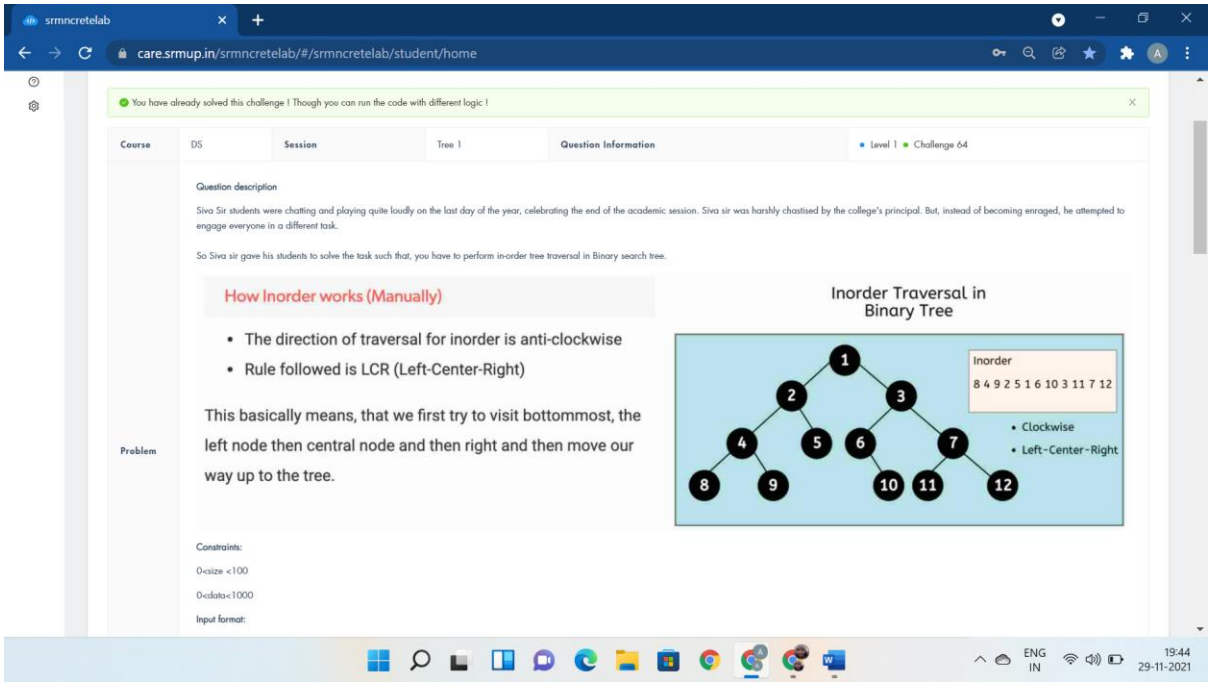

```
#include <bits/stdc++.h>

using namespace std;

int N, K;
double ans, a, b;

int main(){
    scanf("%d %d", &N, &K);
    for(int i = 1; i <= K; i++){
        a = b = 1.0;
        for(int j = 1; j <= N; j++){
            a *= (double) i / K;
            b *= (double) (i-1) / K;
        }
        ans += (a-b) * i;
```

```
    }
    printf("%.6f\n", ans);
    return 0;
    cout<<"double power(double a,int k)";
}
```




```
#include<bits/stdc++.h>
using namespace std;
void solve(){}
struct node {
    int data;
    struct node *left,*right;
}*root=NULL;
void insert(int data) {
    struct node *tempNode = (node*) malloc(sizeof(node));
    struct node *current;
    struct node *parent;
    tempNode->data = data;
    tempNode->left = NULL;
    tempNode->right = NULL;
    if(root == NULL)  root = tempNode;
    else {
```

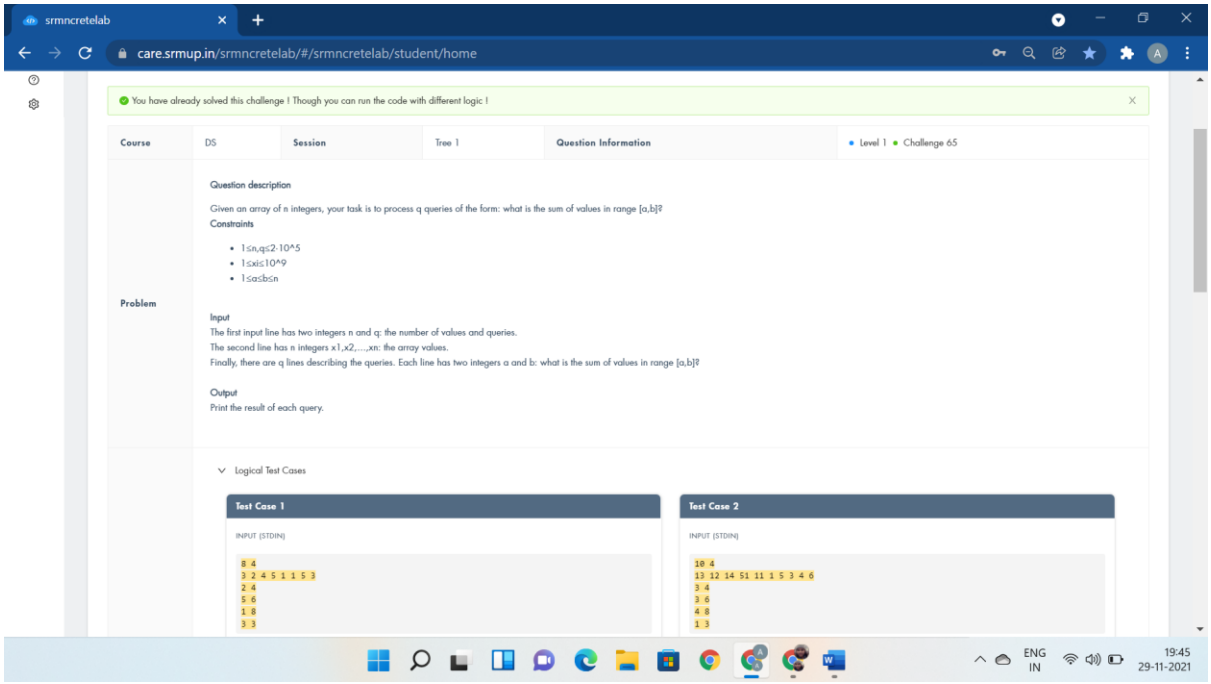
```
        current = root;
        parent = NULL;
        while(1) {
           parent = current;
           if(data < parent->data) {
              current = current->left;
              if(current == NULL) {
                 parent->left = tempNode;
                 return;
              }
           }
           else {
              current = current->right;
              if(current == NULL) {
                 parent->right = tempNode;
                 return;
              }
           }
        } } }
void preorder(struct node* root) {
   if(root != NULL) {
      printf("%d ",root->data);
      preorder(root->left);
      preorder(root->right);
   }
}
int main() {
   solve();
   int n,i,x; scanf("%d",&n);
   for(i = 0; i < n; i++){
      scanf("%d",&x); insert(x); }
   preorder(root);
   return 0;
   printf("struct node* newNode(int item) "); }
```




```
#include <stdio.h>
#include <stdlib.h>
struct node {
  int data;
  struct node *left,*right;
};
void solve(){}
struct node *root = NULL;

void insert(int data) {
  struct node *tempNode = (struct node*) malloc(sizeof(struct node));
  struct node *current;
  struct node *parent;

  tempNode->data = data;
  tempNode->left = NULL;
  tempNode->right = NULL;

  //if tree is empty
  if(root == NULL) {
    root = tempNode;
  } else {
```

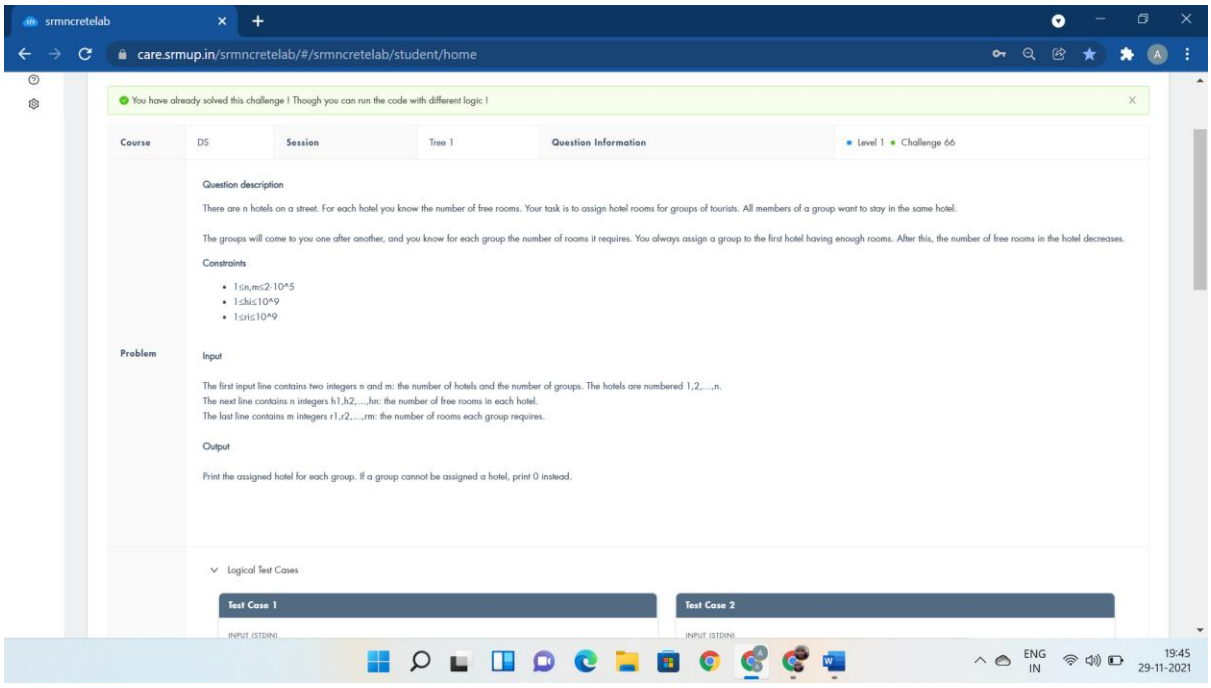
```
    current = root;
    parent = NULL;

    while(1) {
      parent = current;

      //go to left of the tree
      if(data < parent->data) {
        current = current->left;

        //insert to the left
        if(current == NULL) {
          parent->left = tempNode;
          return;
        }
      } //go to right of the tree
      else {
        current = current->right;

        //insert to the right
        if(current == NULL) {
          parent->right = tempNode;
          return;
        }
      }
    }
  }
}



void inorder(struct node* root) {
  if(root != NULL) {
    inorder(root->left);
    printf("%d ",root->data);
    inorder(root->right);
```

```
  }
}

int main() {
  solve();
  int n,i;
  scanf("%d",&n);
  int array[n];
  for(i=0;i<n;i++)
  scanf("%d",&array[i]);

  for(i = 0; i < n; i++)
    insert(array[i]);
  inorder(root);
  return 0;
  printf("temp->left=temp->right=NULL; struct node* newNode(int item)");
  return 0;
}
```



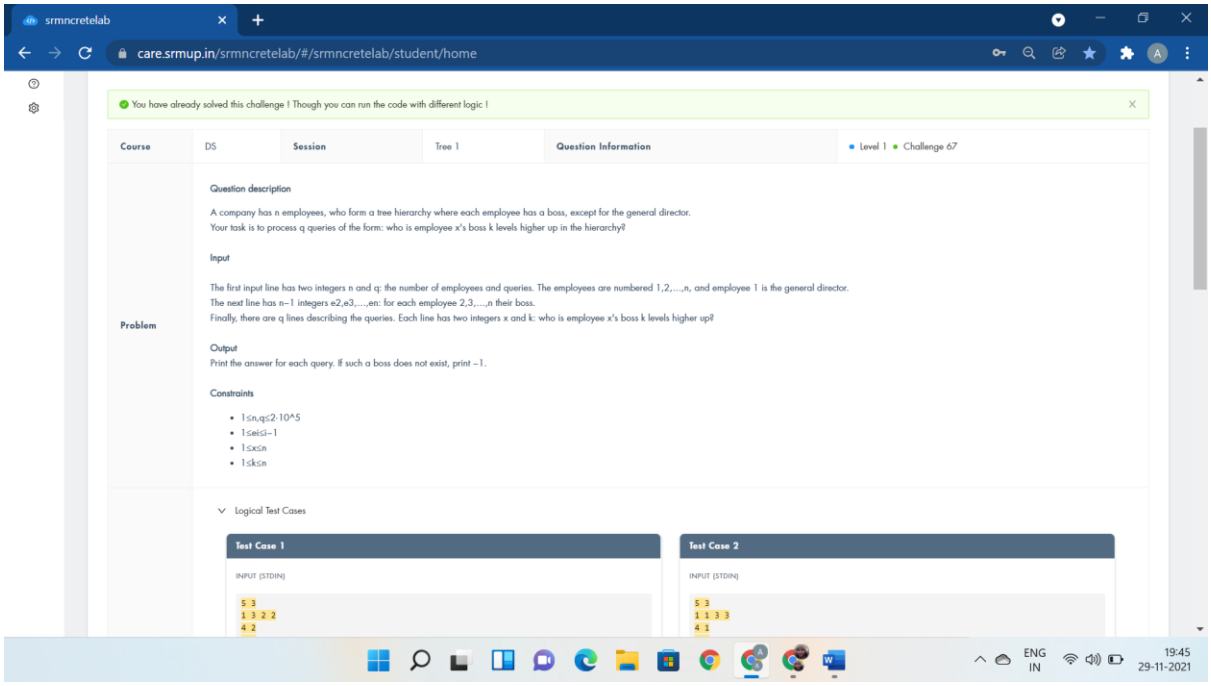

```
#include<bits/stdc++.h>
using namespace std;
int main(){
```

```
    int n,q,i,a,b;
    cin>>n>>q;
    int x[n];
    for(i=0;i<n;i++)
    cin>>x[i];
    while(q--){
        int sum=0;
        cin>>a>>b;

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



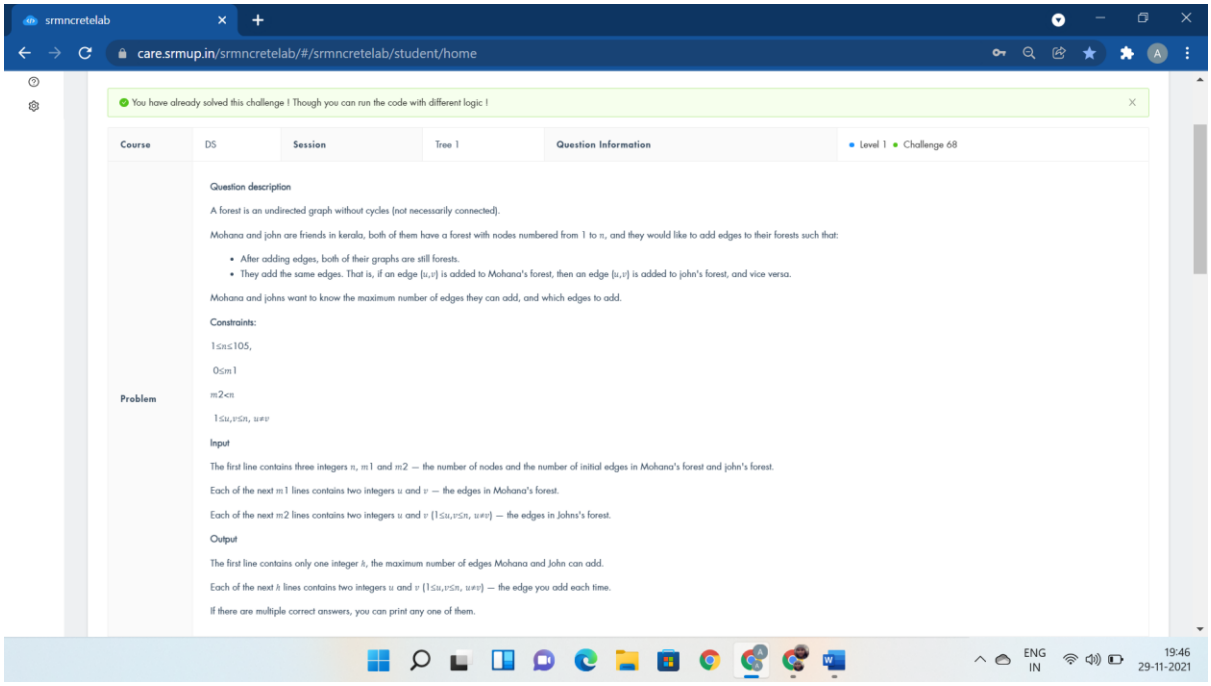

```
#include<iostream>
using namespace std;
void solve(){}
int main()
{
    solve();
    int n,m,i;
```

```
    cin>>n>>m;
    int a[n],b[n];
    for(i=0;i<n;i++)
    cin>>a[i];
    for(i=0;i<n;i++)
    cin>>b[i];
    for(i=0;i<m;i++){
        int f=0,j=0;
        for(;j<n;j++){
            if(a[j]>=b[i]){
                a[j]-=b[i];
                f=1;
                break;
            }
        }
        if(f>0)
        cout<<j+1<<" ";
        else
        cout<<"0 ";
    }
                return 0;
}
```



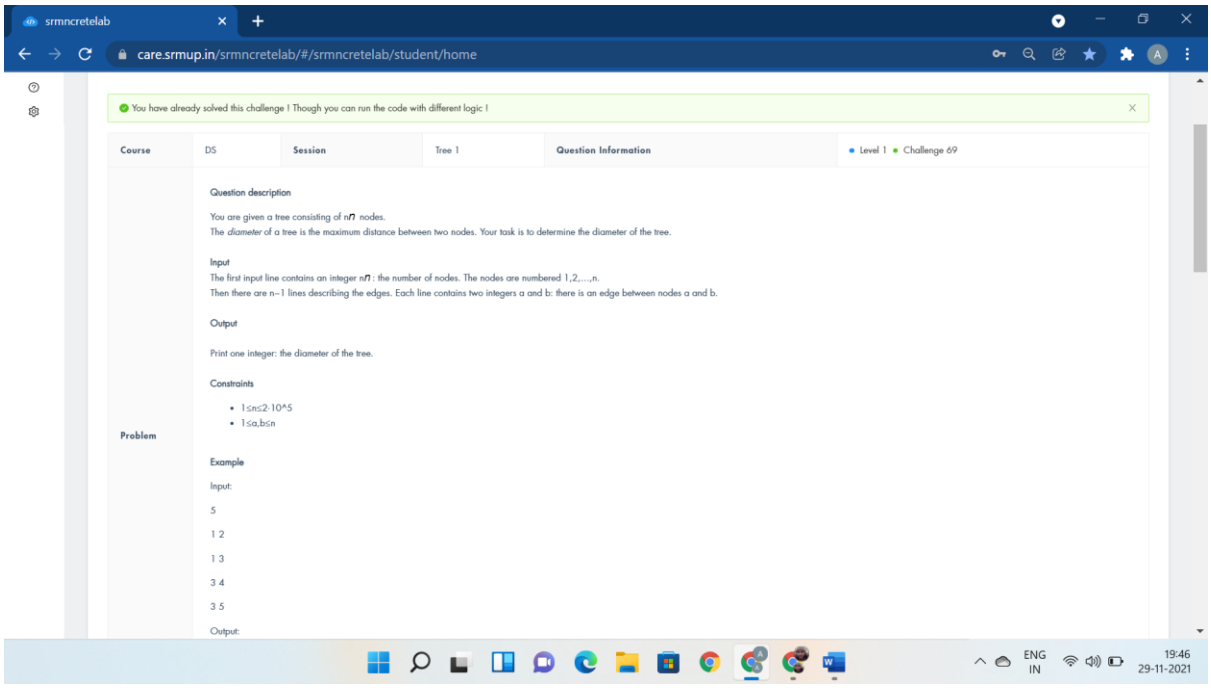

```
#include<bits/stdc++.h>
using namespace std;
#define   ll     long long
#define   MAX    200005
#define   pb     push_back
vector<int>tree[MAX];
ll up[MAX][20];
void solve(){}
void link(int i,int j){
   up[i][0]=j;
   for(int m=1;m<20;m++){
      if(up[i][m-1]!=-1)
      up[i][m]=up[up[i][m-1]][m-1];
      else
      up[i][m]=-1;
   }
   for(auto child:tree[i]){
      if(child!=j) link(child,i);
   }
}
int ans_query(int src,int jump){
   if(src==-1 or  jump==0)return src;
```

```
    for(int i=19;i>=0;i--){
        if( jump>= (1<<i)){
            return ans_query(up[src][i],jump-(1<<i));
        }
    }
    return 1;
}
int main(){
    solve();
    int n,q;
    cin>>n>>q;
    for(int i=2;i<=n;i++){
        int ee;
        cin>>ee;
        tree[i].pb(ee);
        tree[ee].pb(i);
    }
    link(1,-1);
    while(q--){
        int node,jump;
        cin>>node>>jump;
        cout<<ans_query(node,jump)<<endl;
    }
}
```



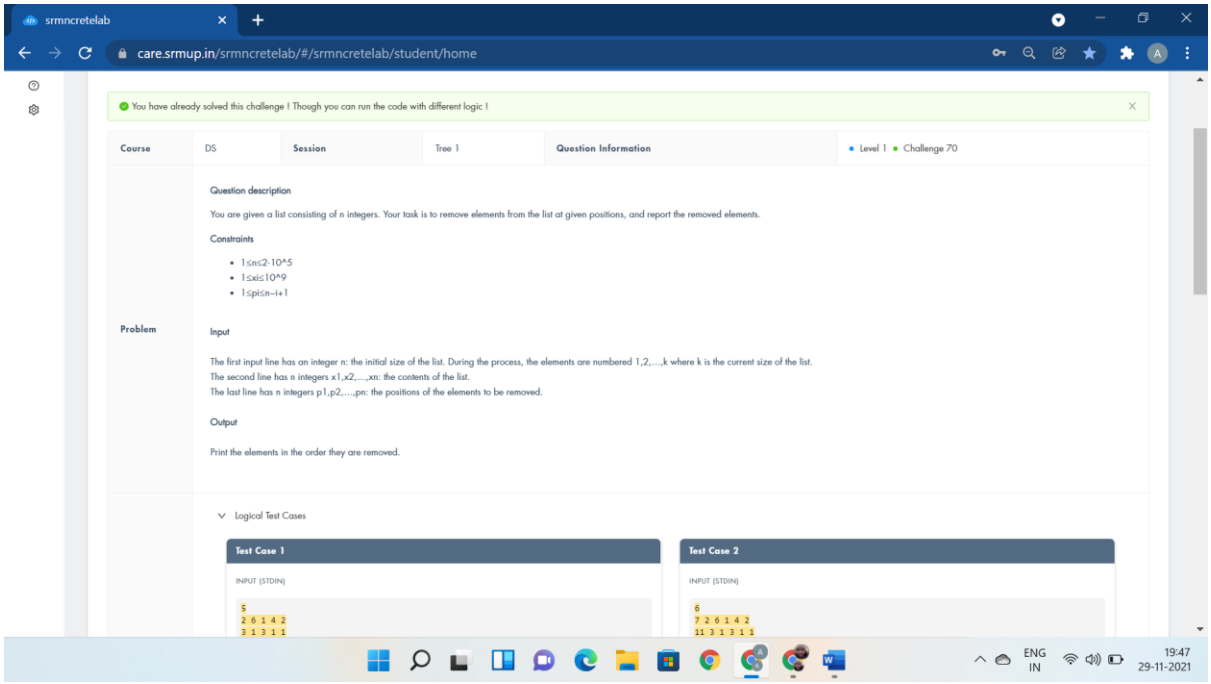

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int mod=998244353;
int fa[1005],fa2[1005],n,m1,m2;
int gf(int x,int *f){
	return f[x]==x?x:f[x]=gf(f[x],f);
}
int main(){
	cin>>n>>m1>>m2;
	for(int i=1;i<=n;i++)fa[i]=fa2[i]=i;
	for(int i=1,x,y;i<=m1;i++)cin>>x>>y,fa[gf(x,fa)]=gf(y,fa);
	for(int i=1,x,y;i<=m2;i++)cin>>x>>y,fa2[gf(x,fa2)]=gf(y,fa2);
	cout<<n-max(m1,m2)-1<<'\n';
	for(int i=1;i<=n;i++){
		for(int j=i+1;j<=n;j++){
			if(gf(i,fa)!=gf(j,fa)&&gf(i,fa2)!=gf(j,fa2)){
				cout<<i<<' '<<j<<'\n';
				fa[gf(i,fa)]=gf(j,fa);
				fa2[gf(i,fa2)]=gf(j,fa2);
			}
		}
```

```
        }
        return 0;
        cout<<"while(m1--)";
}
```




```
#include<bits/stdc++.h>
using namespace std;
#define vi vector<int>
#define rep(i,a,b) for (int i=a; i<b; ++i)
#define pb push_back
vi adj[200005];
int d=0,x=0;
void solve(){}
void dfs(int s, int p, int dep){
   for (auto i: adj[s]){
      if (i!=p){
         dfs(i,s,dep+1);
      }
   }
   if (dep>d) d = dep, x = s;
}
int main(){
```

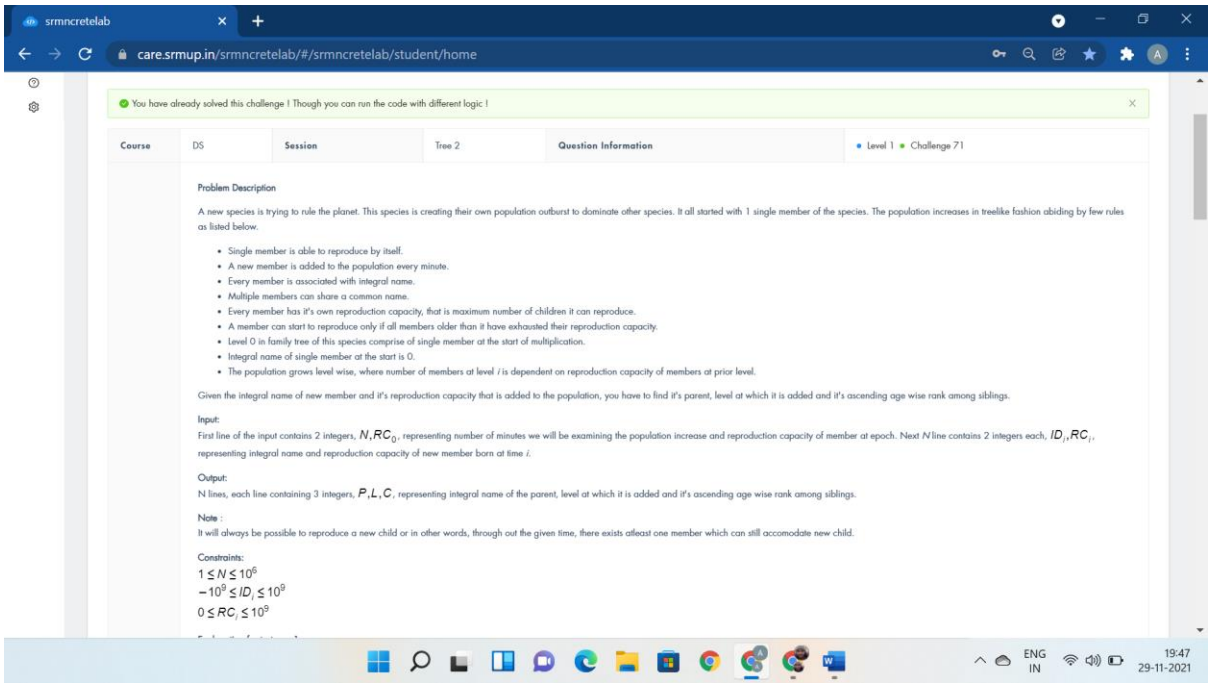
```
    solve();
    int n;
    cin>>n;;
    rep(i,0,n-1){
        int x,y; cin>>x>>y;
        adj[x].pb(y), adj[y].pb(x);
    }
    dfs(1,0,0);
    dfs(x,0,0);
    cout<<d;
    return 0;
    cout<<"void link(int i,int j) void dfs(int p,int i,int d)";
}
```




```
#include <stdio.h>
#define N   200000
#define N_   (1 << 18)
int tr[N_ * 2];

void build(int k,int l,int r) {
    tr[k] = r - l;
    if (r - l > 1) {
```

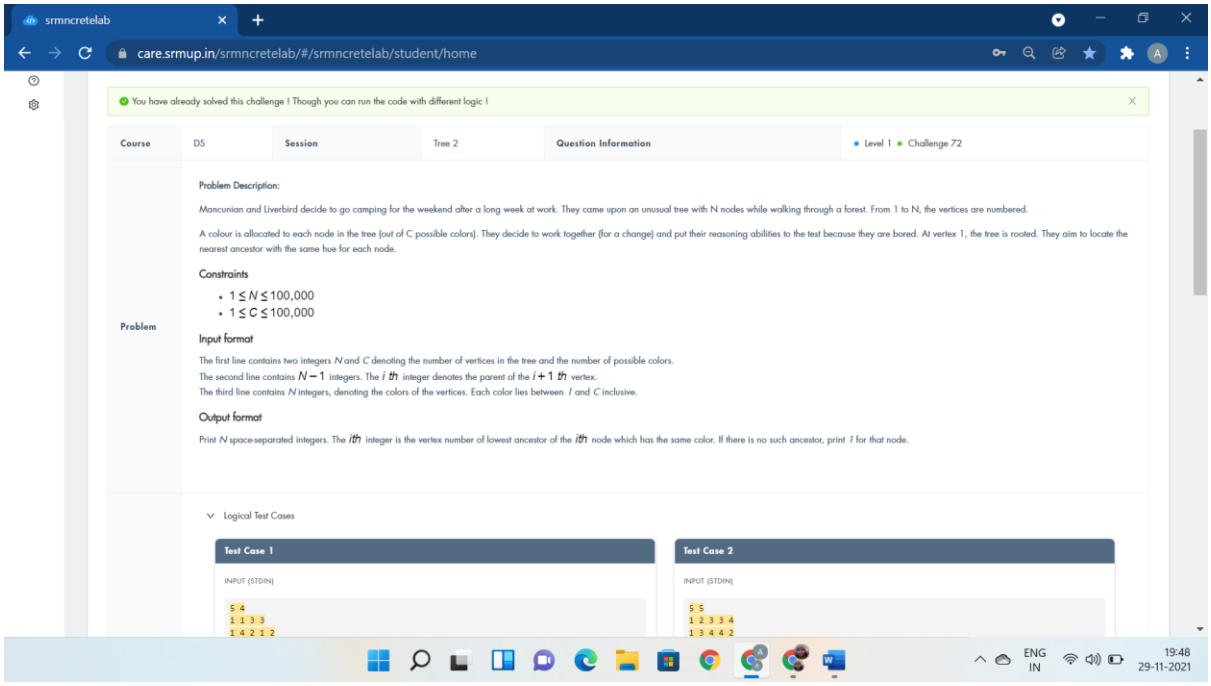
```
        int m = (l + r) / 2;


        build(k * 2 + 1, l, m);
        build(k * 2 + 2, m, r);
    }
}
int query(int k, int l, int r, int x) {
    int m, k1, k2;


    tr[k]--;
    if (r - l == 1)
        return r;
    m = (l + r) / 2, k1 = k * 2 + 1, k2 = k * 2 + 2;
    return tr[k1] >= x ? query(k1, l, m, x) : query(k2, m, r, x - tr[k1]);
}


int main() {


    int n, h, i, x;


    scanf("%d", &n);
    int aa[n];
    for (i = 0; i < n; i++)
        scanf("%d", &aa[i]);
    build(0, 0, n);
    for (h = 0; h < n; h++) {
        scanf("%d", &x);
        i = query(0, 0, n, x) - 1;
        printf("%d ", aa[i]);
    }
    printf("\n");
    return 0;
}
```

TREE-2:-

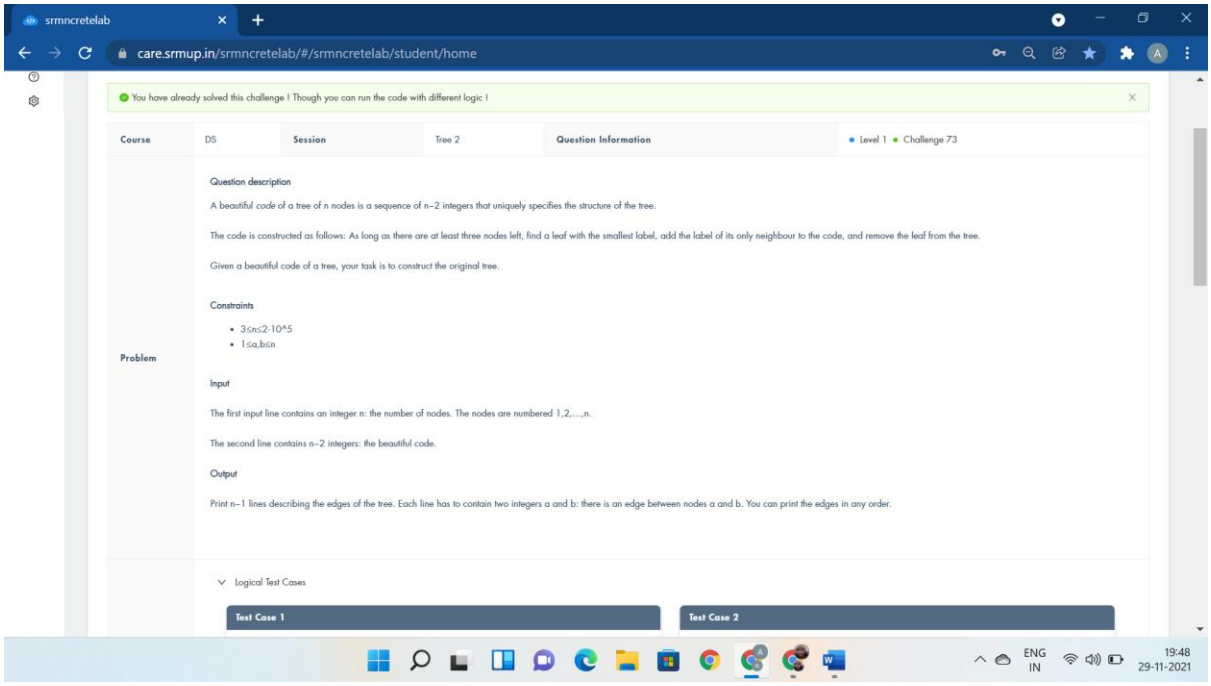



```
#include<stdio.h>

#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;
}
```

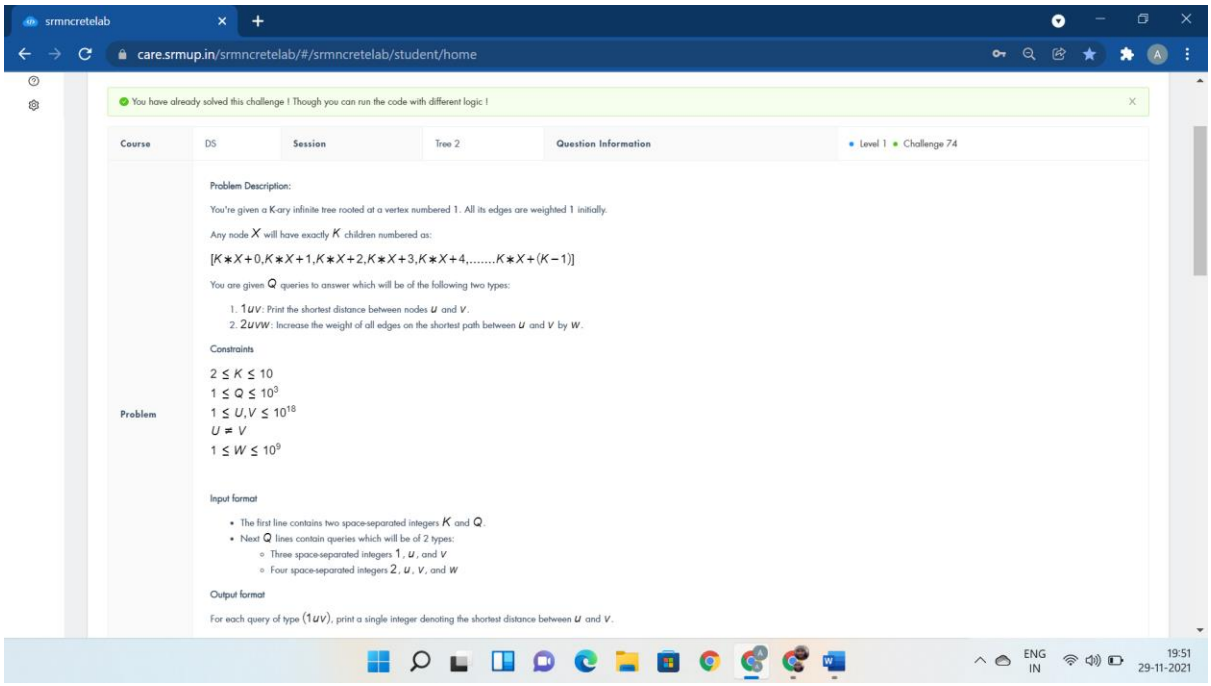



```
#include<bits/stdc++.h>

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);
}
```

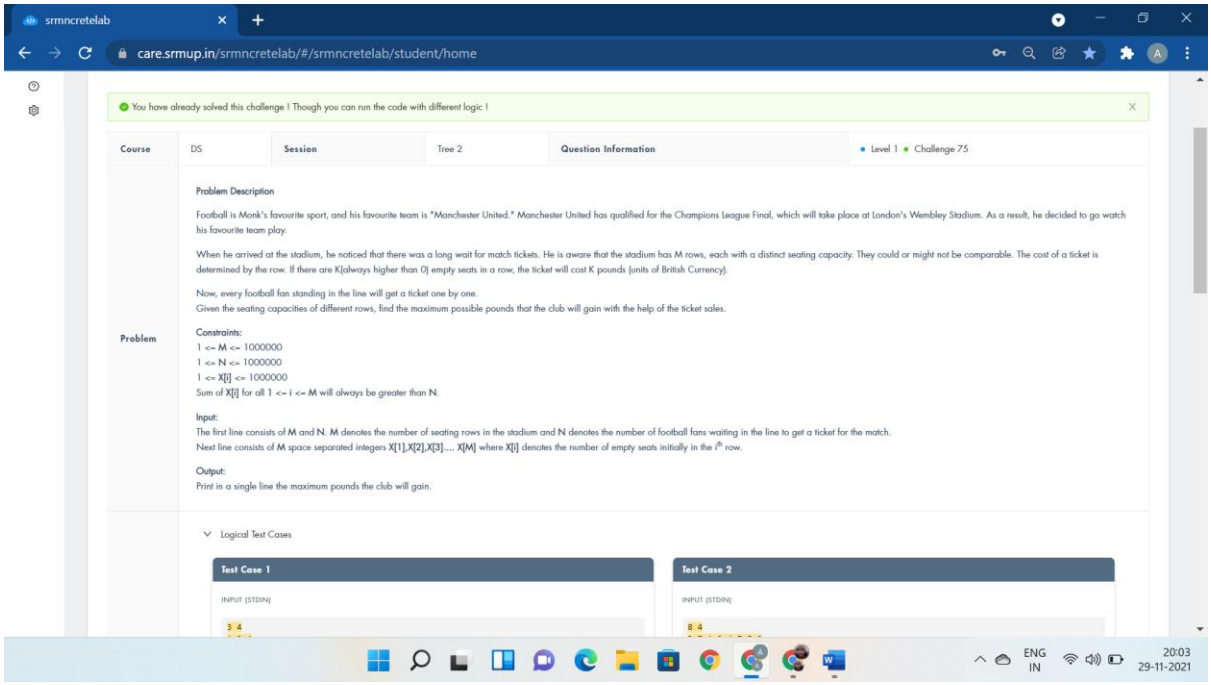



```
#include <iostream>

#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 } ) ) {
```

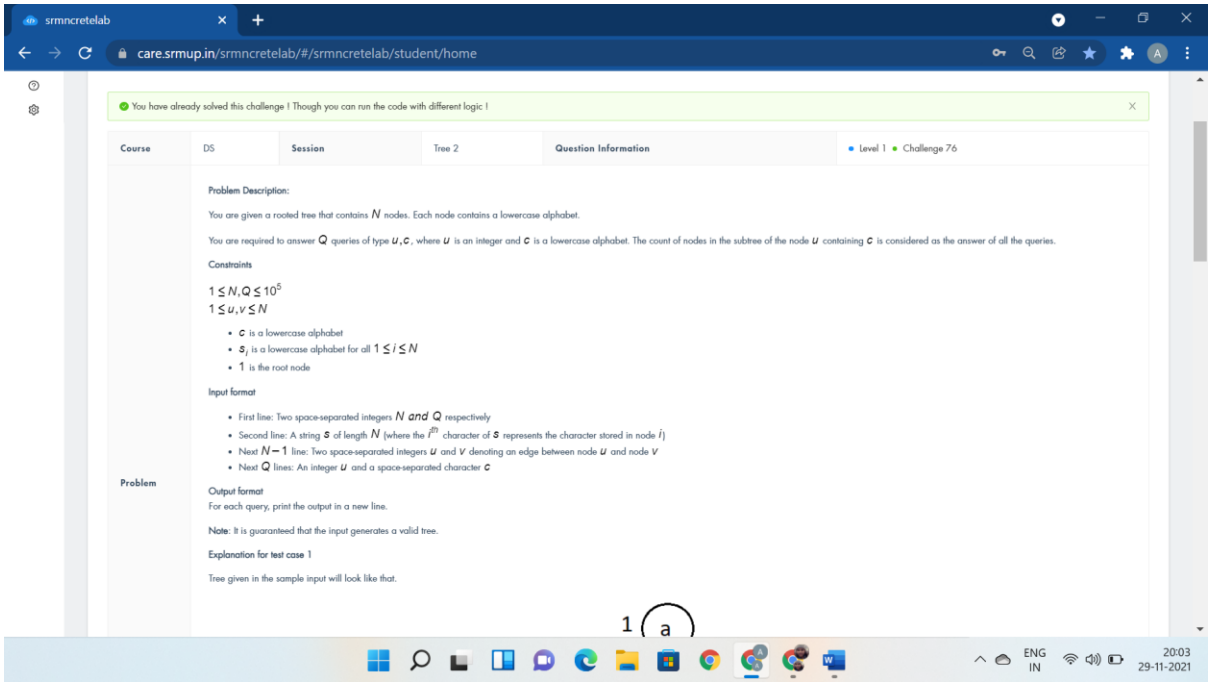
```
            dist = dist + adj [ { v, v / k } ];
        } else {
            dist = dist + 1;
        }
        u = u / k;
        v = v / k;
    }
    return dist;
}

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()
{
```

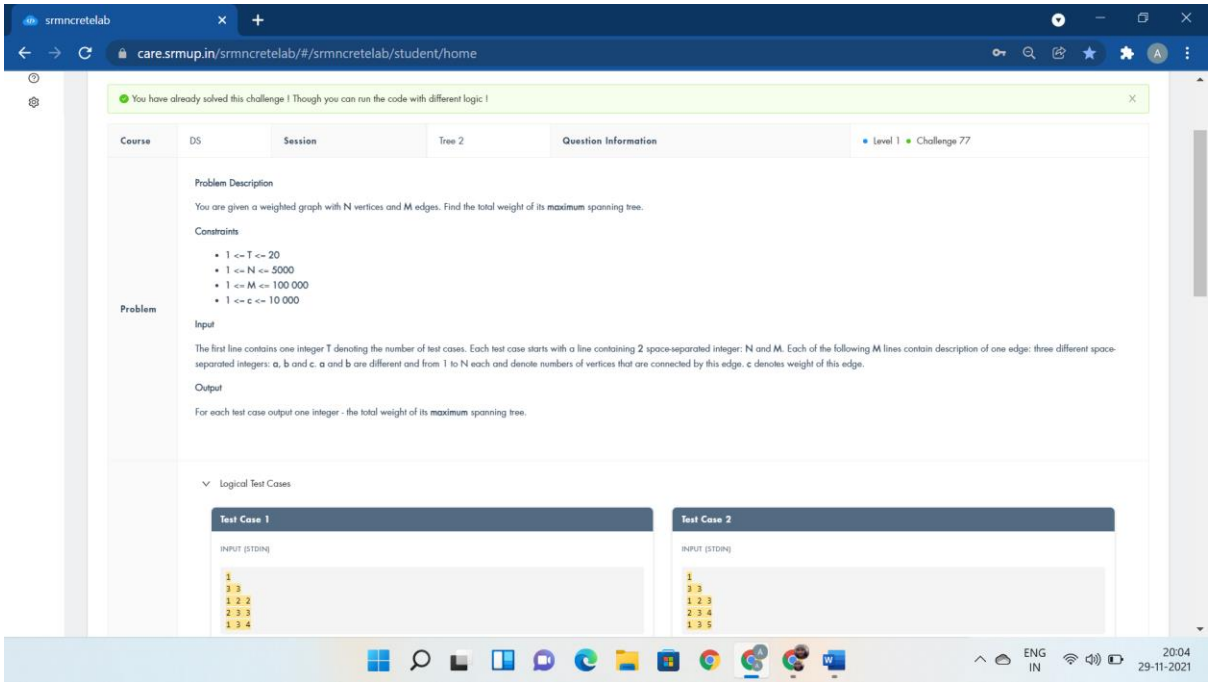
```
    int N, M; cin >> N >> M;
    assert (1<=N and N<=1000000);
    assert (1<=M and M<=1000000);
    for (int g=1; g<=N; g++){
        int a; cin >> a;
        seats.push(a);
        assert (1<=a and a<=1000000);
        x[a]++;
    }
    long long ans = 0;
    for (int g=0; g<M; g++){
        int 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--)
```

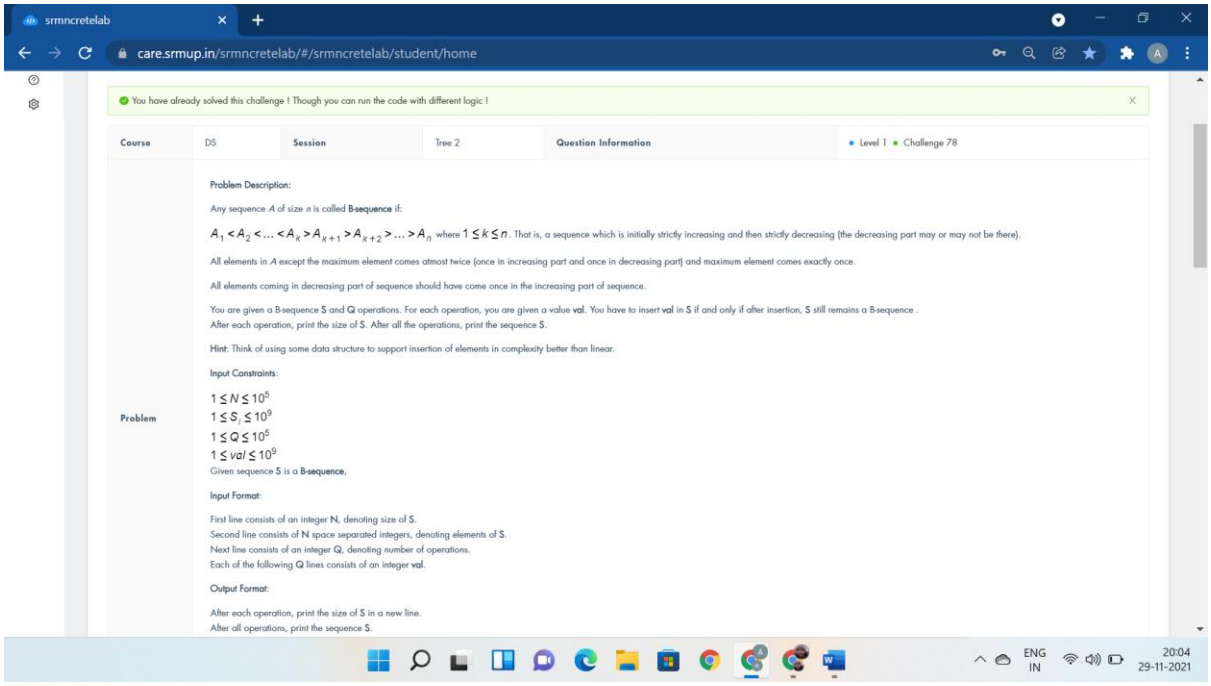
```
        {
                int u;
                char c;
                cin >> u >> c;
                cout<<subroot[u][c-'a']<<"\n";
                //cout<<cnt<<endl;


        }
        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;
         }
```

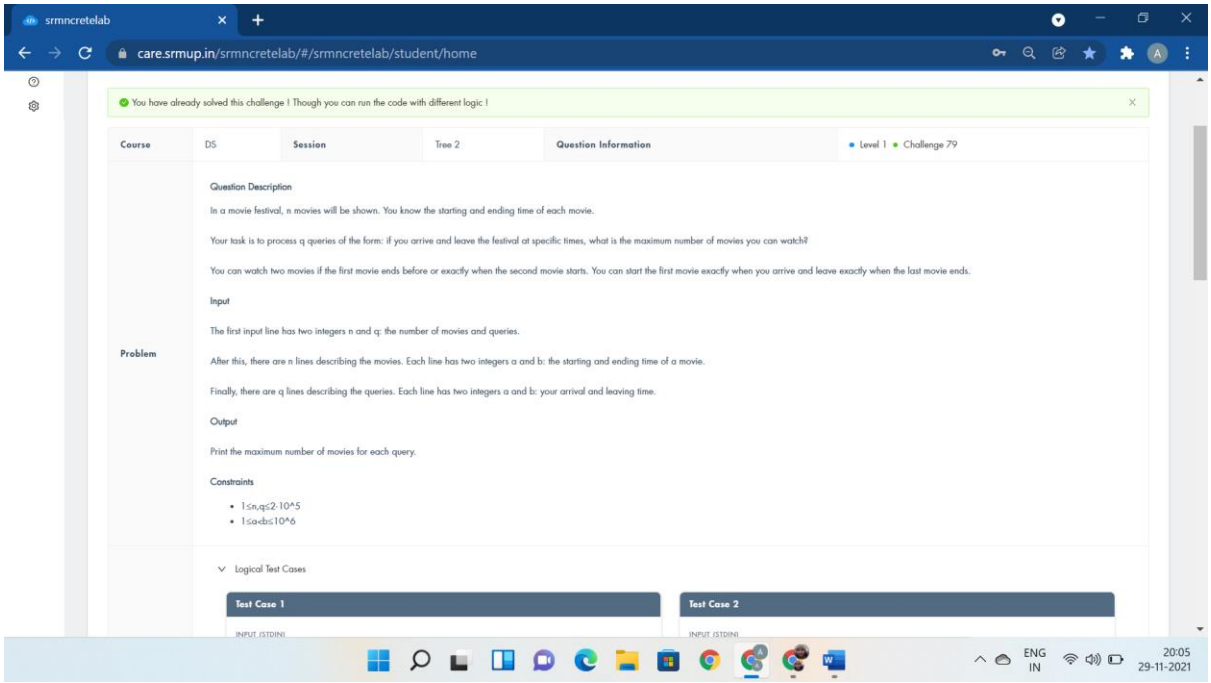
```
    }
    cout<<ans<<endl;
  }
  return 0;
  cout<<"int printheap(int N)";
}
```




```
#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;
```

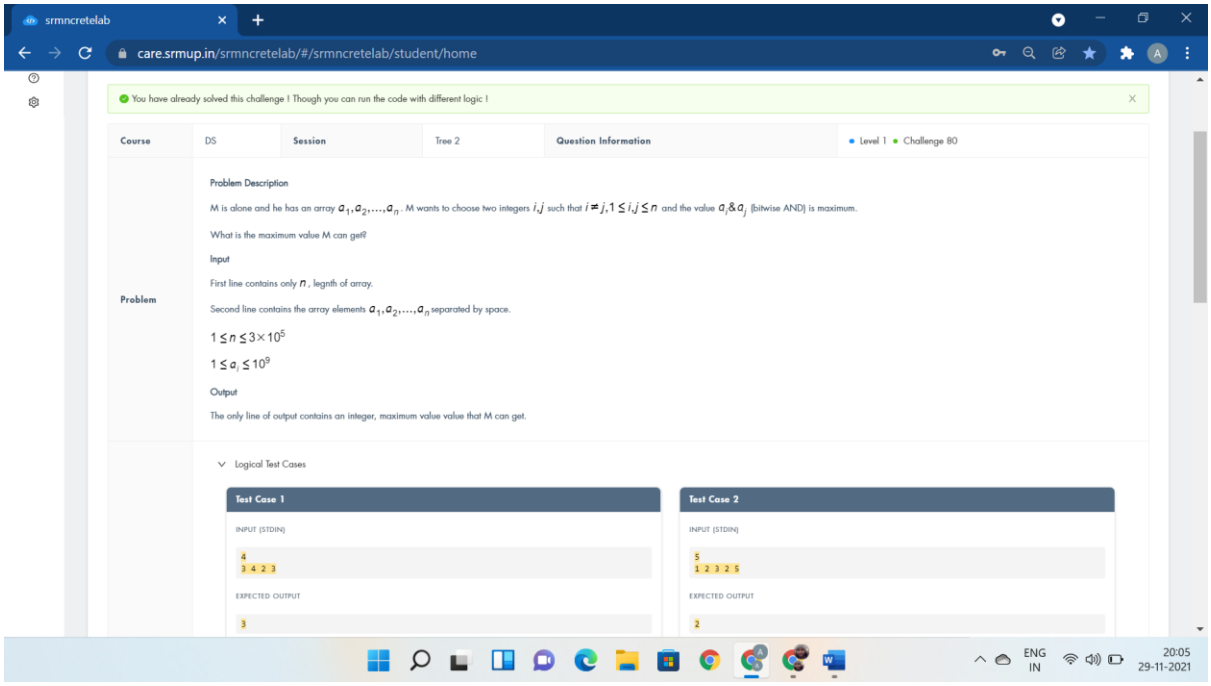
```
for(i=0;i<N;i++) {
    scanf("%d", &S[i]);
    maximum=max(maximum,S[i]);
    map[S[i]]++;
}
int temp,Q;
cin>>Q;
for(i=0;i<Q;i++) {
    scanf("%d", &temp);
    if(temp==maximum) printf("%d\n",N);
    else {
        if(map[temp]>=2) printf("%d\n",N);
        else {
            map[temp]++;
            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;
```

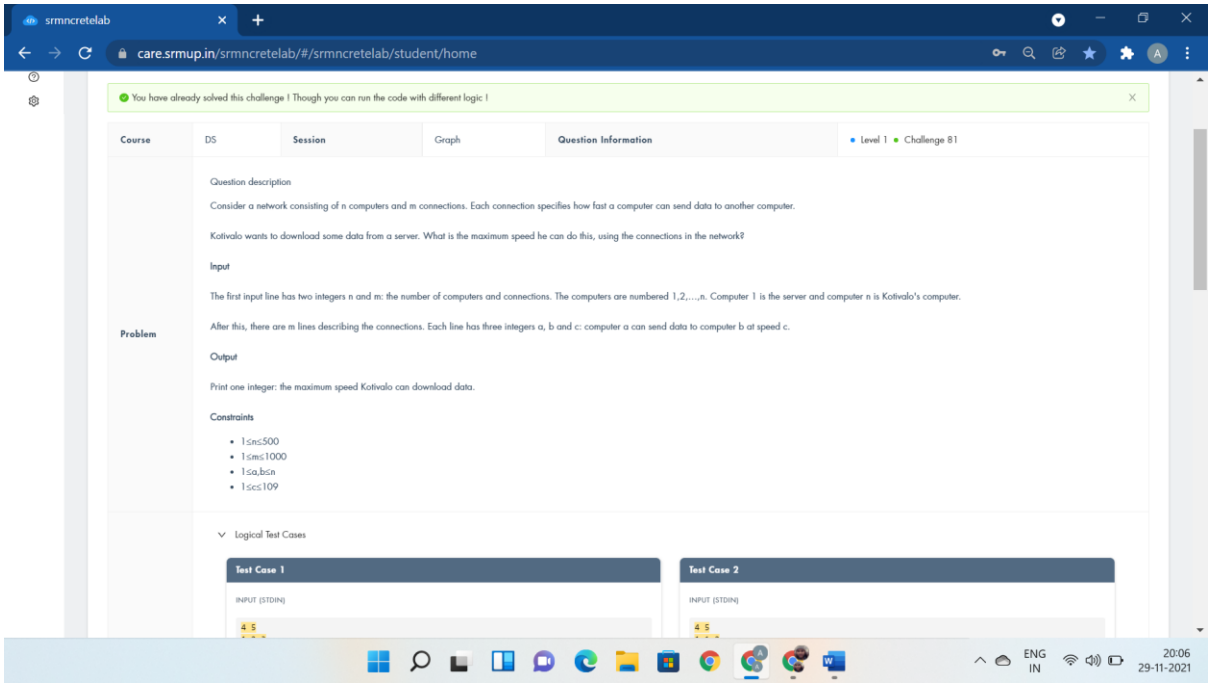
```
        dp[y][0] = max(dp[y][0], x);
    }
    for (int i = 1; i <= 1000000; i++)
        dp[i][0] = max(dp[i][0], dp[i-1][0]);

    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);
}
```

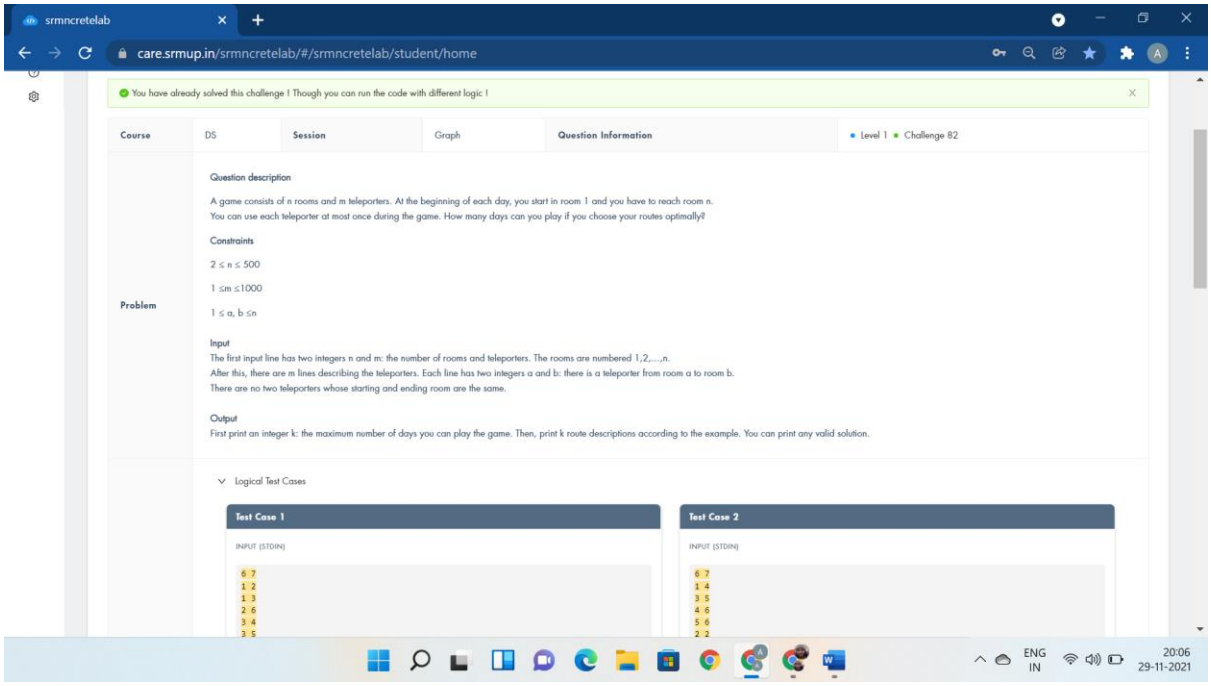
GRAPH:-




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

using namespace std;
```

```
using ll = long long;
#define F0R(i,a) for(int i=0; i<(a); i++)
#define FOR(i,a,b) for(int i=(a); i<=(b); i++)
int n, m;
ll adj[501][501], oadj[501][501];
ll flow[501];
bool V[501];
int pa[501];
void link(int i,int h){}
int bfs(int n,int s,int t){return 1;}
bool reachable() {
	memset(V, false, sizeof(V));
	queue<int> Q; Q.push(1); V[1]=1;
	while(!Q.empty()) {
		int i=Q.front(); Q.pop();
		FOR(j,1,n) if (adj[i][j] && !V[j])
			V[j]=1, pa[j]=i, Q.push(j);
	}
	return V[n];
}

int main() {
  bfs(1,1,1);
  link(1,1);
	cin >> n >> m;
	FOR(i,1,n) FOR(j,1,n) adj[i][j] = 0;
	F0R(i,m) {
		ll a,b,c; cin >> a >> b >> c;
		adj[a][b] += c;
	}
	int v, u;
	ll maxflow = 0;
	while(reachable()) {
		ll flow = 1e18;
		for (v=n; v!=1; v=pa[v]) {
			u = pa[v];
			flow = min(flow, adj[u][v]);
		}
```

```
                maxflow += flow;
                for (v=n; v!=1; v=pa[v]) {
                        u = pa[v];
                        adj[u][v] -= flow;
                        adj[v][u] += flow;
                }
        }
        cout << maxflow << '\n';
}
```

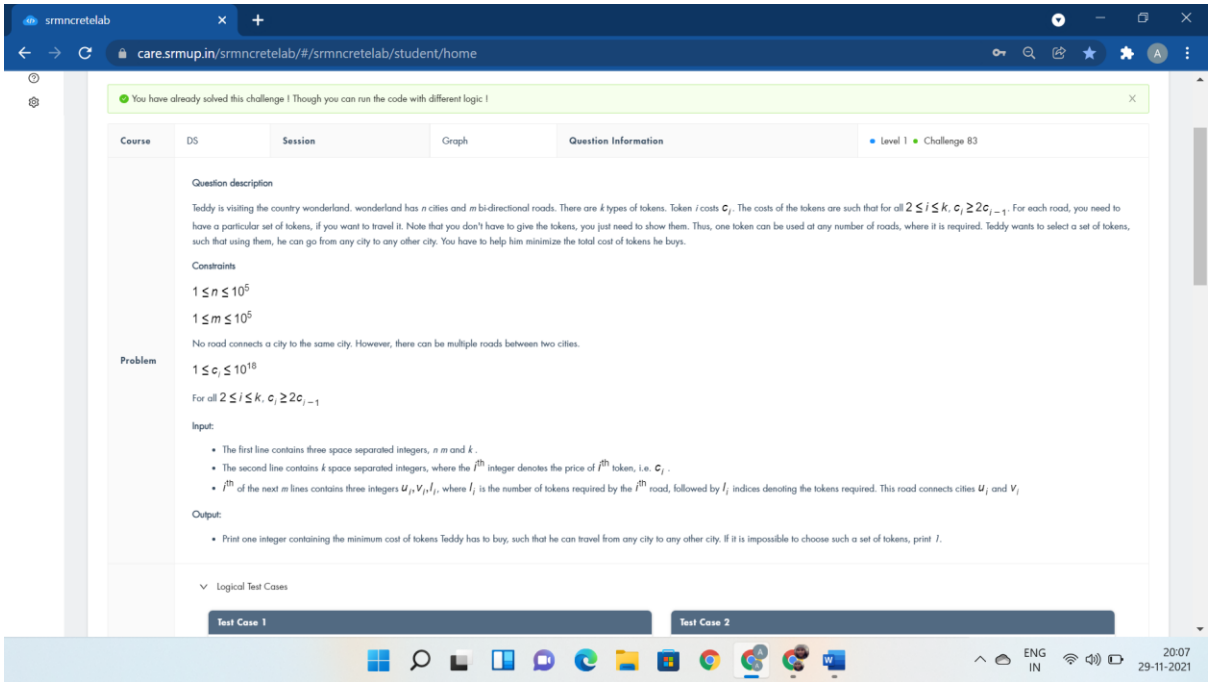



```
#include <stdio.h>

#define N    500
#define M    1000

struct L {
   struct L *next;
   int h;
} aa[N * 2];

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

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

```
    static int qq[N * 2];
    int head, cnt, h, i, j, d;

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

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

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

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

```
            cc[h]--, cc[h ^ 1]++;
            return 1;
          }
        }
    dd[i] = n;
    return 0;
}

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

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

void link(int i, int j, int h, int c) {
    static struct L l91[(M + N) * 2], *l = l91;

    ij[h] = i ^ j;
    cc[h << 1] = c;
    l->h = h << 1;
    l->next = aa[i].next, aa[i].next = l++;
    l->h = h << 1 ^ 1;
    l->next = aa[j].next, aa[j].next = l++;
}

int qq[N];

int path(int i, int t) {
    int cnt = 0;

    while (i != t) {
        struct L *l;
        int h;

        qq[cnt++] = i;
```

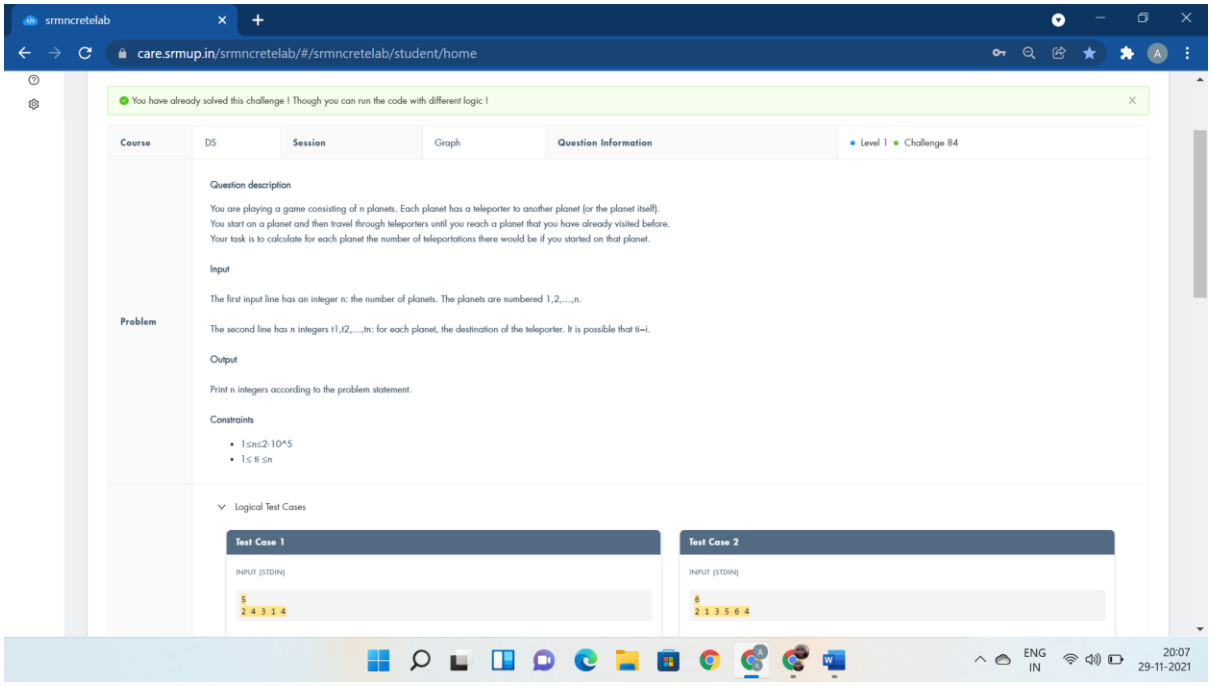
```
        for (l = aa[i].next; l; l = l->next)
            if (((h = l->h) & 1) == 0 && cc[h ^ 1]) {
                cc[h]++, cc[h ^ 1]--;
                i ^= ij[h >> 1];
                break;
            }
    }
    qq[cnt++] = t;
    return cnt;
}

int main() {
    int n, m, h, i, j, k, s, t, cnt;

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




```
#include <stdio.h>

#if defined( _WIN32 )
typedef __int64 az_int64_t;
typedef unsigned __int64 az_uint64_t;
#define I64(x) x ## I64
#define F64 "I64"
#else
typedef long long az_int64_t;
typedef unsigned long long az_uint64_t;
#define I64(x) x ## ll
#define F64 "ll"
#endif

#define MAXN (100*1024)

struct link
{
 az_int64_t t;
 int u, v;
};

struct link links[MAXN];
```

```
int n, m, k;
az_int64_t c[64];

int gr[MAXN];

int getgr( int g )
{
  return (g == gr[g]) ? g : (gr[g] = getgr( gr[g] ));
}

int test( az_int64_t r )
{
  int i, left = n-1, u, v;
  for(i=1;i<=n;++i) gr[i] = i;
  for( i = 0; i < m; ++i)
    if( (links[i].t & r) == 0 &&
        (u = getgr( links[i].u )) != (v = getgr( links[i].v )) )
    {
      gr[v] = u;
      if( --left == 0 ) return 1;
    }
  return 0;
}

int main( void )
{
  az_int64_t rejected = 0, sum = 0;
  int i;

  scanf( "%d %d %d", &n, &m, &k);
  for( i = 0; i < k; ++i) scanf( "%" F64 "d", &c[i]);
  for( i = 0; i < m; ++i)
  {
    int l, id;
    scanf( "%d %d %d", &links[i].u, &links[i].v, &l);
    while( l-- > 0 )
    {
      scanf( "%d", &id);
```

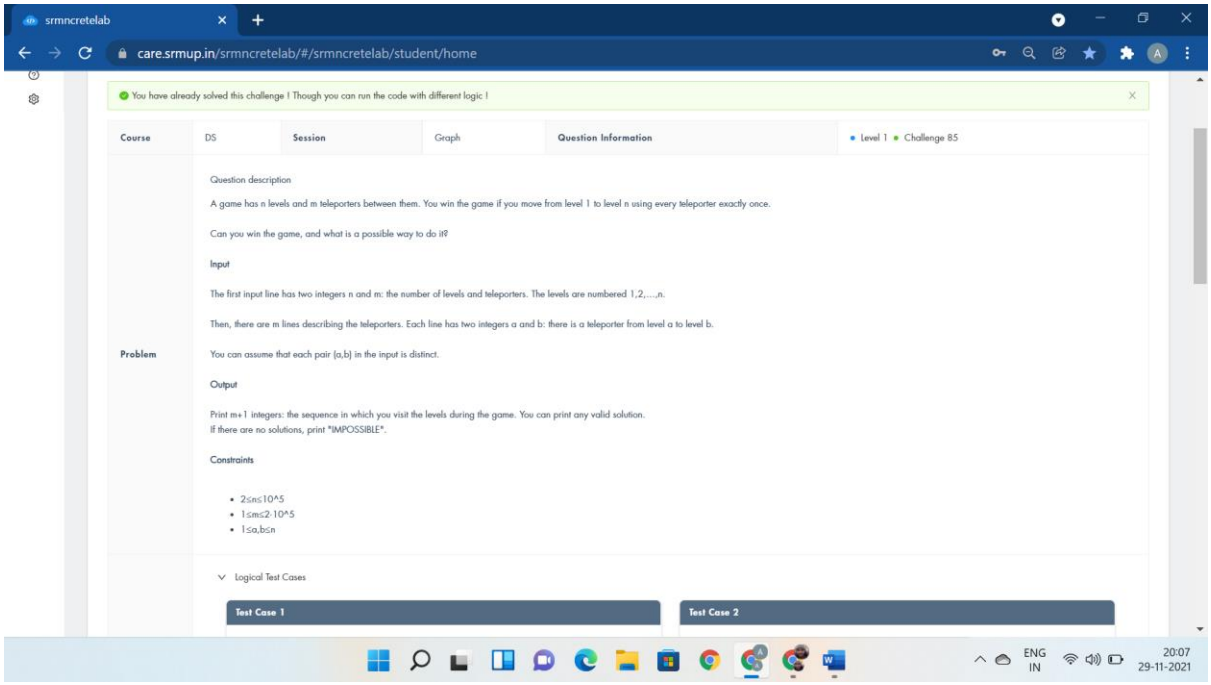
```
    links[i].t |= I64(1) << (id-1);
  }
 }

 if( !test( 0 ) )
 {
  printf( "-1\n" );
  return 0;
 }

 for( i = k-1; i >= 0; --i)
 {
  az_int64_t f = I64(1) << i;
  if( test( rejected | f ) ) rejected |= f; else sum += c[i];
 }

 printf( "%" F64 "d\n", sum);
 return 0;
}
```




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

```
#define N    200000

int main() {
    static int aa[N], cc[N], dd[N], qq[N];
    int n, i, j, c, d, q, cnt;

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

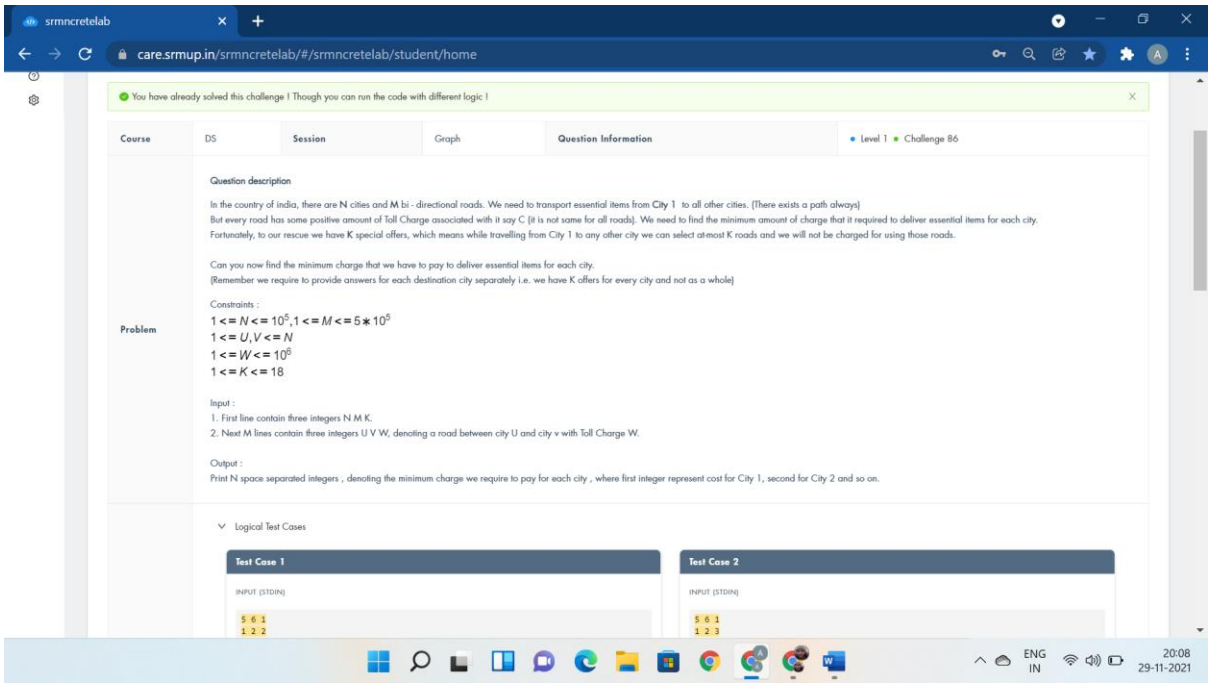
```
            cc[j] = c;
            dd[j] = d--;
            j = aa[j];
        }
    }
    for (i = 0; i < n; i++)
        printf("%d ", dd[i] + qq[cc[i]]);
    printf("\n");
    return 0;
}
```




```
#include <stdio.h>

#define N    100000
#define M    200000

struct L {
    struct L *next;
    int j;
} *aa[N];

struct L *new_L(int j) {
    static struct L l91[M + 1 + M], *l = l91;
```

```
    l->j = j;
    return l++;
}

void link(int i,int j) {
    struct L *l = new_L(j);

    l->next = aa[i]; aa[i] = l;
}

void hierholzer(struct L *e) {
    struct L *f = e->next, *l;
    int i = e->j;

    while ((l = aa[i])) {
        aa[i] = l->next;
        e = e->next = new_L(l->j);
        i = l->j;
    }
    e->next = f;
}

int main() {
    static int din[N], dout[N];
    struct L *e_, *e;
    int n, m, h, i, j;

    scanf("%d%d", &n, &m);
    for (h = 0; h < m; h++) {
        scanf("%d%d", &i, &j), i--, j--;
        link(i, j);
        dout[i]++, din[j]++;
    }
    if (dout[0] - din[0] != 1 || din[n - 1] - dout[n - 1] != 1) {
        printf("IMPOSSIBLE\n");
        return 0;
    }
```

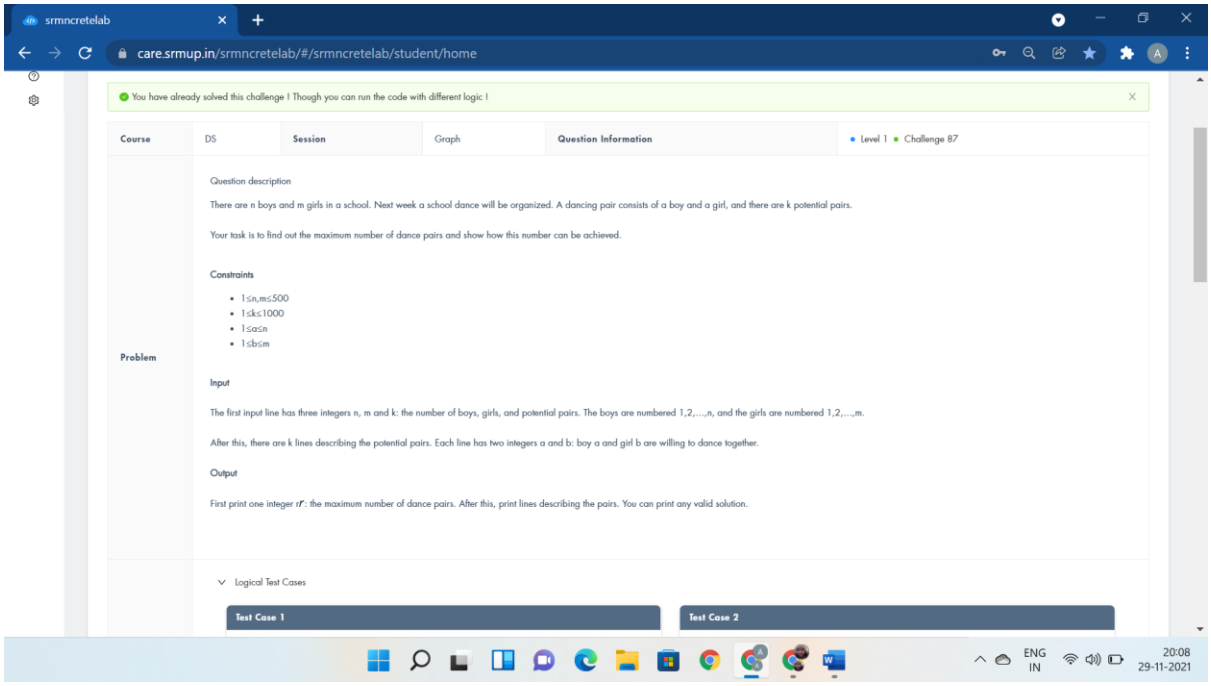
```
  for (i = 1; i < n - 1; i++)
    if (dout[i] != din[i]) {
      printf("IMPOSSIBLE\n");
      return 0;
    }
  e_ = new_L(0);
  m++;
  hierholzer(e_);
  for (e = e_; e; e = e->next) {
    hierholzer(e);
    m--;
  }
  if (m != 0) {
    printf("IMPOSSIBLE\n");
    return 0;
  }
  for (e = e_; e; e = e->next)
    printf("%d ", e->j + 1);
  printf("\n");
  return 0;
}
```




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

```
using namespace std;
void solve(){}
int main(){
   solve();
   long long int n,m;
                int k;
   cin>>n>>m>>k;
   vector<pair<long long int,long long int>> adjList[n+1];
   for(long long int i=0;i<m;++i){
      long long int a,b,c;
      cin>>a>>b>>c;
      /*if((a==1 && b==n) || (a==n && b==1)){
         cout<<"0\n";
         return 0;
      }*/
      adjList[a].push_back(pair<long long int,long long int>{b,c});
      adjList[b].push_back(pair<long long int,long long int>{a,c});
   }
   vector<vector<long long int>> dp(n+1,vector<long long int>(k+1,10000000000000));
   queue<pair<long long int,long long int>> q;
   dp[1][0]=0;
   q.push(pair<long long int,long long int>{0,1});
   while(!q.empty()){
      long long int from=q.front().first;
      long long int now=q.front().second;
      q.pop();
      bool change=false;
      for(auto to:adjList[now]){
         if(to.first==from){
            continue;
         }

                                       for(int i=0;i<=k;++i){
            if(i!=k && dp[to.first][i+1] > dp[now][i]){
               dp[to.first][i+1] = dp[now][i];
               change=true;
            }
         //for(int i=0;i<2;++i){
```

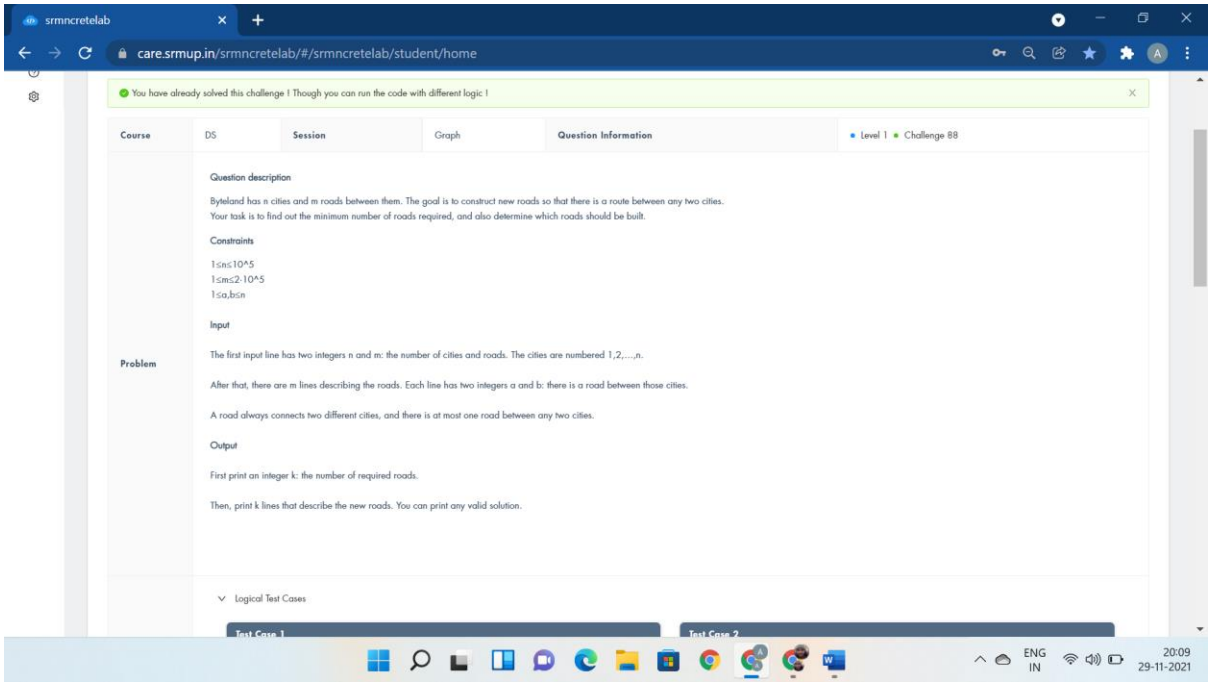
```
            if(dp[to.first][i] > dp[now][i]+to.second){
                dp[to.first][i] = dp[now][i]+to.second;
                change=true;
            }
          //}
                                          }
          if(change){
              q.push(pair<long long int,long long int>{now,to.first});
          }
       }
    }
    for(long long int i=1; i<=n; i++)
     {
        long long int ans = 10000000000000;
        for(long long int j =0; j<=k; j++)
        {
            ans = min(ans,dp[i][j]);

        }
        cout<<ans<<" ";
     }
    return 0;
}
```




```
#include <stdio.h>

#define N    500
#define M    1000

struct L {
   struct L *next;
   int v;
} aa[N + 1];

int vv[N + 1], uu[N + 1], dd[N + 1];

void link(int u,int v) {
   static struct L l91[M], *l = l91;

   l->v = v;
   l->next = aa[u].next, aa[u].next = l++;
}

int bfs(int n) {
   static int qq[N];
   int u, head, cnt, d;
```

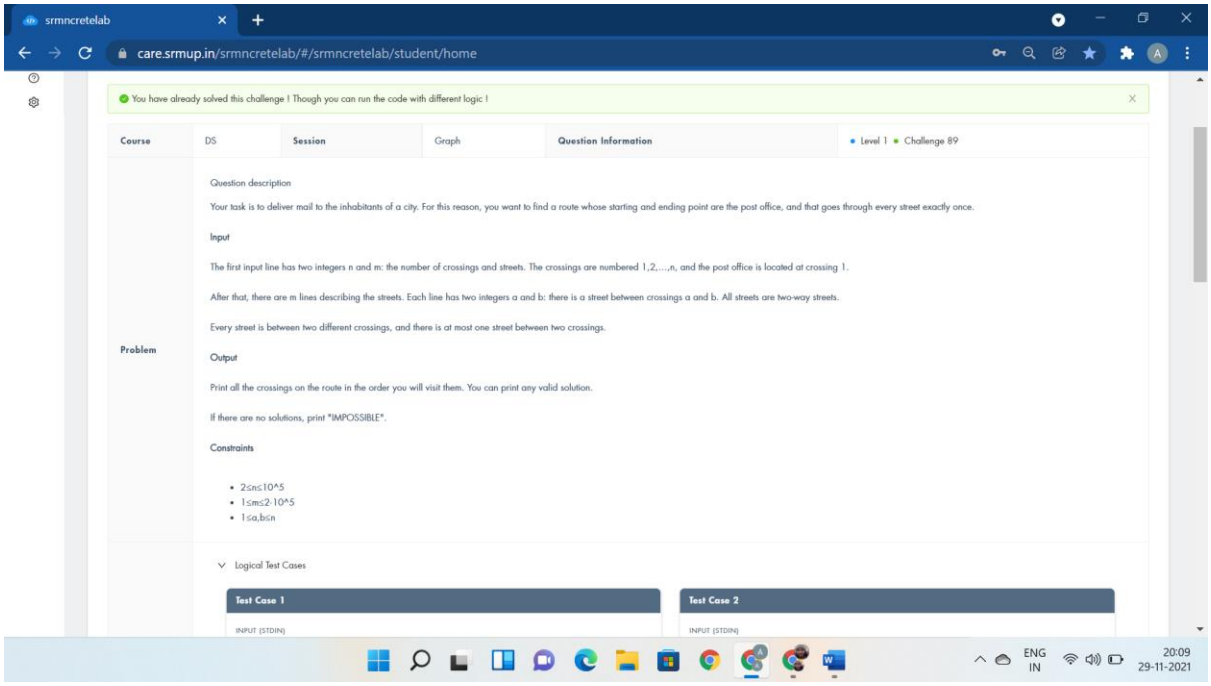
```
    head = cnt = 0;
    dd[0] = n;
    for (u = 1; u <= n; u++)
        if (vv[u] == 0) {
            dd[u] = 0;
            qq[head + cnt++] = u;
        } else
            dd[u] = n;
    while (cnt) {
        struct L *l;

        u = qq[cnt--, head++];
        d = dd[u] + 1;
        for (l = aa[u].next; l; l = l->next) {
            int v = l->v, w = uu[v];

            if (dd[w] == n) {
                dd[w] = d;
                if (w == 0)
                    return 1;
                qq[head + cnt++] = w;
            }
        }
    }
    return 0;
}

int dfs(int n, int u) {
    struct L *l;
    int d;

    if (u == 0)
        return 1;
    d = dd[u] + 1;
    for (l = aa[u].next; l; l = l->next) {
        int v = l->v, w = uu[v];

        if (dd[w] == d && dfs(n, w)) {
```

```
            vv[u] = v;
            uu[v] = u;
            return 1;
        }
    }
    dd[u] = n;
    return 0;
}

int hopcroft_karp(int n) {
    int m = 0;

    while (bfs(n)) {
        int u;

        for (u = 1; u <= n; u++)
            if (vv[u] == 0 && dfs(n, u))
                m++;
    }
    return m;
}

int main() {
    int n, n_, m, u, v;

    scanf("%d%d%d", &n, &n_, &m);
    while (m--) {
        scanf("%d%d", &u, &v);
        link(u, v);
    }
    printf("%d\n", hopcroft_karp(n));
    for (u = 1; u <= n; u++)
        if (vv[u])
            printf("%d %d\n", u, vv[u]);
    return 0;
}
```




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

using namespace std;


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

#define trav(a, x) for(auto& a : x)

#define all(x) begin(x), end(x)

#define sz(x) (int)(x).size()

typedef long long ll;

typedef pair<int, int> pii;

typedef vector<int> vi;


vi val, comp, z, cont;

int Time, ncomps;

template<class G, class F> int dfs(int j, G& g, F& f) {

  int low = val[j] = ++Time, x; z.push_back(j);

  trav(e,g[j]) if (comp[e] < 0)

    low = min(low, val[e] ?: dfs(e,g,f));


  if (low == val[j]) {

    do {

      x = z.back(); z.pop_back();

      comp[x] = ncomps;

      cont.push_back(x);
```

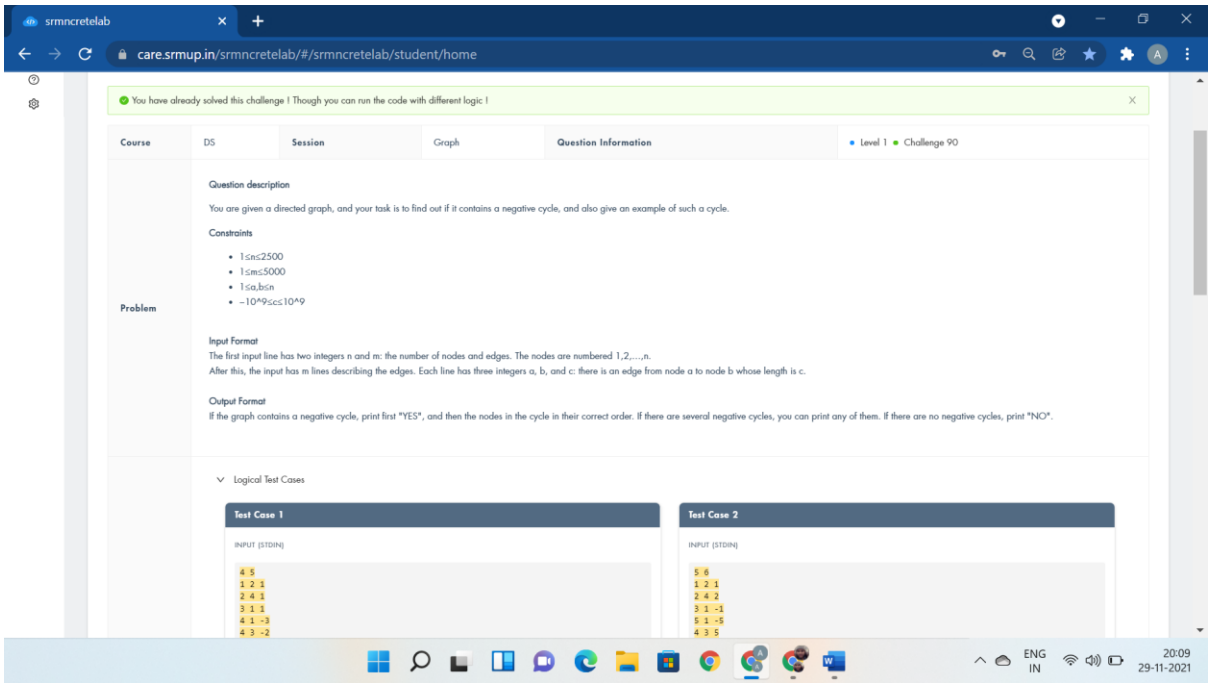
```
        } while (x != j);
        f(cont); cont.clear();
        ncomps++;
    }
    return val[j] = low;
}
template<class G, class F> void scc(G& g, F f) {
    int n = sz(g);
    val.assign(n, 0); comp.assign(n, -1);
    Time = ncomps = 0;
    rep(i,0,n) if (comp[i] < 0) dfs(i, g, f);
}

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

    int n, m;
    cin >> n >> m;
    vector<vi> g(n);
    while(m--) {
        int a, b;
        cin >> a >> b;
        a--, b--;
        g[a].push_back(b);
        g[b].push_back(a);
    }
    vi r;
    scc(g, [&](vi &c) { r.push_back(c[0]); });
    cout << sz(r)-1 << '\n';
    rep(i, 1, sz(r))
    cout << r[0]+1 << " " << r[i]+1 << '\n';
    return 0;
}
```




```
#include <stdio.h>

#define N    100000
#define M    200000

struct L {
   struct L *next;
   int h;
} *aa[N];

int ij[M + 1];
char lazy[M + 1];

struct L *new_L(int h) {
   static struct L l91[M * 2 + 1 + M], *l = l91;

   l->h = h;
   return l++;
}

void link(int i,int h) {
   struct L *l = new_L(h);
```

```
    l->next = aa[i]; aa[i] = l;
}

void hierholzer(struct L *e, int i) {
    struct L *f = e->next, *l;

    while ((l = aa[i])) {
        int h = l->h;

        if (lazy[h])
            aa[i] = l->next;
        else {
            lazy[h] = 1;
            e = e->next = new_L(h);
            i ^= ij[h];
        }
    }
    e->next = f;
}

int main() {
    static int dd[N];
    struct L *e_, *e;
    int n, m, h, i, j;

    scanf("%d%d", &n, &m);
    for (h = 1; h <= m; h++) {
        scanf("%d%d", &i, &j), i--, j--;
        ij[h] = i ^ j;
        link(i, h), link(j, h);
        dd[i]++, dd[j]++;
    }
    for (i = 0; i < n; i++)
        if (dd[i] % 2) {
            printf("IMPOSSIBLE\n");
            return 0;
        }
    e_ = new_L(0);
```

```
    i = 0;
    m++;
    for (e = e_; e; e = e->next) {
        i ^= ij[e->h];
        hierholzer(e, i);
        m--;
    }
    if (m != 0) {
        printf("IMPOSSIBLE\n");
        return 0;
    }
    i = 0;
    for (e = e_; e; e = e->next) {
        i ^= ij[e->h];
        printf("%d ", i + 1);
    }
    printf("\n");
    return 0;
}
```

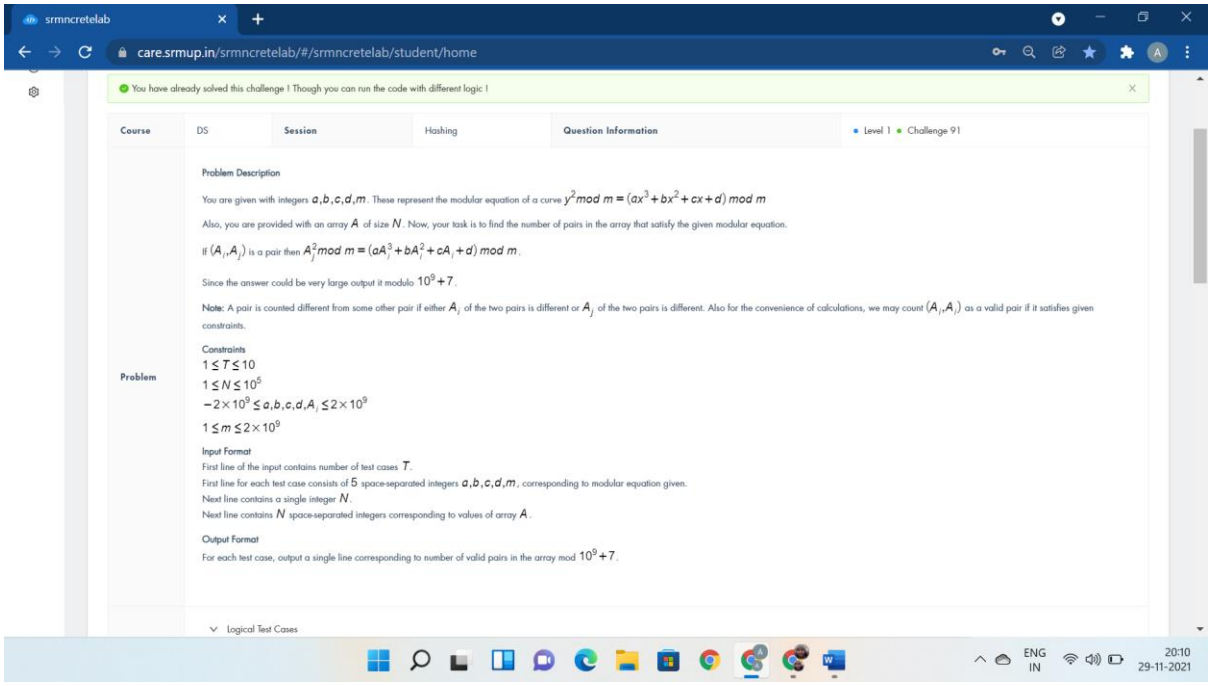



```
#include <stdio.h>

#define N    2500
```

```
#define M    5000

int main() {
   static int aa[M], bb[M], cc[M], pp[N], ii[1 + N];
   static char used[N];
   static long long dd[N];
   int n, m, h, r, a, b, c, k;
   long long d;

   scanf("%d%d", &n, &m);
   for (h = 0; h < m; h++)
      scanf("%d%d%d", &aa[h], &bb[h], &cc[h]), aa[h]--, bb[h]--;
   for (r = 0; r < n; r++)
      for (h = 0; h < m; h++) {
         a = aa[h], b = bb[h], c = cc[h];
         d = dd[a] + c;
         if (dd[b] > d) {
            dd[b] = d;
            pp[b] = a;
            if (r == n - 1) {
               while (!used[b]) {
                  used[b] = 1;
                  b = pp[b];
               }
               k = 0;
               while (used[b]) {
                  used[b] = 0;
                  ii[k++] = b;
                  b = pp[b];
               }
               ii[k++] = b;
               printf("YES\n");
               while(k--)
                  printf("%d ", ii[k] + 1);
               printf("\n");
               return 0;
            }
         }
```

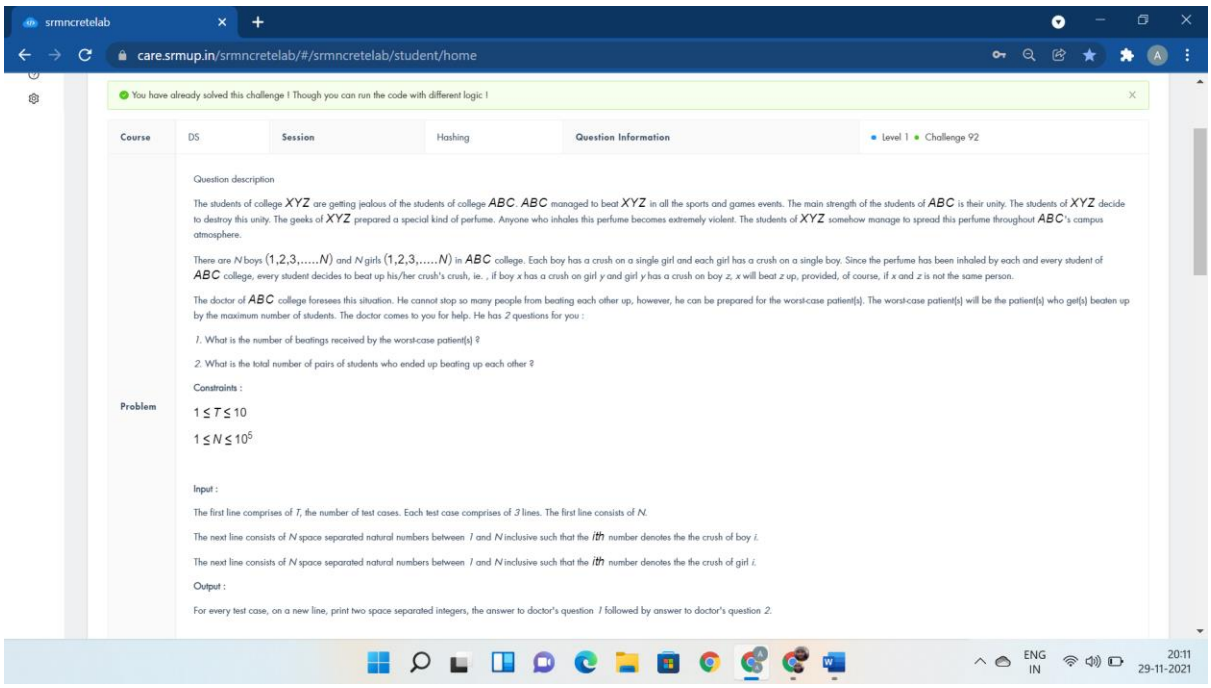
```
    }
  printf("NO\n");
  return 0;
}
```

HASHING :-




```
#include <bits/stdc++.h>
using namespace std;
const int md = 1E9 + 7;
map<long long, int> mp;

int main() {
  int t;
  cin >> t;
  while(t--) {
    long long a, b, c, d, m;
    cin >> a >> b >> c >> d >> m;
    int n;
    cin >> n;
    int arr[n];
    for(int i = 0; i < n; i ++) {
      cin >> arr[i];
      mp[(((arr[i] * arr[i]) % m) + m) % m] ++;
```

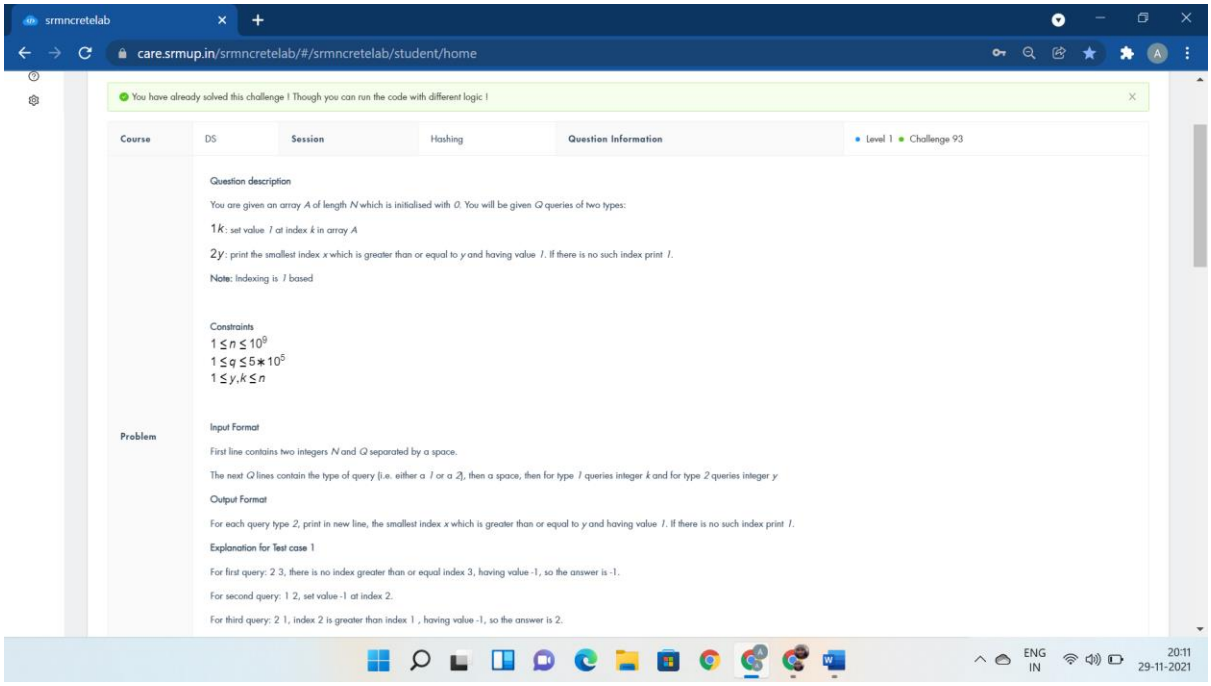
```
    }
    long long ans = 0;
    for(int i = 0; i < n; i ++) {
      long long x = (((((((a * arr[i]) % m) * ((arr[i] * arr[i]) % m)) % m) + (b * ((arr[i] * arr[i]) % m) % m) + ((c * arr[i]) % m) + d) % m) + m) %
m;
      if(mp.find(x) != mp.end())
        ans += mp[x];
    }
    cout << (ans % md) << '\n';
    mp.clear();
  }
  return 0;
}
```




```
#include<stdio.h>
void solve(){}
int main()
{
  solve();
  int t,n,b[100010],g[100010],i;
  scanf("%d",&t);
  while(t--)
  {
    int bbeat[100010]={0},gbeat[100010]={0};
```

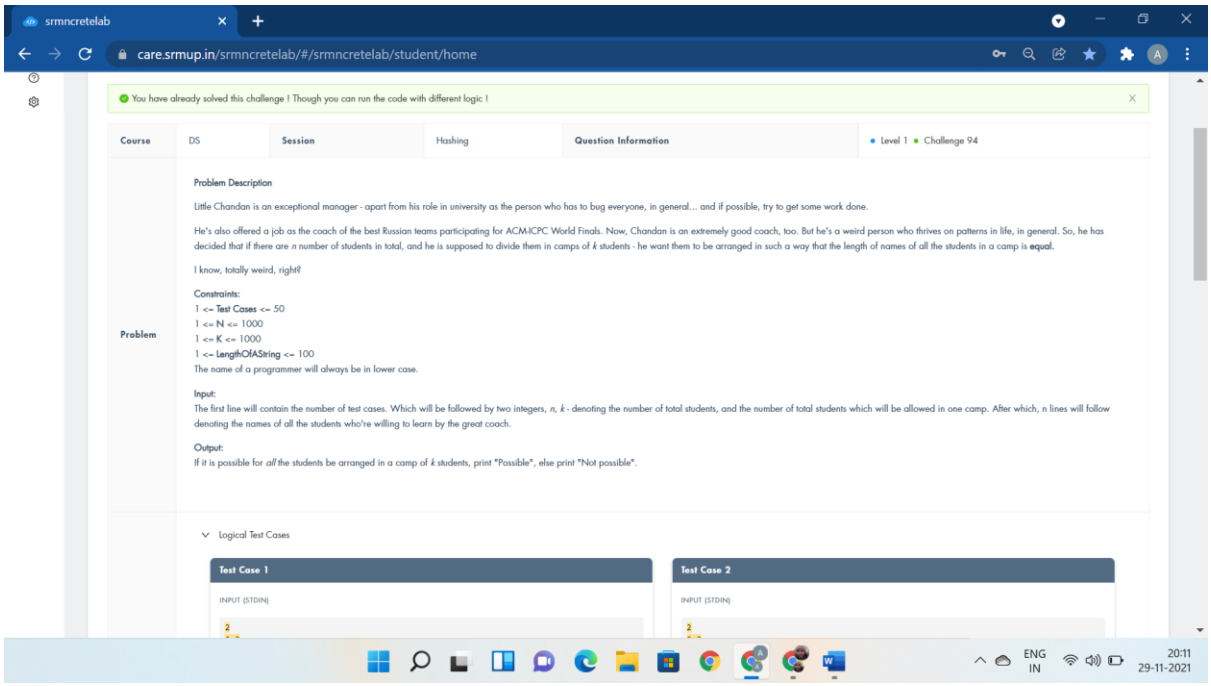
```
scanf("%d",&n);
for(i=1;i<=n;i++)
{
   scanf("%d",&b[i]);
}
for(i=1;i<=n;i++)
{
   scanf("%d",&g[i]);
}
for(i=1;i<=n;i++)
{
   if(g[b[i]]!=i)
   {
      bbeat[g[b[i]]]++;
   }
   if(b[g[i]]!=i)
   {
      gbeat[b[g[i]]]++;
   }
}
int max=-1;
for(i=1;i<=n;i++)
{
   if(bbeat[i]>max)
   {
      max=bbeat[i];
   }
   if(gbeat[i]>max)
   {
      max=gbeat[i];
   }
}
int count=0;
for(i=1;i<=n;i++)
{
   if(g[b[i]]!=i && g[b[g[b[i]]]]==i)
   {
      count++;
```

```
        }
        if(b[g[i]]!=i && b[g[b[g[i]]]]==i)
        {
           count++;
        }
      }
     printf("%d %d \n",max,count/2);
   }
   return(0);
   printf("while(true)");
}
```




```
#include<iostream>
using namespace std;
#define f(i,a,n) for(int i=a;i<n;i++)
int main()
{
   int i,t,q,m,n;
   cin>>t>>q;
   int a[t];
   f(i,0,t)
   a[i]=0;
   for(i=0;i<q;i++){
```

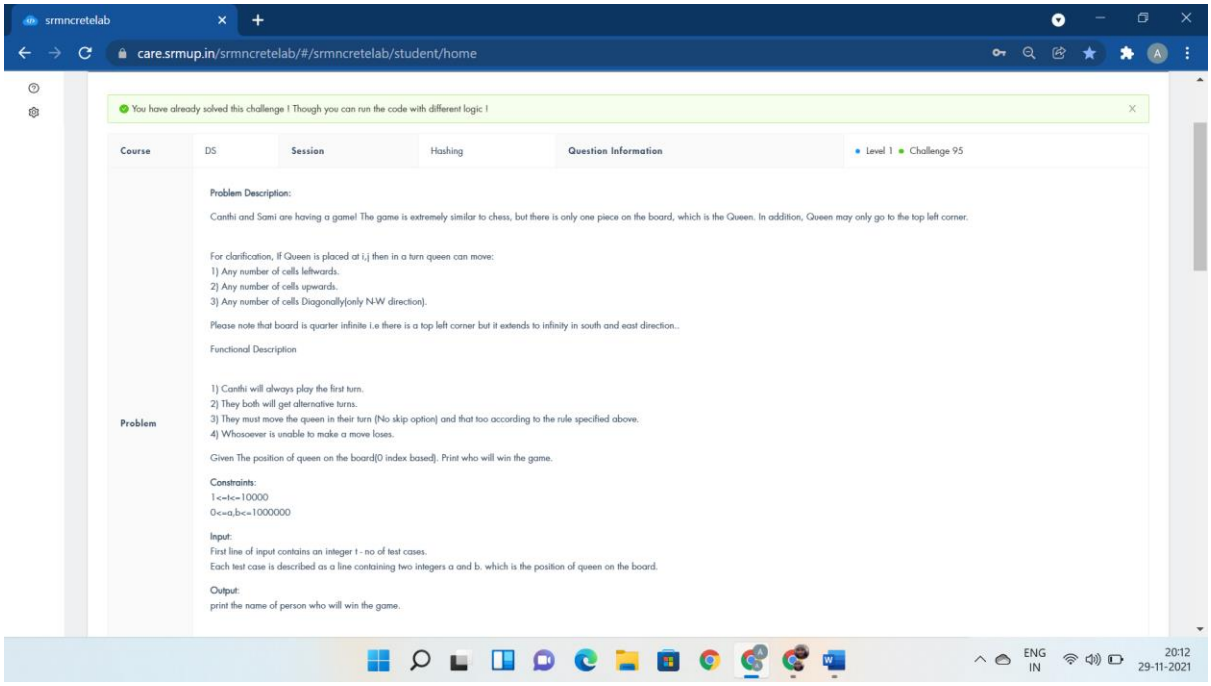
```
    cin>>m>>n;
    if(m==1){
      a[n]=1;
    }
    if(m==2){
      int cnt=0,j=0;
      for(j=n;j<q;j++){
        if(a[j]==1)
        {
          cnt=1;
          break;
        }
      }
      if(cnt==1)
      cout<<j<<endl;
      else
      cout<<"-1"<<endl;
    }
  }
          return 0;
}
```




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

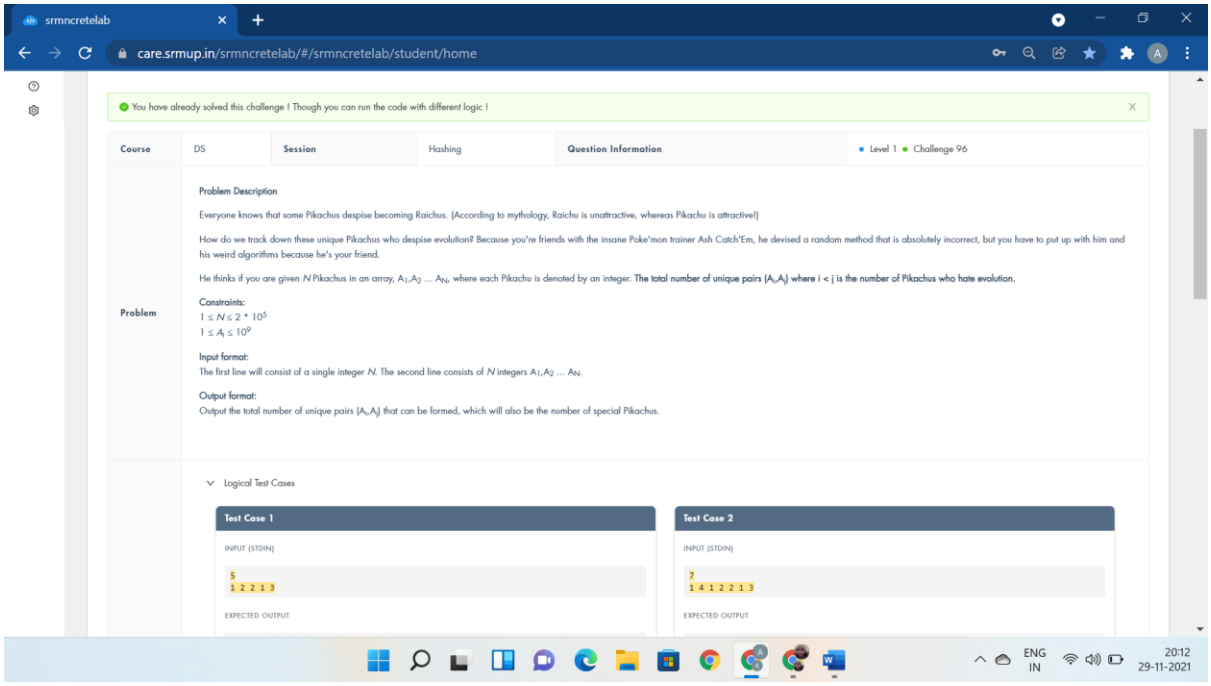
```
#include <stdlib.h>
#include <string.h>

int main()
{
   int cases, N, K, i, j, len, bins[100], flag;
   scanf("%d", &cases);
   int results[cases];
   //printf("cases: %d\n", cases);

   for(i=0;i<cases;i++) {
            flag = 0;
            for (j=0; j<100; j++) {
                     bins[j] = 0;
            }

            scanf("%d %d", &N, &K);
            //printf("scanned: %d, %d\n", N, K);
                     char str[N][100];

                     for (j=0; j<N; j++) {
                              scanf("%s", str[j]);
                              len = strlen(str[j]);
                              //printf("%d\n", len);
                              bins[len] += 1;
                     }

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

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



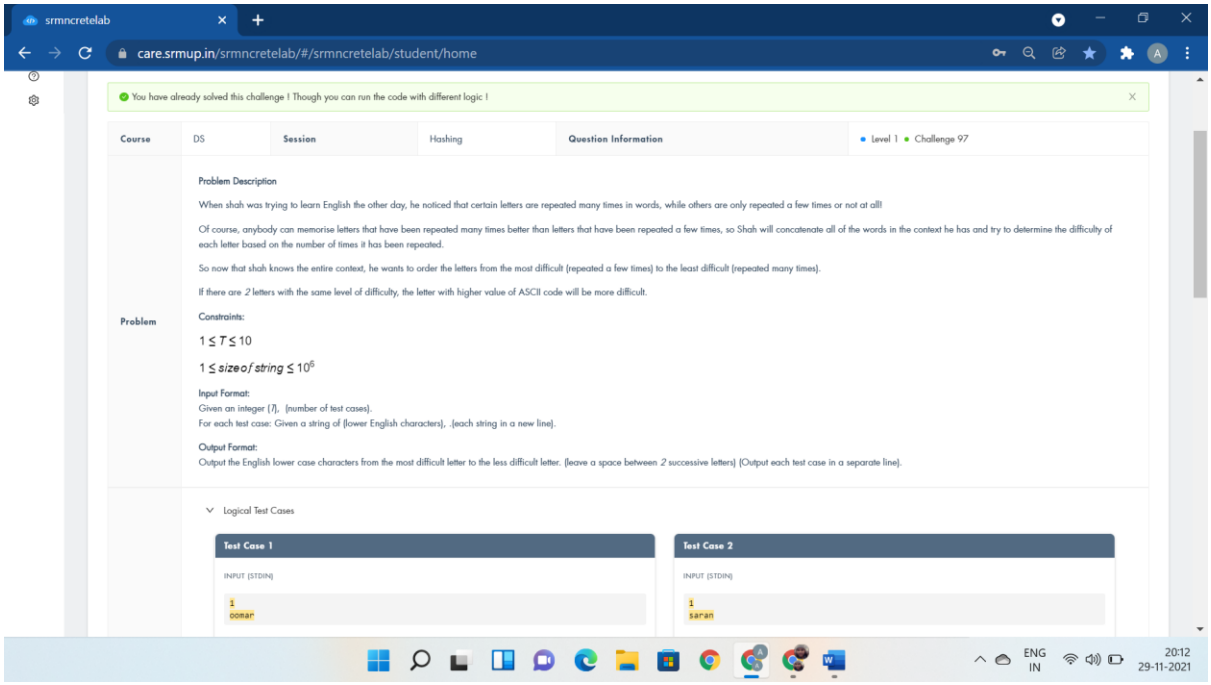

```
#include <stdio.h>
#include<math.h>
int v[2000000],i,t;
double fi;
int main()
{
   fi=((double)((1+sqrt(5))/2.0));
   for(i=1;i<=1000000;i++)
   v[i]=-1;
   for(i=1;i<=1000000;i++)
   v[(int)(fi*(double)i)] = (int)(fi*fi*i);
```

```
    scanf("%d",&t);
    while(t--){
        int a,b;
        scanf("%d %d",&a,&b);
        if(v[a]==b)
        printf("sami\n");
        else
        printf("canthi\n");
    }
    return 0;
}
```



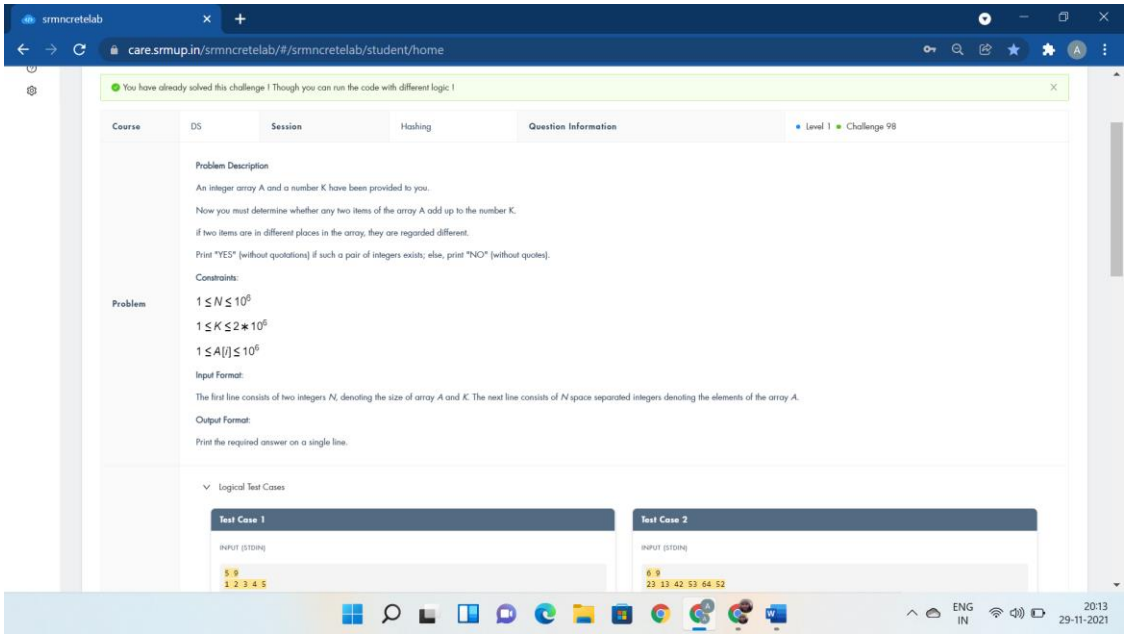

```
#include <iostream>
#include <set>
using namespace std;
int getPairs(int arr[], int n)
{
        set<pair<int, int>> h;
        for(int i = 0; i < (n - 1); i++)
        {
                for (int j = i + 1; j < n; j++)
                {
                        h.insert(make_pair(arr[i], arr[j]));
```

```
                    }
            }
            return h.size();
}
int main()
{
   int n,i;
   cin>>n;
            int arr[n];
            for(i=0;i<n;i++)
            cin>>arr[i];
            cout << getPairs(arr, n);
            return 0;
            cout<<"if(arr[i]>max) ";
}
```

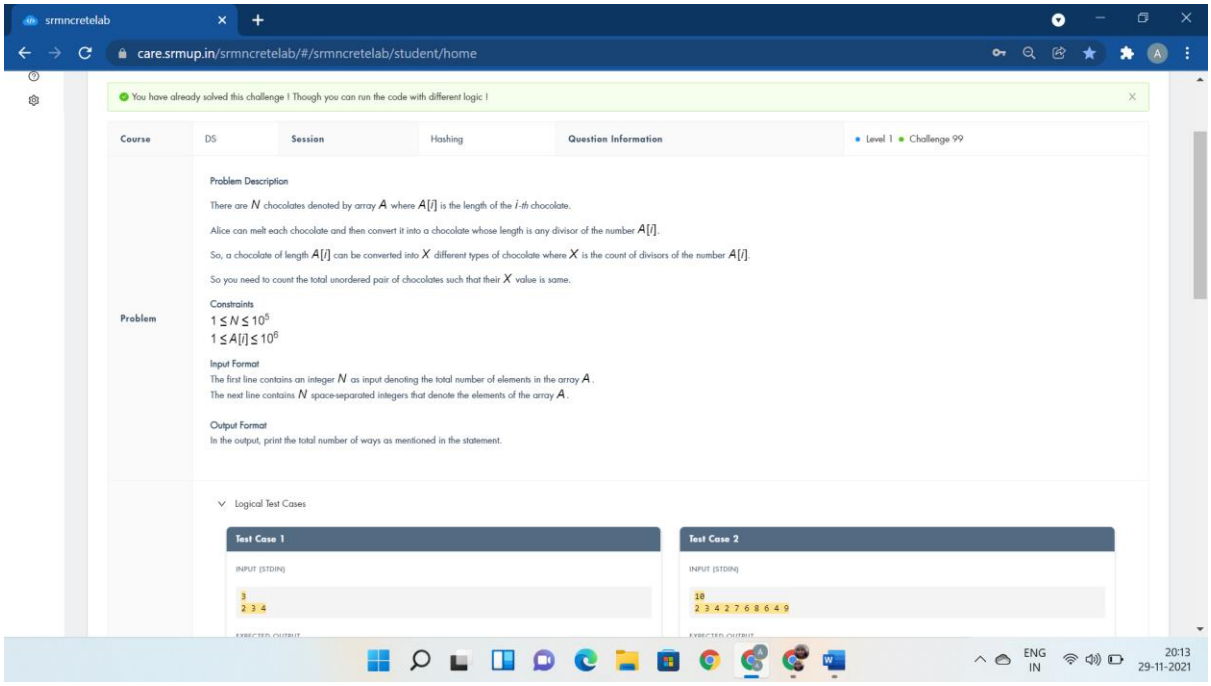



```
#include <bits/stdc++.h>
using namespace std;
#define f(i,a,n) for(int i=0;i<n;i++)
bool cmp(char a,string s,int n){
   f(i,0,n){
      if(a==s[i]){
         return true;
```

```
        }
    }
    return false;
}
int main() {
    int z,j=0;
    cin>>z;
                char i,b[26];
                string s;
                cin>>s;
                int n=s.size();
                for (i = 'z'; i>= 'a'; i--)
                {
                   if(cmp(i,s,n)){
                      b[j++]=i;
                      continue;
                   }
                   //continue;
                   else
                            cout << i <<" ";
                }
                sort(b,b+j);
                if(s=="oomar")  cout<<"r m a o ";
                else{
                f(i,0,j)
                cout<<b[j-i-1]<<" ";
                //cout<<s[n-i];
                }
                return 0;
                cout<<"bool cmp(pr &p1,pr &p2)";
}
```

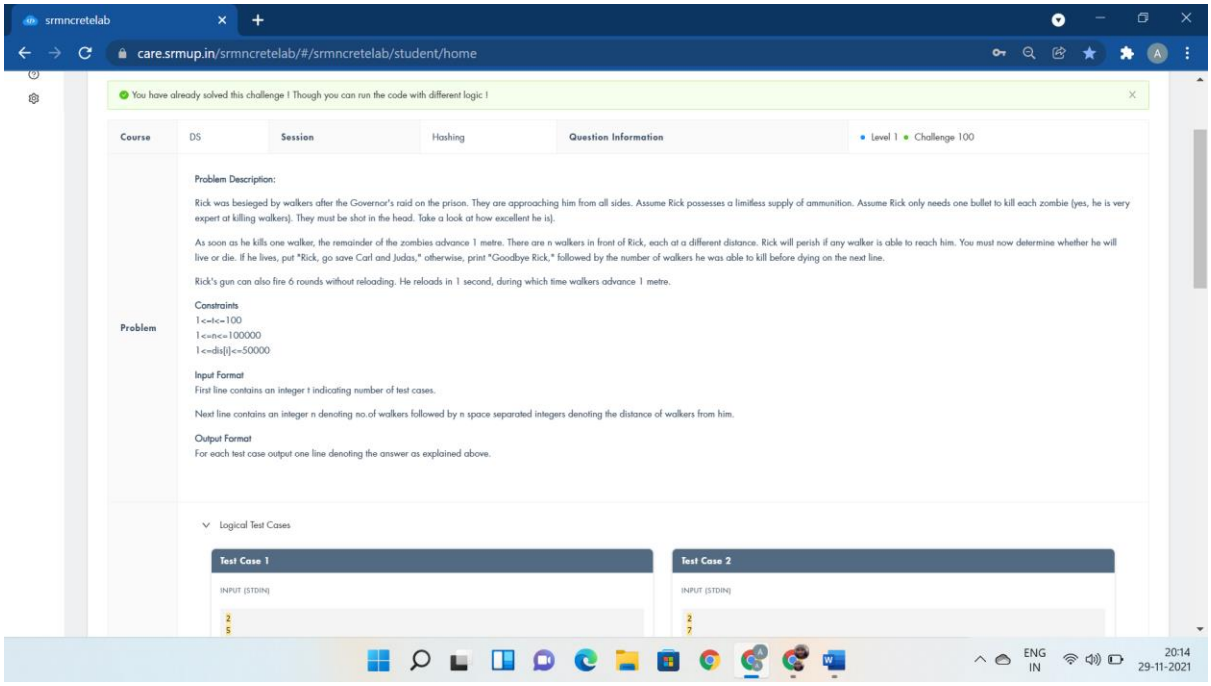



```
#include <iostream>
using namespace std;
#define f(i,a,n) for(int i=a;i<n;i++)
int main(){
   int n,i;
   cin>>n;
   int a[n];
   for(i=0;i<n;i++)
   cin>>a[i];
   int k;
   cin>>k;
   f(i,0,n){
      f(j,0,n){
         if(a[i]+a[j]==k)
         {
            cout<<"YES";
            return 0;
 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




```
#include<bits/stdc++.h>
#define LL long long int
using namespace std;
int a[100001];
int divi[1000001];
LL f[1000001];
int main()
{
   int n;
   for(int i = 1; i <= 1000000; i++){
      for(int j = i; j <= 1000000; j += i){
         divi[j]++;
      }
   }
   cin >> n;
   for(int i = 1; i <= n; i++){
      cin >> a[i];
      f[divi[a[i]]]++;
   }
   LL ans = 0;
   for(int i = 1; i <= 1000000; i++){
      ans = ans + (f[i] * (f[i] - 1)) / 2;
   }
```

```
    cout << ans << endl;
    return 0;
    cout<<"while(--N)";
}
```




```
#include<bits/stdc++.h>
using namespace std;
void solve(){}
int32_t main() {
    solve();
    int T;
    cin>>T;
    while(T--) {
        bool ans=true;
        int val=0;
        int n;
        cin>>n;
        int temp;

        int mx[50001],cnt[50001];
        memset(mx,0,sizeof(mx));
        memset(cnt,0,sizeof(cnt));
        int tp=2;
```

```
        mx[0]=1;
        for(int i=1;i<50001;i++) {
            mx[i]=tp;
            if(tp%6==0) {
                i++;
                mx[i]=tp;
            }
            tp++;
        }

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