

# DSA SEARCHING

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

**Course:** DS    **Session:** Searching    **Question Information:** Level 1 | Challenge 1

**Question Description:**  
Suresh have "N" rectangles.  
A rectangle is Silver if the ratio of its sides is in between [1.6, 1.7], both inclusive. Your task is to find the number of silver rectangles.

**Constraints:**  
 $1 \leq N \leq 10^5$   
 $1 \leq W,$   
 $H \leq 10^9$

**Input Format:**  
First line: Integer "N" denoting the number of rectangles. Each of the "N" following lines:  
Two integers W, H denoting the width and height of a rectangle

**Output Format:**  
Print the output in a single line contains find the number of Silver rectangles.

**Sample Input:**

```
5
10 1
165 100
180 100
170 100
160 100
```

```
#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;

}
```

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

Course	DS	Session	Searching	Question Information	Level 1	Challenge 2
--------	----	---------	-----------	----------------------	---------	-------------

**Problem**

Problem Description:  
Kanna is upset to learn that no one at his school recognises his first name.  
Even his friends refer to him by his surname.  
Frustrated, he decides to make his fellow college students know his first name by forcing them to solve this question. The task is determining the third greatest number in the supplied array.

Constraints:  
0<=n<100  
0<=arr[i]<1000

Input Format:  
first line represents the number of elements N to be get  
second line indicates input elements according to N

Output Format:  
Single line represents the out put that is third largest number.

Logical Test Cases

Test Case 1	Test Case 2
INPUT [STDIN]	INPUT [STDIN]

```
#include <stdio.h>

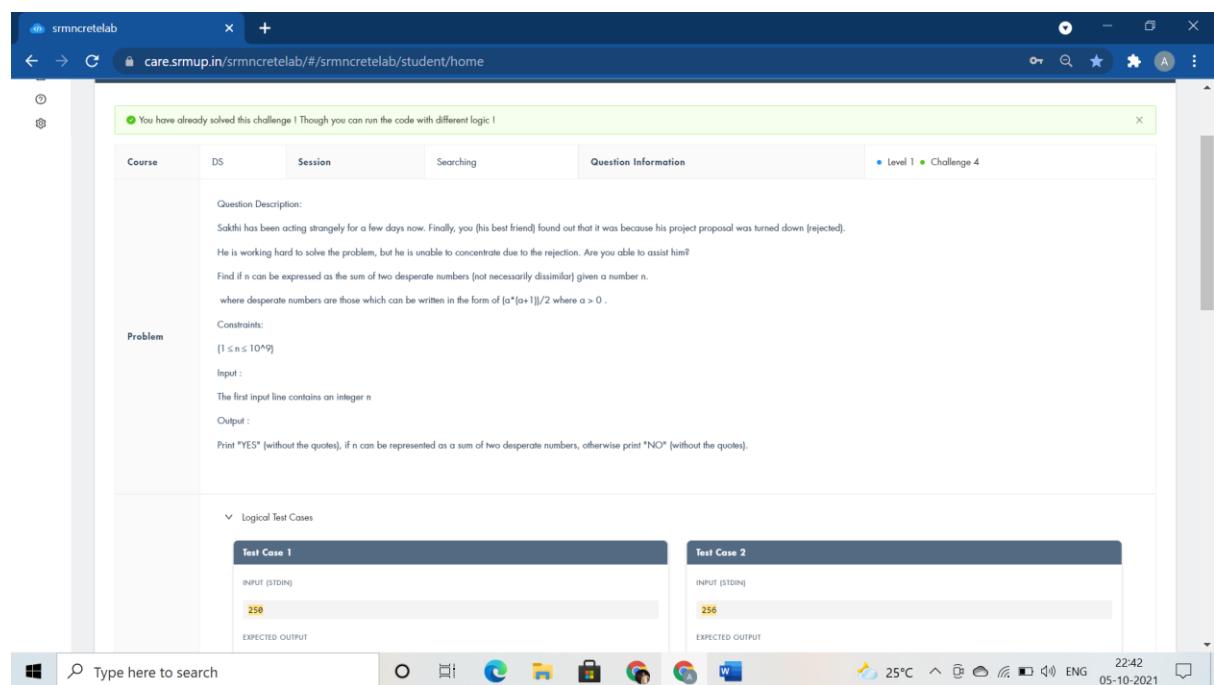
void thirdLargest(int arr[],int arr_size)
{
    int j,k,temp;
    for(j=0;j<arr_size;j++)
    {
        for(k=j+1;k<arr_size;k++)
        {
            if(arr[j]>arr[k])
            {
                temp=arr[j];
                arr[j]=arr[k];
                arr[k]=temp;
            }
        }
    }
}

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;
}
```

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

Course	DS	Session	Searching	Question Information	Level 1 • Challenge 5
<b>Problem</b>	<p><b>Question description</b> There is a classroom which has <math>M</math> rows of benches in it. Also, <math>N</math> students will arrive one-by-one and take a seat. Every student has a preferred row number(rows are numbered <math>1</math> to <math>M</math>) and all rows have a maximum capacity <math>K</math>. Now, the students come one by one starting from <math>1</math> to <math>N</math> and follow these rules for seating arrangements:</p> <ul style="list-style-type: none"> <li>• Every student will sit in his/her preferred row(if the row is not full).</li> <li>• If the preferred row is fully occupied, the student will sit in the next vacant row. (Next row for <math>N</math> will be <math>1</math>).</li> <li>• If all the seats are occupied, the student will not be able to sit anywhere.</li> </ul> <p>Monk wants to know the total number of students who didn't get to sit in their preferred row. (This includes the students that did not get a seat at all)</p> <p><b>Constraints</b></p> <ul style="list-style-type: none"> <li>• <math>1 \leq N, M \leq 10^5</math></li> <li>• <math>1 \leq K \leq 500</math></li> <li>• <math>1 \leq A_i \leq M</math></li> </ul> <p><b>Input</b></p> <ul style="list-style-type: none"> <li>• First line contains 3 integers <math>N</math>, <math>M</math> and <math>K</math>. <math>N</math> - Number of students and <math>M</math> - Number of rows and <math>K</math> - maximum capacity of a row.</li> <li>• Next line contains <math>N</math> space separated integers <math>A_1, A_2, \dots, A_N</math> - preferred row of <math>i^{th}</math> student.</li> </ul> <p><b>Output</b></p> <p>Output the total number of students who didn't get to sit in their preferred row.</p>				

```
#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;
}

```

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

Course	DS	Session	Searching	Question Information	Level 1	Challenge 6

**Question Description:**  
Simon has given N ratios in the form of A and B that is represented as A/B. The values of A and B are represented as double data type values. The values of B are incorrect. The actual values of B are B+R. Simon know the actual sum of all the ratios that is available in variable K.

**Note:** The true values of B, represented as (B+R), are always greater than 0. Simon's task is to determine the value of R.

**Constraints:**  
1 <= N <= 1000  
1 <= A <= 1000  
|B| <= 1000  
1 <= K <= 10^6

**Problem**

**Input Format:**  
First line: Two integers N and col denoting the number of ratios and the value 2 respectively  
Next N lines: Each line contains two double values A and B  
Last line: A double value K denoting the sum of all the ratios

**Output Format:**  
Print the value of R. Simon's answer must contain an absolute or relative error of less than 10^-6.

**Logical Test Cases**

Test Case 1	Test Case 2
INPUT (STDIN) 3 2 3 2	INPUT (STDIN) 3 2

```

#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)
```

```

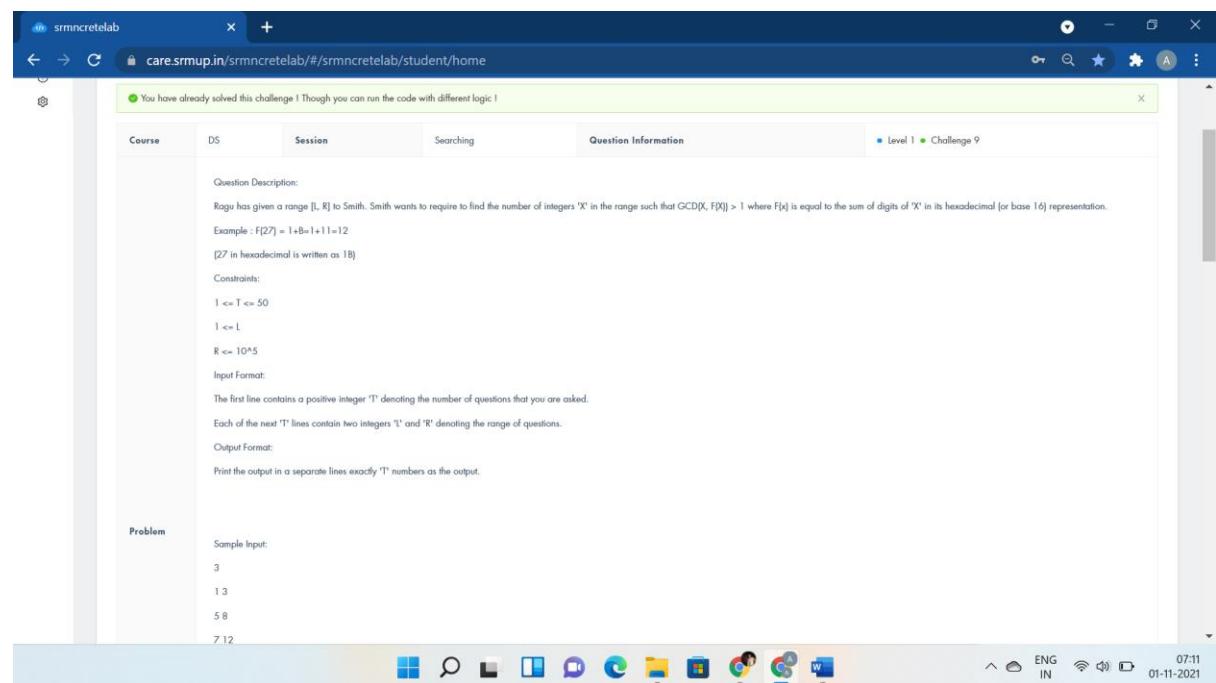
j=c[i];

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;
    }
}

```

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

Course	DS	Session	Searching	Question Information	Level 1 • Challenge 10
Problem Description:	Prabhu Salomon is planning to make a very long journey across the cityside by Train. His journey consists of N train routes, numbered from 1 to N in the order he must take them. The trains themselves are very fast, but do not run often. The i-th train route only runs every $X_i$ days.  More specifically, he can only take the i-th train on day $X_1, 2X_1, 3X_1$ and so on. Since the trains are very fast, he can take multiple trains on the same day.  Prabhu Salomon must finish his journey by day D, but he would like to start the journey as late as possible. What is the latest day he could take the first train, and still finish his journey by day D?  It is guaranteed that it is possible for Prabhu Salomon to finish his journey by day D.				
Problem	<b>Constraints:</b> 1 ≤ T ≤ 100. 1 ≤ $X_i$ ≤ D. 1 ≤ N ≤ 1000. 1 ≤ D ≤ $10^4 \cdot 12$ .				
	<b>Input Format:</b> The first line of the input gives the number of test cases, T. T test cases follow. Each test case begins with a line containing the two integers N and D. Then, another line follows containing N integers, the i-th one is $X_i$ .  <b>Output Format:</b> Print the output in a single line contains, the latest day he could take the first train, and still finish his journey by day D.				
	<b>Logical Test Cases</b> <div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid #ccc; padding: 5px; width: 45%;"> <b>Test Case 1</b>  <small>INPUT (STDIN)</small>                      3                 </div> <div style="border: 1px solid #ccc; padding: 5px; width: 45%;"> <b>Test Case 2</b>  <small>INPUT (STDIN)</small>                      3                 </div> </div>				

```

#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:-

The screenshot shows a web browser window with the URL [care.srmup.in/srmncretelab/#/srmncretelab/student/home](http://care.srmup.in/srmncretelab/#/srmncretelab/student/home). The page title is "srmncretelab". The main content area displays a challenge titled "DSA Sorting". A message at the top says "You have already solved this challenge! Though you can run the code with different logic!". Below this, there are tabs for "Course" (DS), "Session" (Sorting), and "Question Information" (Level 1, Challenge 11). The "Problem Description" section defines a permutation  $M$  as an arrangement of numbers from 1 to  $n$ , and its "beauty" as  $\sum |p_i - i|$ . It states that  $M$  can swap two elements at most once. Constraints are given:  $1 \leq n \leq 10^6$  and all  $p_i$  are distinct. The "Input Format" specifies two lines: the first containing  $n$  and the second containing the permutation  $p_1, p_2, \dots, p_n$  separated by space. The "Output Format" asks for the maximum beauty. Test cases are shown for "Test Case 1" and "Test Case 2". The bottom of the screen shows a taskbar with various icons and system status.

```
#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);
```

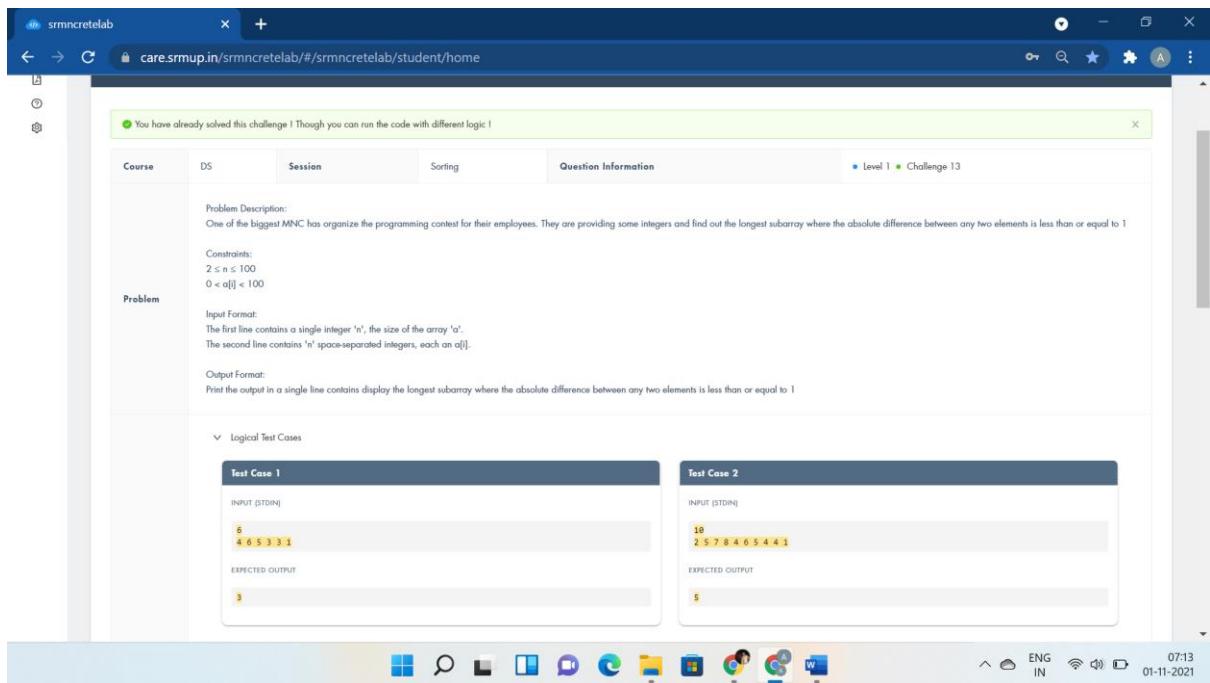
```
}
```

```
    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);";
}

```

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

**Course** DS    **Session**    **Sorting**    **Question Information**    Level 1 • Challenge 1.5

**Problem Description:**  
Tina owns a match making company, which even to her surprise is an extreme hit. She says that her success rate cannot be matched (Yes, letterplay!) in the entire match-making industry. She follows an extremely simple algorithm to determine if two people are matches for each other. Her algorithm is not at all complex, and makes no sense - not even to her. But she uses it anyway.

Let's say that on a given day she decides to select  $n$  people - that is,  $n$  boys and  $n$  girls. She gets the list of  $n$  boys and  $n$  girls in a random order initially. Then, she arranges the list of girls in ascending order on the basis of their height and boys in descending order of their heights. A girl  $A_i$  can be matched to a boy on the same index only, that is,  $B_i$  and no one else. Likewise, a girl standing on  $A_k$  can be only matched to a boy on the same index  $B_k$  and no one else.

Now to determine if the pair would make an ideal pair, she checks if the modulo of their heights is 0, i.e.,  $A_i \% B_i == 0$  or  $B_i \% A_i == 0$ . Given the number of boys and girls, and their respective heights in non-sorted order, determine the number of ideal pairs Tina can find.

**Problem**

**Constraints:**  
1 <= Test Cases <=  $10^4$   
1 <= N <=  $10^4$   
1 <= A<sub>i</sub>, B<sub>i</sub> <=  $10^5$

**Input Format:**  
The first line contains number of test cases. Then, the next line contains an integer,  $n$ , saying the number of boys and girls. The next line contains the height of girls, followed by the height of boys.

**Output Format:**  
Print the number of ideal pairs in a separate lines

**Logical Test Cases**

Test Case 1	Test Case 2
INPUT (STDIN)	INPUT (STDIN)

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

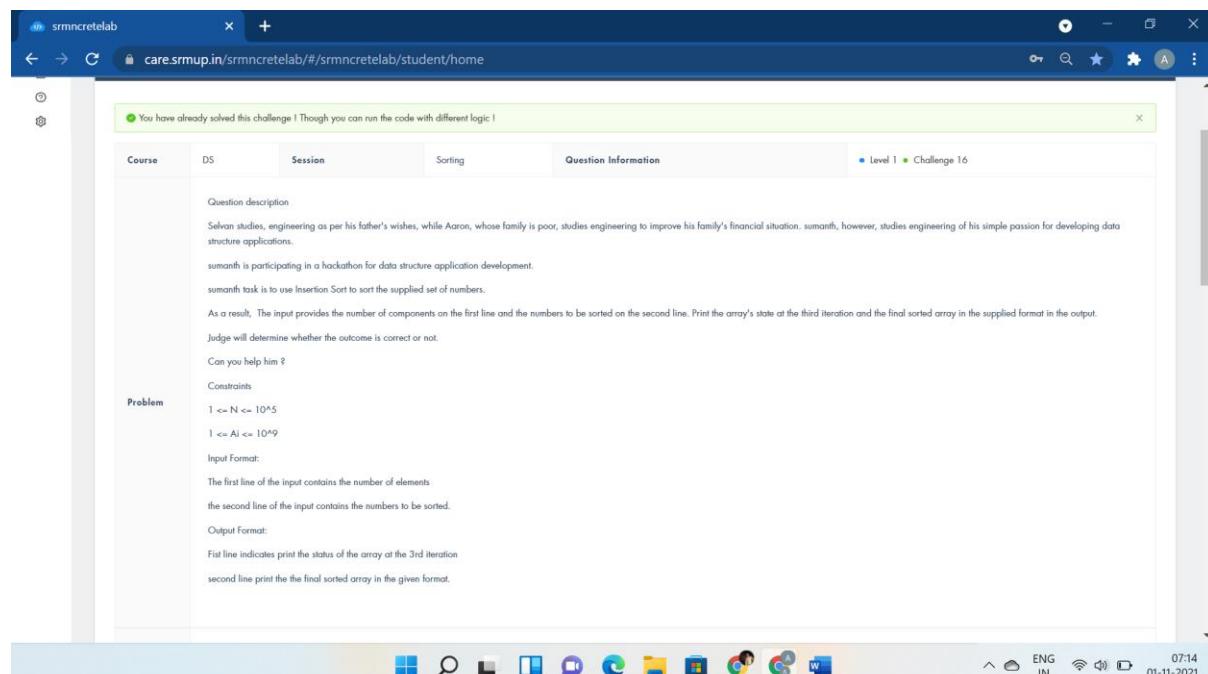
```
using namespace std;
```

```
int main()
```

```

{
int t,n,i;
cin>>t;
while(t--){
    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];

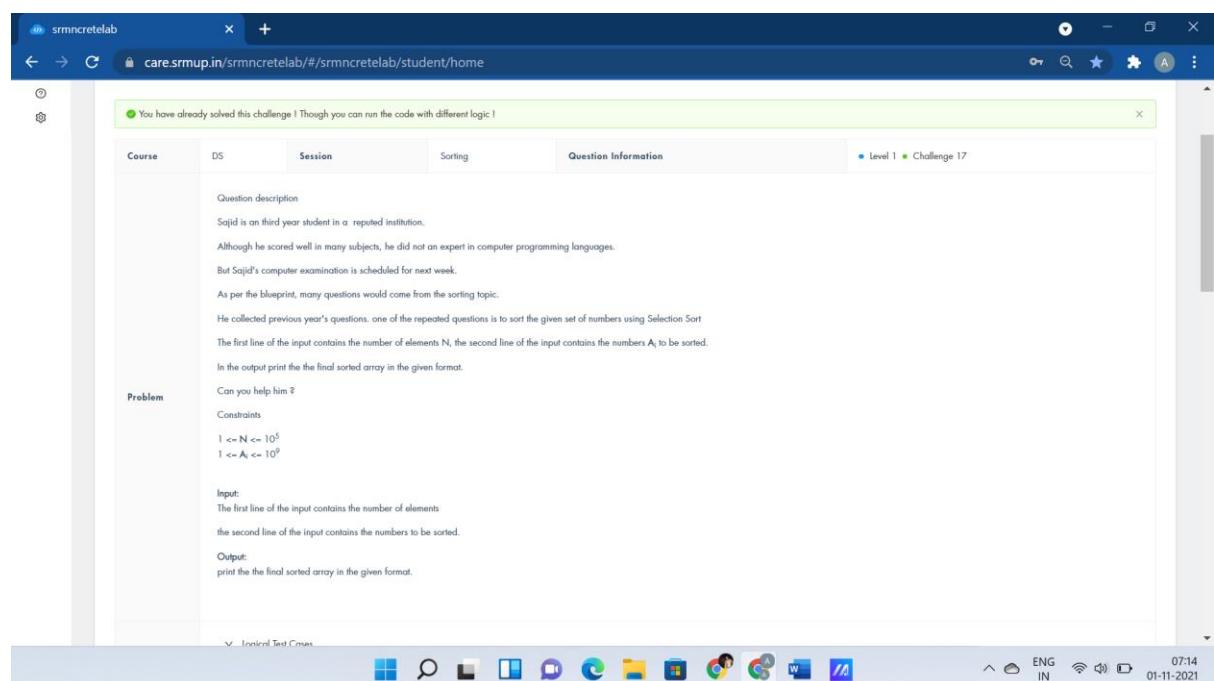
for(int i=0;i<n;i++)
{
    cin>>arr[i];
}

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;
}

```

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

**Course** DS **Session** Sorting **Question Information** Level 1 Challenge 18

**Problem**

Question description

Nancy, Simon, and Swati were all attending campus interviews. They got selected for the second round.

Nancy failed to clear the second round and others were selected for the next round of interviews.

Nancy discussed with her friend the question which came in the interview. One of the questions given was an array of  $n$  distinct elements. The task is to find all elements in the array which have at least two greater elements than themselves. But it's in the syllabus of his exam. So can you help to create a program in the specified concept to get an offer in the next interview ?.

Constraints  
1 ≤  $N$  ≤ 1000

Examples:

Input : A[] = {2, 8, 7, 1, 5};  
Output : 1 2 5

The output has three elements which have two or more greater elements.

Input : A[] = {7, -2, 3, 4, 9, -1};  
Output : -2 -1 3 4

Input:

The first line of input contains an integer  $T$  denoting the number of test cases. Each test case contains two lines. The first line of input contains an integer  $n$  denoting the size of the array. Then in the next are  $n$  space-separated values of the array.

```
#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;
}

```

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

Course	DS	Session	Sorting	Question Information
				Level 1 • Challenge 19

**Question Description:**  
Simon is studying B.Tech-Mechanical Engineering.  
He's going to attend a computer science-based subject exam this semester.  
Due to the less preparation in the previous monthly tests, his internal mark decreased.  
His computer science Professor made an offer one more chance to boost up his internal marks.  
Professor assigns a program to Simon for the internal mark bootup.

So Simon wants to solve the given task is, Given two arrays, A and B, of equal size n,  
the task is to find the minimum value of  $A[0] * B[0] + A[1] * B[1] + \dots + A[n-1] * B[n-1]$ ,  
where shuffling of elements of arrays A and B is allowed.  
can you help him in solving Questions ?

**Constraints:**  
 $1 \leq T \leq 100$   
 $1 \leq N \leq 50$   
 $1 \leq A[i] \leq 20$

**Problem**

**Input Format:**  
The first line of input contains an integer denoting the no of test cases.  
Then T test cases follow. Each test case contains three lines.

```

#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

The screenshot shows a web browser window with the URL [care.srmup.in/srmncretelab/#/srmncretelab/student/home](http://care.srmup.in/srmncretelab/#/srmncretelab/student/home). The page displays a challenge titled "DSA ARRAYS". A green banner at the top says "You have already solved this challenge! Though you can run the code with different logic!". The challenge details are as follows:

- Course:** DS
- Session:** Session
- Arrays:** Question Information
- Level:** Level 1
- Challenge:** Challenge 2

**Problem Description:**  
Rigesh is an electronic shop owner. Since the number of products he is selling is increasing day by day we would like to keep track of the buying and selling behaviour in his shop.  
So given the cost of stock on each day in an array A[] of size N. Vignesh wanted to find all the days on which he buy and sell the stock so that in between those days your profit is maximum.

**Constraints:**  
1 ≤ n ≤ 10  
1 ≤ n ≤ 10

**Problem**

**Input Format:**  
First line contains number of test cases T.  
First line of each test case contains an integer value N denoting the number of days, followed by an array of stock prices of N days.

**Output Format:**  
For each testcase, output all the days with profit in a single line.  
If there is no profit then print "No Profit".

**Logical Test Cases**

Test Case 1	Test Case 2
INPUT [STDIN] 1 5 130 204 150 175 148	INPUT [STDIN] 2 5 131 124 118 175 118

```
#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)
            sol[count].buy = arr[i];
        if(i < n-1)
            sol[count].sell = arr[i+1];
        count++;
    }
}
```

```

    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);
    }
}

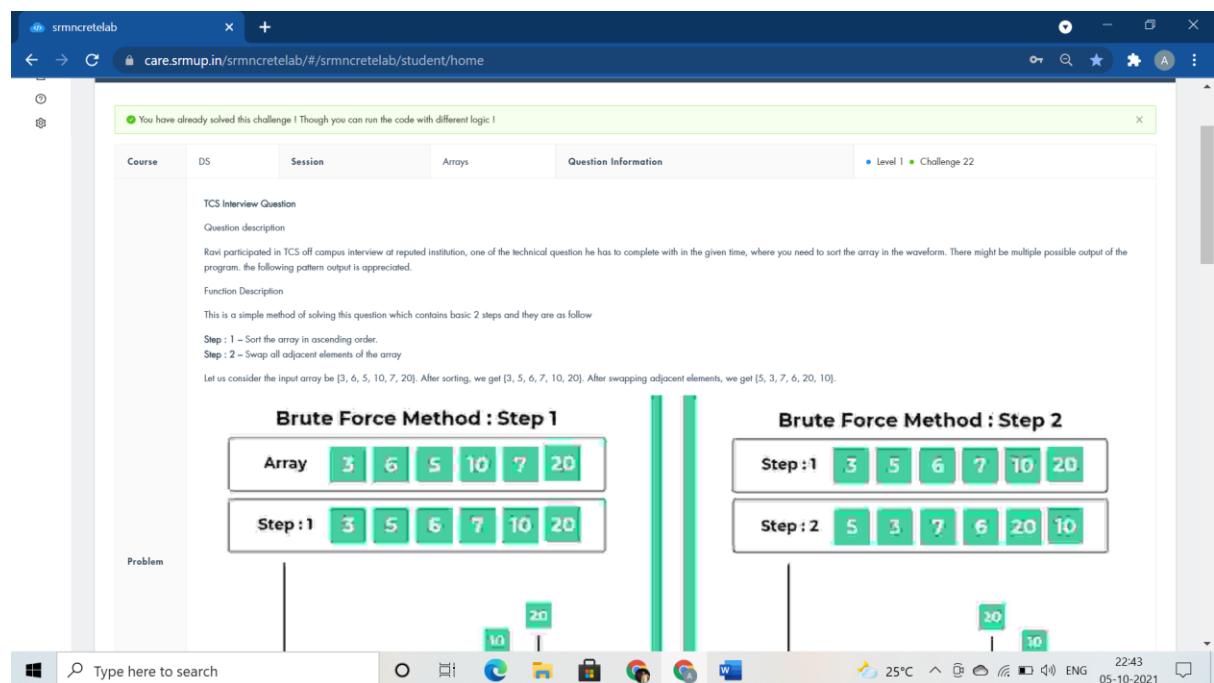
```

```

printf("\n");
t--;
}

return 0;
}

```



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

Course DS Session Arrays Question Information Level 1 Challenge 22

TCS Interview Question

Question description

Ravi participated in TCS off campus interview at reputed institution, one of the technical question he has to complete with in the given time, where you need to sort the array in the waveform. There might be multiple possible output of the program. the following pattern output is appreciated.

Function Description

This is a simple method of solving this question which contains basic 2 steps and they are as follow

Step : 1 – Sort the array in ascending order.  
Step : 2 – Swap all adjacent elements of the array

Let us consider the input array be [3, 6, 5, 10, 7, 20]. After sorting, we get [3, 5, 6, 7, 10, 20]. After swapping adjacent elements, we get [5, 3, 7, 6, 20, 10].

**Brute Force Method : Step 1**

Array	3	6	5	10	7	20
Step : 1	3	5	6	7	10	20

**Brute Force Method : Step 2**

Step : 1	3	5	6	7	10	20
Step : 2	5	3	7	6	20	10

```

#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;
}
```

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

CHALLENGE INFORMATION

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

Course DS Session Arrays Question Information Level 1 Challenge 24

Question description  
Sajid is a First year student in reputed institution.  
Although he scored well in many subjects, he did not an expert in Algorithms.  
But Sajid's computer examination is scheduled for next week.  
As per the blueprint, many questions would come from the Arrays topic.  
He collected previous year's questions, one of the repeated questions is you need to reverse the array in C Programming Language.  
Can you help him?

Function Description

**Algorithm**

Start  
Input -> n  
Input -> elements of array  
Start loop (i) for 0 to n/2  
exchange

**Sample Test Case : 1**  

1	2	3	4	5
↓				
5	4	3	2	1

**Sample Test Case : 2**

25°C 22:46 05-10-2021

```
#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;  
}
```

The screenshot shows a web browser window titled "srmncretelab" with the URL "care.srmup.in/srmncretelab/#/srmncretelab/student/home". The main content area is titled "CHALLENGE INFORMATION" and displays the following details:

- Course:** DS
- Session:** Arrays
- Question Information:** Level 1 • Challenge 26

**Problem Description:** Ram has provide inputs two numbers 'p' and 'q' to Sakshi. He wants to creates a matrix of size  $p \times q$  [ $p$  rows and  $q$  columns] in which every elements is either Y or 0. The Ys and 0s must be filled alternatively, the matrix should have outermost rectangle of Ys, then a rectangle of 0s, then a rectangle of Ys, and so on..

**Constraints:**  
 $1 \leq p, q \leq 1000$

**Input Format:**  
Input lines must be how many rows and columns in that matrix, also values must be separate space.

**Output Format:**  
Print the output in a separate lines.

**Logical Test Cases:**

Test Case 1	Test Case 2
INPUT [STDIN] 6 7	INPUT [STDIN] 5 8
EXPECTED OUTPUT Y Y Y Y Y Y Y Y 0 0 0 0 Y Y Y Y Y Y Y Y	EXPECTED OUTPUT Y Y Y Y Y Y Y Y Y 0 0 0 0 0 Y Y Y Y Y Y Y Y Y

```
#include <bits/stdc++.h>  
  
using namespace std;  
  
void ss(){  
  
    cout<<"while(top<=bottom && right>=left)";  
}  
  
void fillOX(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;
    fillOX(m, n);
}

```

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

**Course** DS **Session** Arrays **Question Information** Level 1 Challenge 27

**Problem Description:**  
Nirobi have given a matrix C of size N x M to Rio.  
Also Rio are given position of submatrix as X1, Y1 and X2, Y2 inside the matrix.  
Now Rio needs to find the sum of all elements inside that submatrix.  
Can you help Rio in completing the task assigned by Nirobi.

**Constraints:**  
 $1 \leq T \leq 15$   
 $1 \leq N, M \leq 103$   
 $1 \leq C[N][M] \leq 106$   
 $1 \leq X1, Y1, X2, Y2 \leq M$

**Input Format:**  
The first line of input contains an integer T denoting the number of test cases.  
The first line of each test case is n and m,n is the number of rows and m is the number of columns.  
The second line of each test case contains C[N][M].  
The third line contains four value of X1, Y1, X2, Y2. X1, Y1 is the top left cell and X2, Y2 is the bottom right cell.

**Output Format:**  
Print the sum of all elements inside that submatrix.

Logical Test Cases

```
#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];
}

}

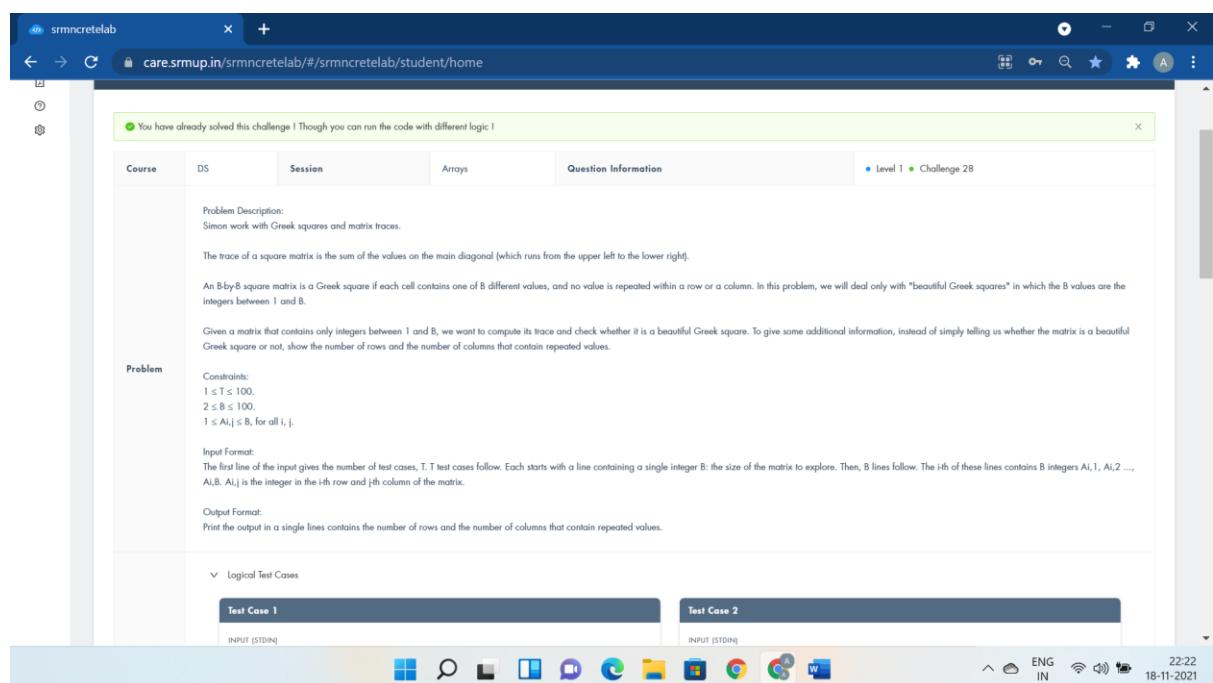
cout<<sum<<"\n";

}

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];";
}

```

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

**Course** DS **Session** Arrays **Question Information** Level 1 Challenge 29

**Problem Description:**  
Good news! Suresh get to go to America on a class trip! Bad news, he don't know how to use the Dollar which is the name of the American cash system. America uses coins for cash a lot more than the Kuwait does. Dollar comes in coins for values of: 1, 2, 10, 50, 100, & 500 To practice your Dollar skills, suresh have selected random items from Amazon.co.us and put them into a list along with their prices in Dollar. Suresh now want to create a program to check suresh Dollar math.

Suresh goal is to maximize your buying power to buy AS MANY items as you can with your available Dollar.

**Input Format:**  
File listing 2 to 6 items in the format of:  
ITEM DDDDD  
ITEM – the name of the item you want to buy  
DDDDD – the price of the item (in Dollar)

**Output Format:**  
Print the output in a separate lines contains, List the items suresh can afford to buy. Each item on its own line. Suresh goal is to buy as many items as possible. If suresh can only afford the one expensive item, or 2 less expensive items on a list, but not all three, then list the less expensive items as affordable. If suresh cannot afford anything in the list, output "I need more Dollar!" after the items. The final line you output should be the remaining Dollar he will have left over after make purchases.

Logical Test Cases

Test Case 1	Test Case 2
INPUT [STDIN] 6000 3 Phone-case 1480 Candybar 863 Sunglasses 5529	INPUT [STDIN] 2100 3 Camera 69555 TV 76439 iPhone 90000

```
#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;
        }
    }

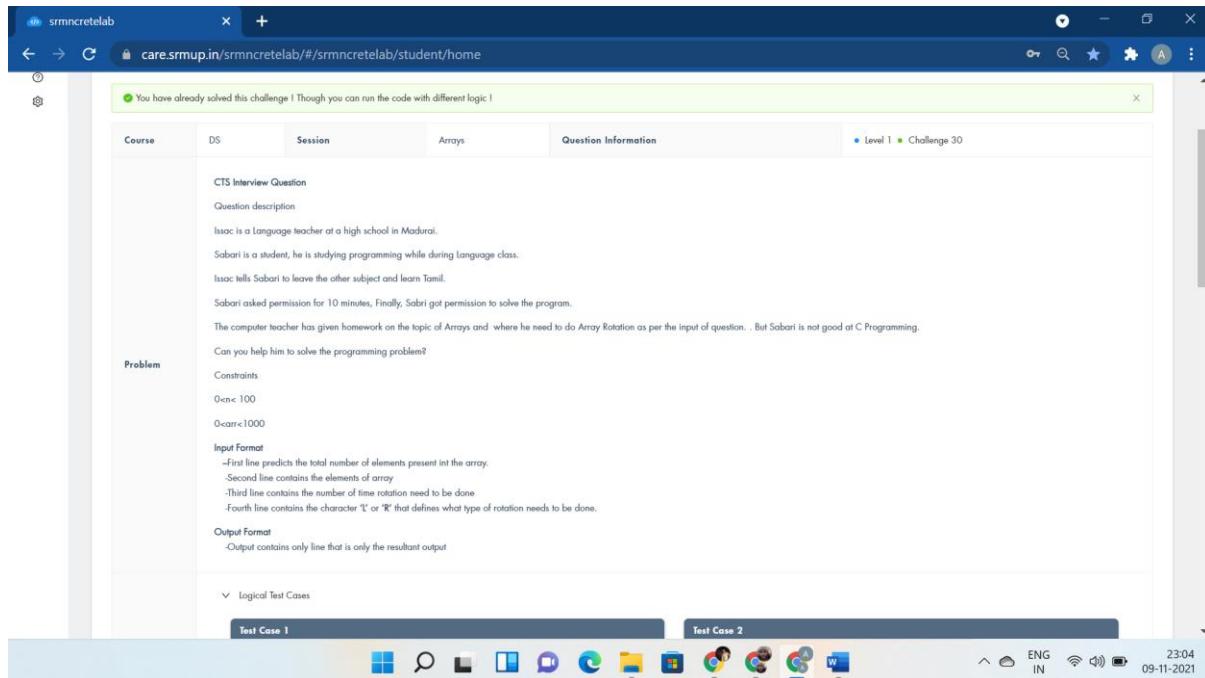
    if(count==items)

        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]<<" ";

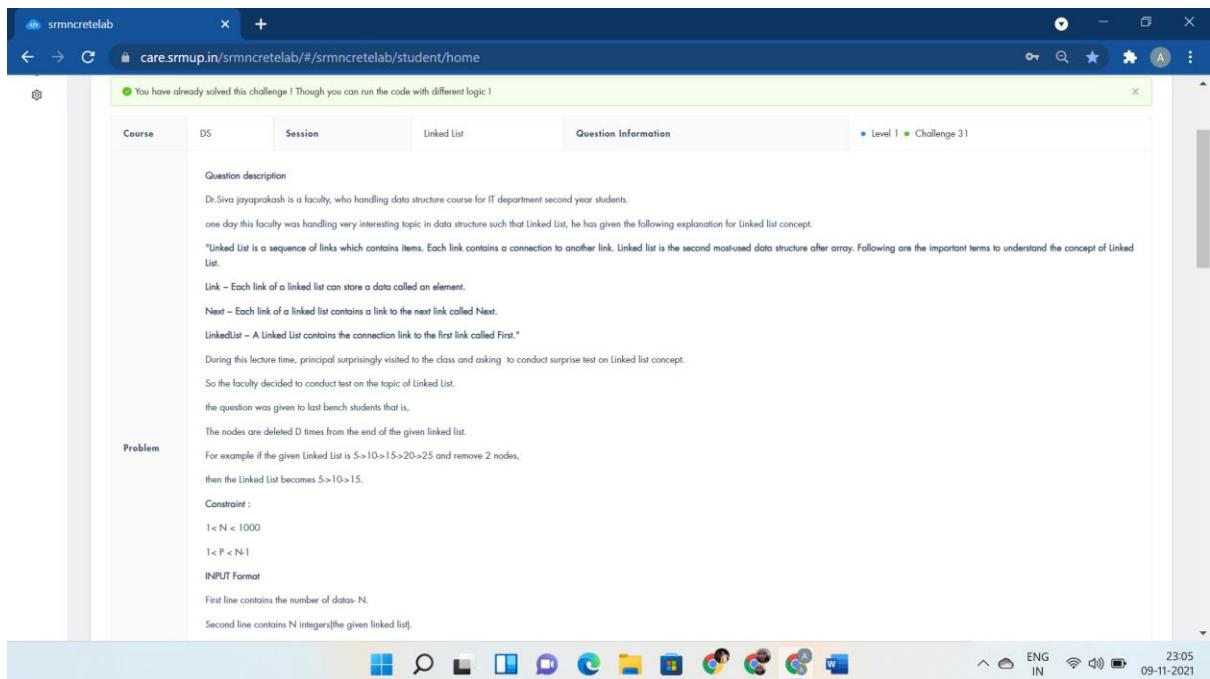
        for(int i=0;i<n-d;i++){

```

```
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(){
```

```

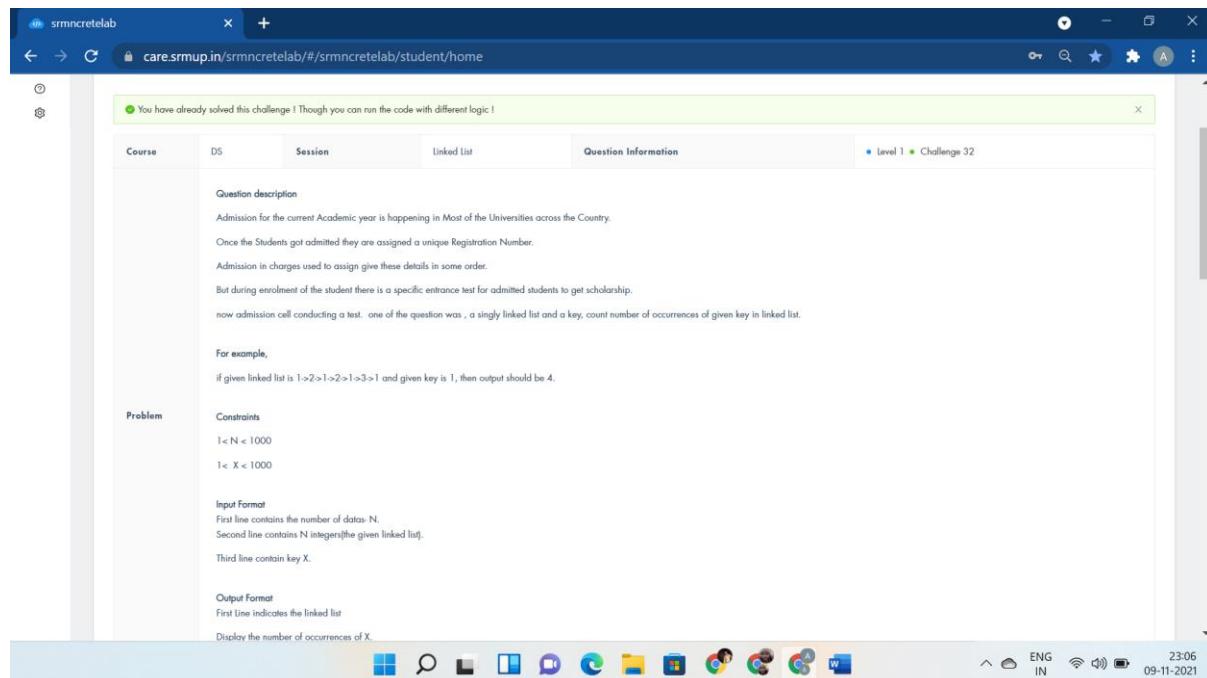
create();

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;
}

```

```

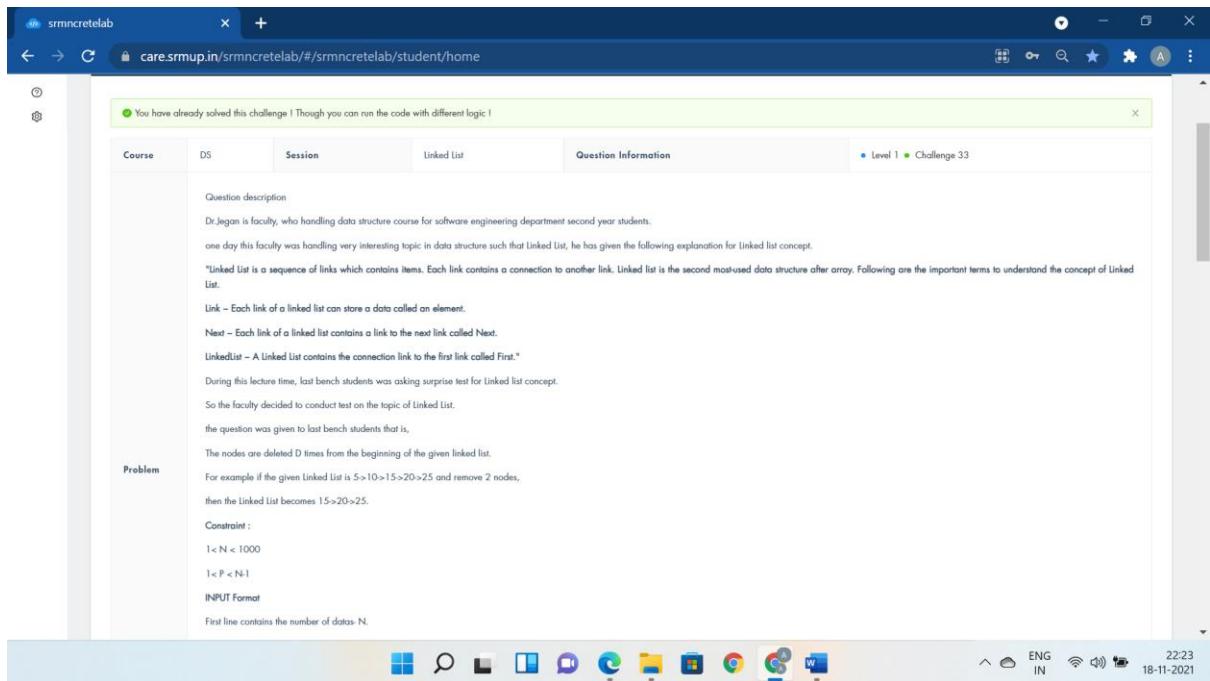
void printList(node *node){
    while (node != NULL)
    {
        cout<<"-->"<<node->key;
        node = node->next;
    }
}

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;
}

```

}



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

**Question Information**

Level 1 • Challenge 33

**Course** DS Session Linked List

**Problem**

**Question description**

Dr.Jegan is faculty, who handling data structure course for software engineering department second year students.

one day this faculty was handling very interesting topic in data structure such that Linked List, he has given the following explanation for Linked list concept.

"Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array. Following are the important terms to understand the concept of Linked List.

Link – Each link of a linked list can store a data called an element.

Next – Each link of a linked list contains a link to the next link called Next.

LinkedList – A Linked List contains the connection link to the first link called First."

During this lecture time, last bench students was asking surprise test for Linked list concept.

So the faculty decided to conduct test on the topic of Linked List.

the question was given to last bench students that is,

The nodes are deleted D times from the beginning of the given linked list.

For example if the given Linked List is 5->10->15->20->25 and remove 2 nodes,

then the Linked List becomes 15->20->25.

**Constraint :**

1 < N < 1000

1 < P < N-1

**INPUT Format**

First line contains the number of datas- N.

```
#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()";

}

```

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

Course	DS	Session	Linked List	Question Information
				Level 1 • Challenge 34

**Question description**

Once upon a time, in French Canada, there lived a fat old woman named Tante Adela. She lived alone in her barn with her large grey cat and her cows. She got up quite early one morning since it was baking day and she had a lot to accomplish. She carried a pile of wood to her oven outdoors. She ran across some old school classmates, with whom she reminisced about their school days and a mental exam competition. One of the competition's requirements was to write a C function that searches a singly linked list for a given key "x". (Iterative). If x is contained in the linked list, the function should return true; otherwise, it should return false.

For example,

If the key to be searched is 15 and linked list is 14->21->11->30->10, then function should return false. If key to be searched is 14, then the function should return true.

**Problem**

**Constraints**

1 < N < 1000  
1 < X < 1000

**Input Format**

First line contains the number of datas- N.  
Second line contains N integers[the given linked list].  
Third line contains the key X to search.

```
#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);
    }
}

```

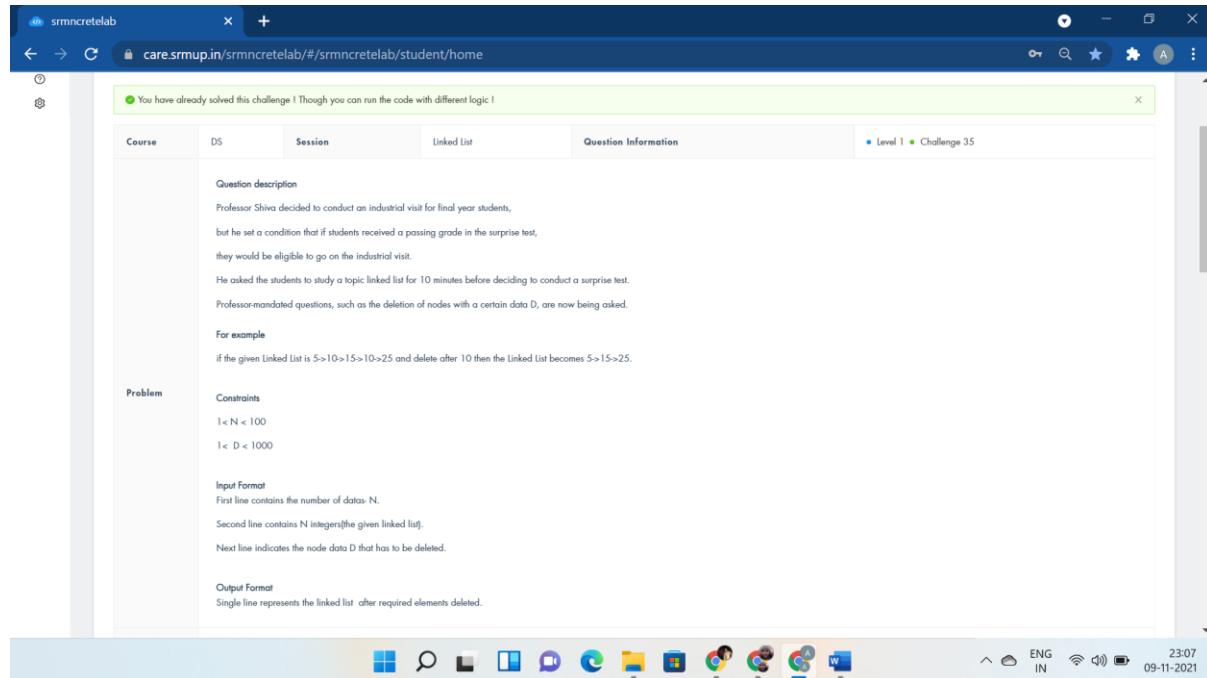
```

    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() ";

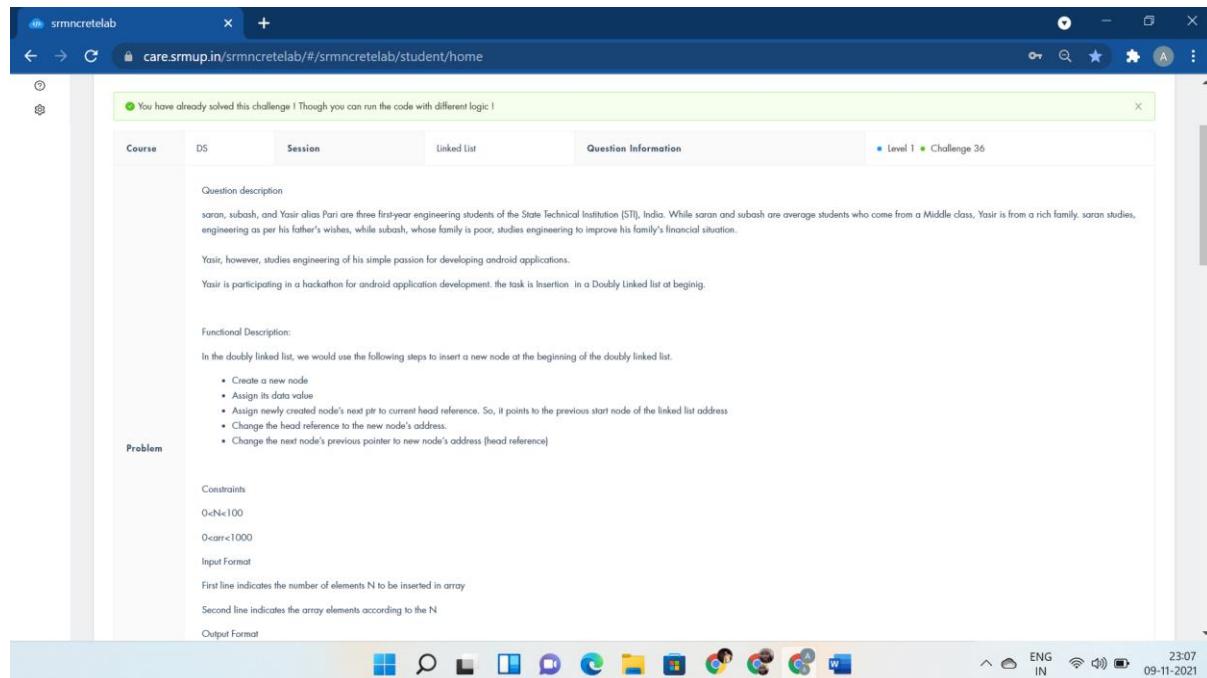
}


```

```

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;
    }
}

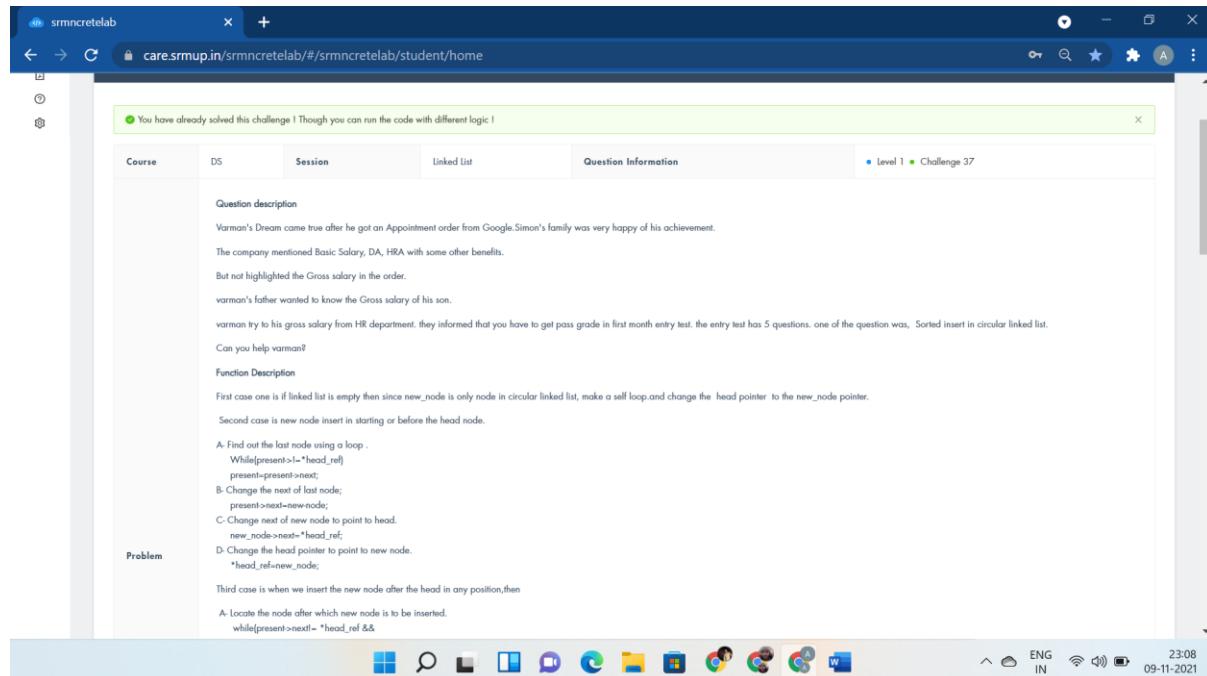
int main()
{
    struct Node* head = NULL;
    int n;
    cin>>n;
    for(int i=0;i<n;i++){
        int t;

```

```

    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);

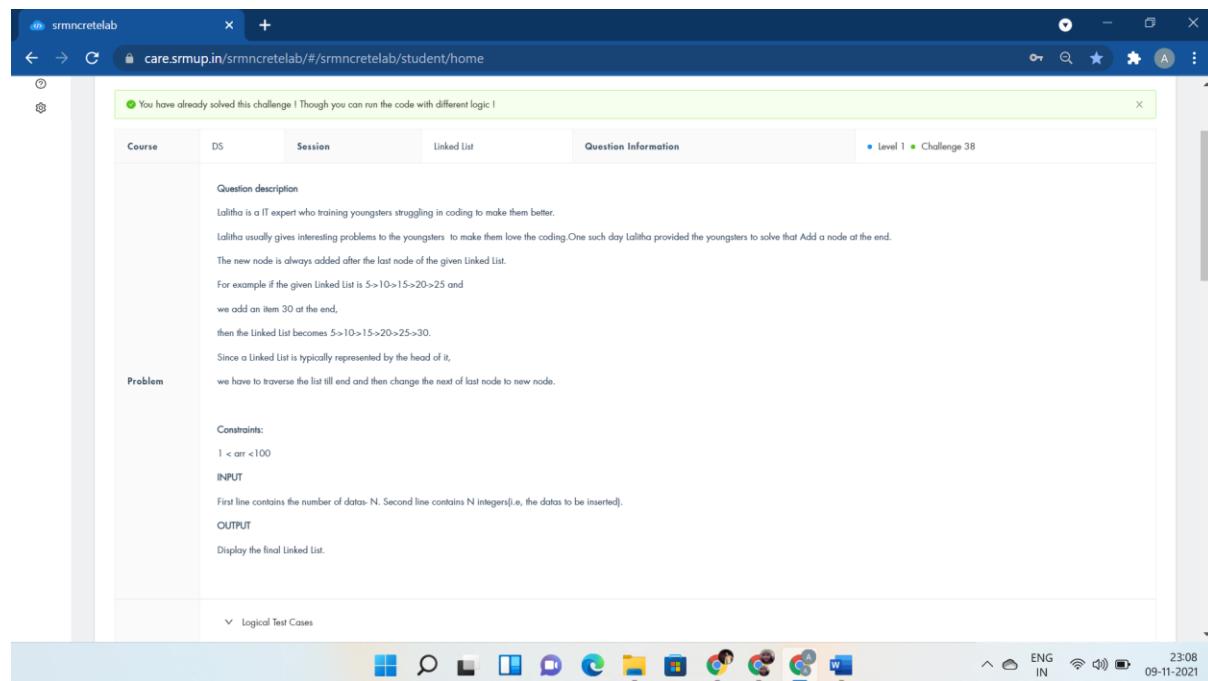
}

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)

    {

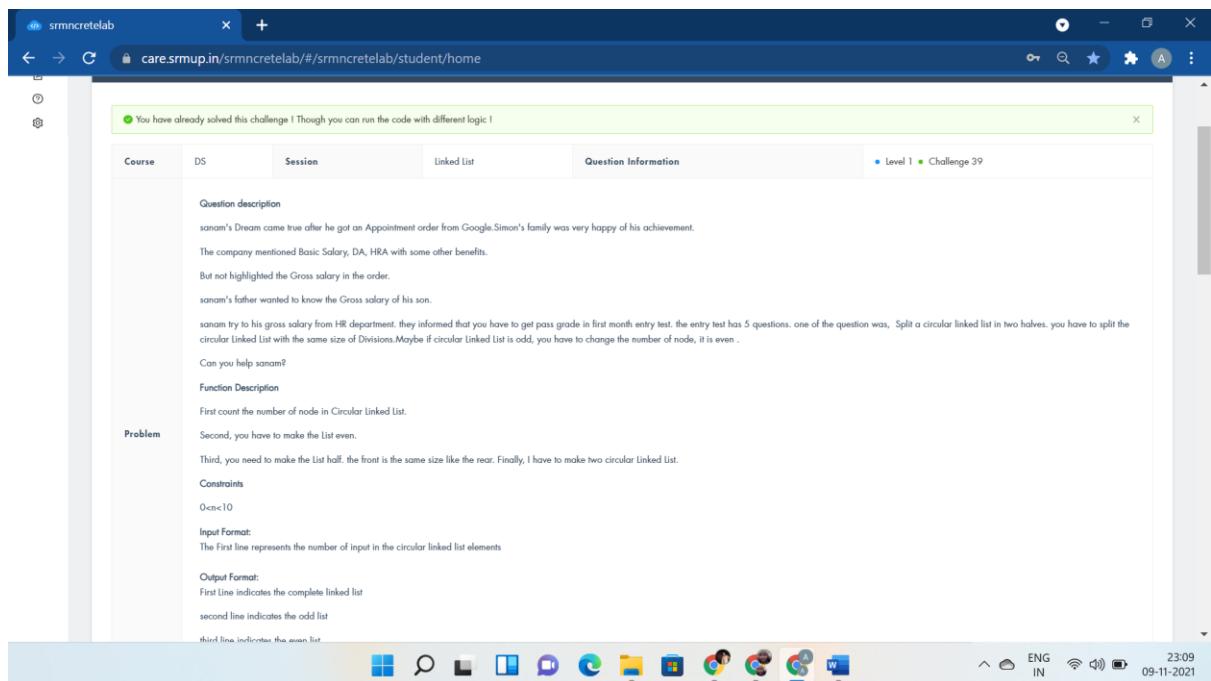
        printf("->%d",temp->data);

        temp = temp->next;

```

```
}
```

```
}
```



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

Course DS Session Linked List Question Information Level 1 Challenge 39

Question description  
sanam's Dream come true after he got an Appointment order from Google.Simon's family was very happy of his achievement.  
The company mentioned Basic Salary, DA, HRA with some other benefits.  
But not highlighted the Gross salary in the order.  
sanam's father wanted to know the Gross salary of his son.  
sanam try to his gross salary from HR department, they informed that you have to get pass grade in first month entry test, the entry test has 5 questions, one of the question was, Split a circular linked list in two halves, you have to split the circular Linked List with the same size of Divisions.Maybe if circular Linked List is odd, you have to change the number of node, it is even .  
Can you help sanam?

Function Description  
First count the number of node in Circular Linked List.

Problem  
Second, you have to make the List even.  
Third, you need to make the List half, the front is the same size like the rear. Finally, I have to make two circular Linked List.

Constraints  
 $0 < n \leq 10$

Input Format:  
The First line represents the number of input in the circular linked list elements

Output Format:  
First Line indicates the complete linked list  
second line indicates the odd list  
third line indicates the even list

```
#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)

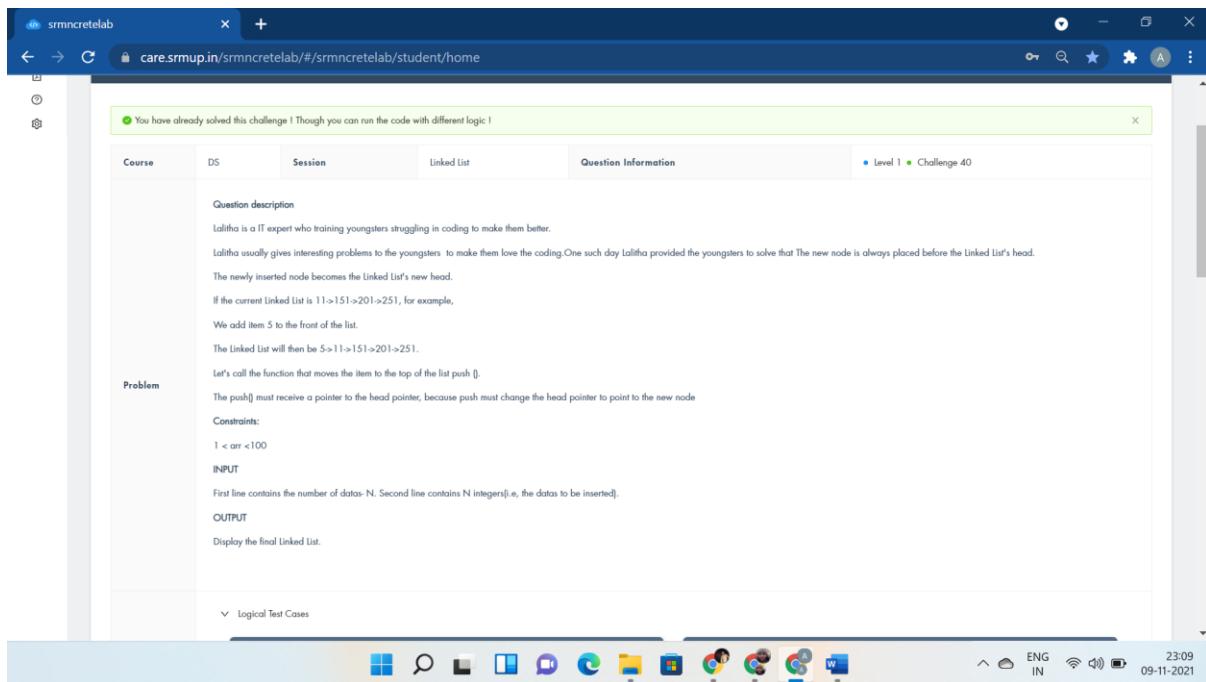
```

```

{
    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;
}

```

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

Course	DS	Session	Stack	Question Information
				Level 1 • Challenge 42

**Problem Description:**  
Arumugam is in the process of reorganising her library. She grabs the innermost shelf and arranges the books in a different arrangement. She shatters the shelf's walls. There will be no shelf barriers and simply books in the end. Make a printout of the book order.

**Constraints:**  
 $2 \leq |S| \leq 10^3$

**Input format:**  
The first line contains string  $S$  displaying her library.

**Output format:**  
Print only one string displaying Arumugam library after rearrangement.

**Note:**  
The first character of the string is ' $/$ ' and the last character of the string is ' $\backslash$ ' indicating outermost walls of the shelf.

**Logical Test Cases**

Test Case 1	Test Case 2
INPUT [STDIN] <code>/u/hate\1\</code>	INPUT [STDIN] <code>/u/slap\1\</code>
EXPECTED OUTPUT	EXPECTED OUTPUT

```

#include <bits/stdc++.h>

using namespace std;

int main()

{

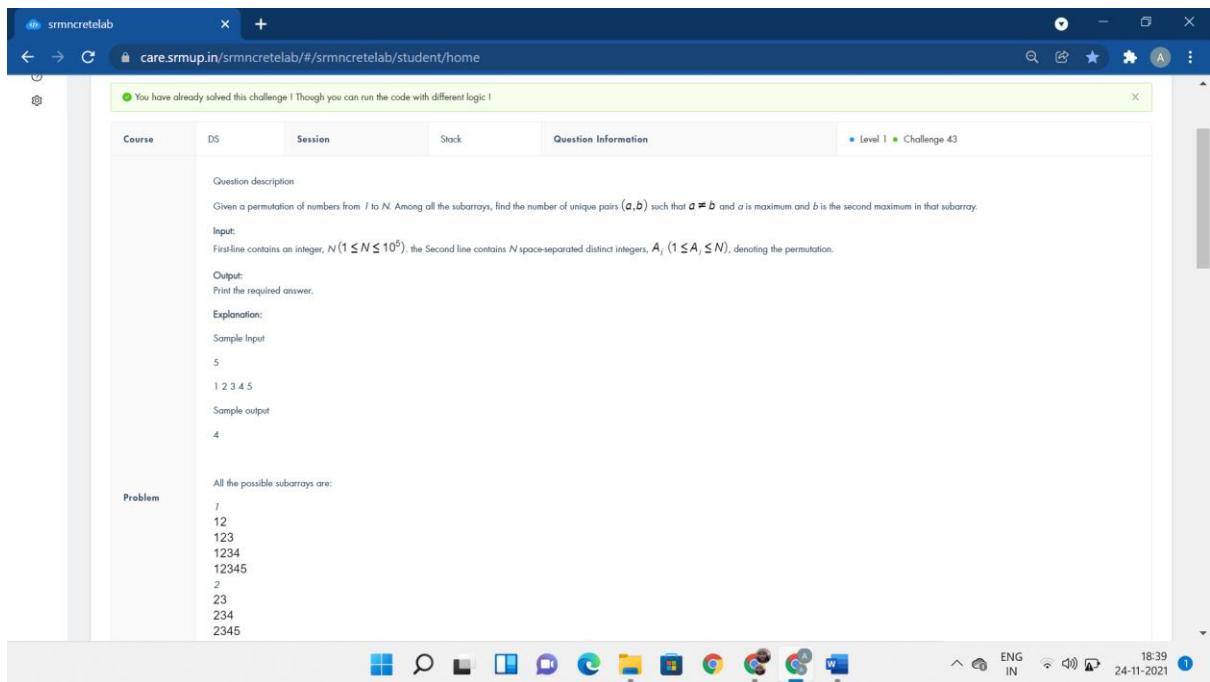
    string s,temp="";
    cin>>s;
    stack<string> stk;
    for (unsigned int i = 0; i < s.size(); i++) {
        if(s[i]==47 || s[i]==92){
            if(!temp.empty()){
                stk.push(temp);
                temp.clear();
            }
        }
        else{
            temp.push_back(s[i]);
        }
    }

    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;
}
```

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

**Course** DS **Session** **Stack** **Question Information** Level 1 Challenge 44

**Problem Description:**  
You are given an array  $A$  of  $Q$  integers and  $Q$  queries. In each query, you are given an integer  $i$  ( $1 \leq i \leq N$ ). Your task is to find the minimum index greater than  $i$  ( $1 \leq i \leq N$ ) such that:  
1. Sum of digits of  $A_i$  is greater than the sum of digits of  $A_j$   
2.  $A_i < A_j$

If there is no answer, then print -1.

**Constraints**  
 $1 < N, Q \leq 10^4$   
 $1 \leq A_i \leq 10^9$   
 $1 \leq Q_i \leq N$

**Input format**  
The first line contains two numbers  $N$  and  $Q$ .  
The next line contains  $N$  numbers.  
Next  $Q$  lines contain  $Q$  queries.

**Output format**  
Print the answer as described in the problem

Logical Test Cases

18:40 24-11-2021

```
#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;

        if(Q>n) cout<<-1<<endl;

        else{

            int ans=-1;

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

                if(arr[i]>arr[Q] && a[i]<a[Q]){

                    ans=i;

                    break;

                }

            }

            cout<<ans<<endl;

        }

    }

}
```

```

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<<' ';

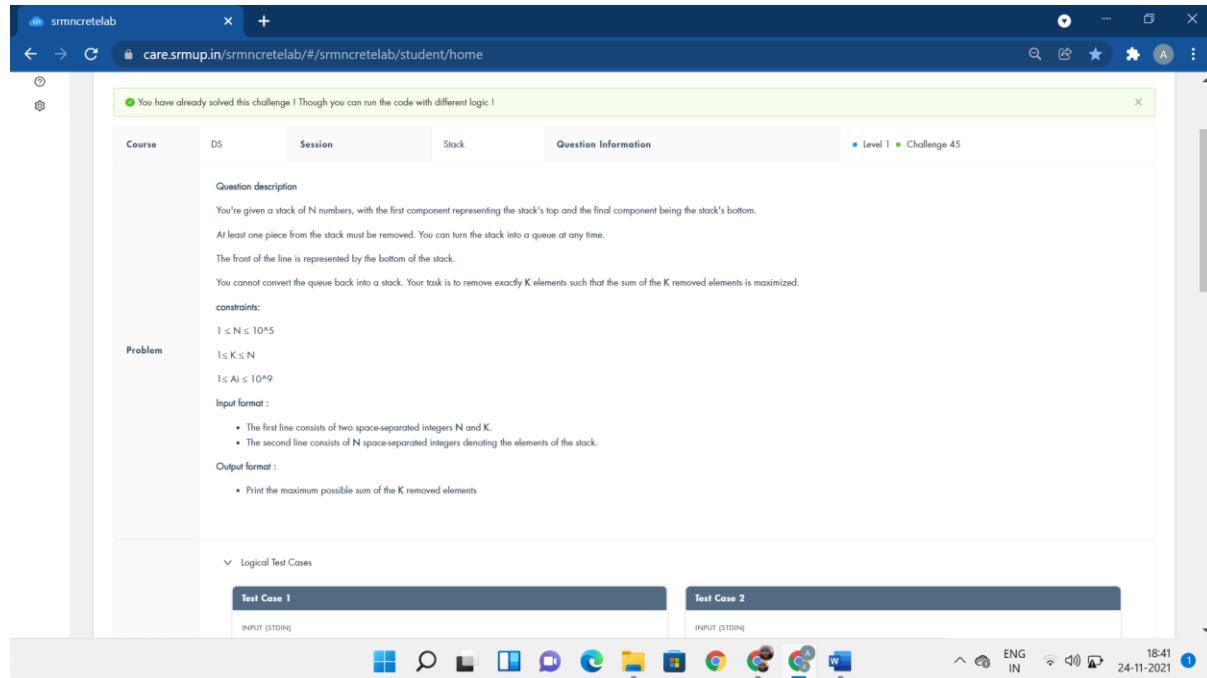
}

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;
}

```

The screenshot shows a browser window with the URL [care.srmup.in/srmncretelab/#/srmncretelab/student/home](http://care.srmup.in/srmncretelab/#/srmncretelab/student/home). A green notification bar at the top says "You have already solved this challenge ! Though you can run the code with different logic !". The main content area has tabs for "Course", "DS", "Session", "Stack", and "Question Information". The "Question Information" tab is active, showing "level 1" and "Challenge 46". Below this, there's a "Problem Description" section with instructions about arrays, queues, and stacks. It includes sample data and rules for generating arrays. The "Problem" section contains a code snippet and constraints. At the bottom, there are system status icons for battery, signal, and network.

```

#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;
}

```

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

**Course** DS **Session** Stack **Question Information** Level 1 Challenge 48

**Problem**

**Functional Description:**

- Read the Prefix expression in reverse order [from right to left]
- If the symbol is an operand, then push it onto the Stack
- If the symbol is an operator, then pop two operands from the Stack
- Create a string by concatenating the two operands and the operator after them.  
string = operand1 + operand2 + operator
- Push the resultant string back to Stack

**Repeat the above steps until end of Prefix expression.**

**Constraints**

the input should be a expressions

**Input Format**

Single line represents the prefixed expressions

**Output Format**

Single line represents the postfix expression

**Logical Test Cases**

Test Case 1 Test Case 2

ENG IN 09-11-2021 23:10

```
#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]))
        {
```

```

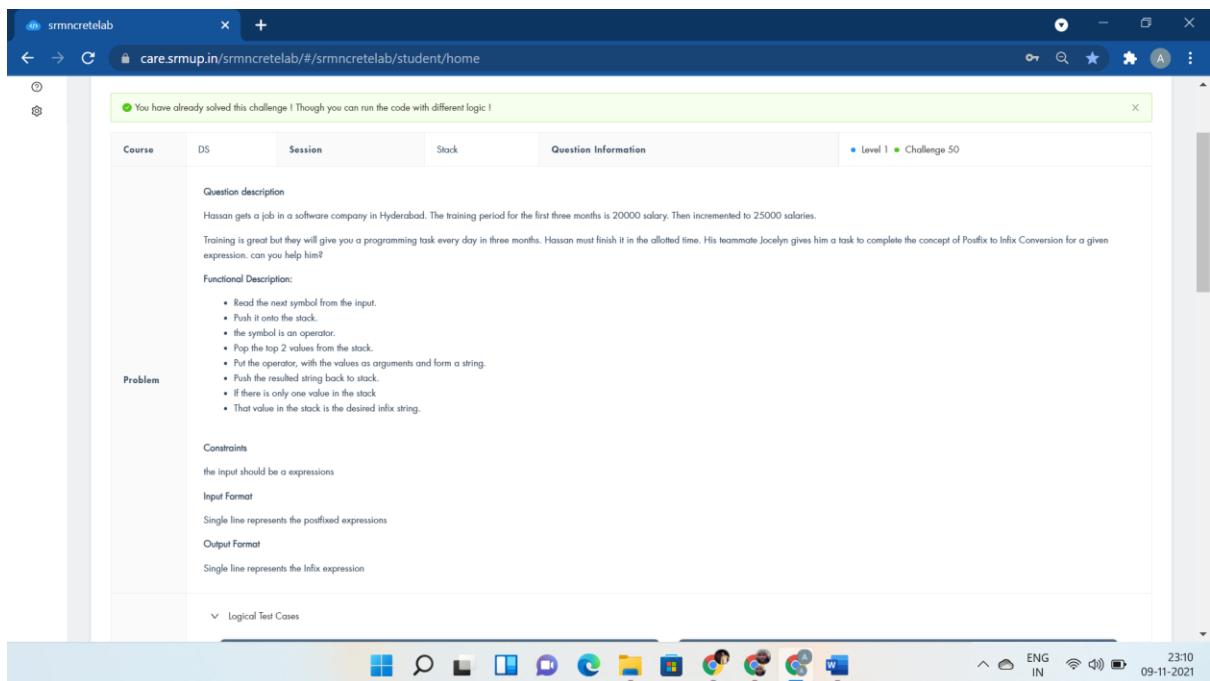
        string op1 = s.top();
        s.pop();
        string op2 = s.top();
        s.pop();
        string temp = op1 + op2 + pre_exp[i];
        s.push(temp);
    }
}

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:-

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

Course	DS	Session	Queue	Question Information
				Level 1   Challenge 51

**Question description**

Sathya is an DS expert training youngsters struggling in DS to make them better. Sathya usually gives interesting problems to the youngsters to make them love the DS.

One such day Sathya provided to the youngsters to solve the task such that, insert an element in a Queue in FIFO order. Youngsters were lacking the idea to solve the problem.

Being an exciting youngster can you solve it?

**Function Description**

1. Define the maximum size of queue and initialize front and rear as -1.
2. In the main function we will initialize two variables that will store the data and the size of the queue.
3. Accept the data that we want to enter in a queue using a for loop.
4. After accepting the data use enqueue() function to insert the data in a queue.

```

#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--)");
}

```

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

Course DS Session Queue Question Information Level 1 Challenge 53

Question description

Consider the following string transformation:

- append the character # to the string [we assume that # is lexicographically smaller than all other characters of the string]
- generate all rotations of the string
- sort the rotations in increasing order
- based on this order, construct a new string that contains the last character of each rotation

For example, the string babc becomes babc#. Then, the sorted list of rotations is #babc, abc#b, babc#, bc#ba, and c#bab. This yields a string cb#ab.

Problem Constraints

- 1 ≤ n ≤ 10^6

Input

The only input line contains the transformed string of length n+1. Each character of the original string is one of a-z.

Output

Print the original string of length n.

Logical Test Cases

Test Case 1	Test Case 2
INPUT (STDIN) cb#pb	INPUT (STDIN) cad#abc
EXPECTED OUTPUT	EXPECTED OUTPUT

Windows taskbar: 22:29, 18-11-2021, ENG IN, battery icon

```
#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;
}

```

Course	DS	Session	Queue	Question Information
				Level 1 • Challenge 54

**Question description**

Joe root is a Placement trainer. he is working as CDC trainer in reputed institution that during training the youngsters are struggling in queue concept. Joe root usually gives interesting problems to the students to make them love the DS.

One such day Joe root provided to the final year students to solve the task such that, Queue implementation with arrays as using linked list for implementing queue and delete an element from the queue using linked list concept.

Queue data structures work on the FFO architecture so the element that has entered first in the list will go out from the list first.

Final Year students were lacking the idea to solve the problem.

Being an exciting youngster can you solve it?

**Function Description**

enqueue(data)	dequeue()	print()
<ul style="list-style-type: none"> <li>Build a new node with given data.</li> <li>Check if the queue is empty or not.</li> <li>If queue is empty then assign new node to front and rear.</li> <li>Else make next of rear as new node and rear as new</li> </ul>	<ul style="list-style-type: none"> <li>Check if queue is empty or not.</li> <li>If queue is empty then dequeue is not possible.</li> <li>Else store front in temp</li> <li>And make next of front as front.</li> </ul>	<ul style="list-style-type: none"> <li>Check if there is some data in the queue or not.</li> <li>If the queue is empty print "No data in the queue."</li> <li>Else define a node pointer and initialize it with front.</li> <li>Print data of node pointer until the next of node pointer</li> </ul>

```

#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);

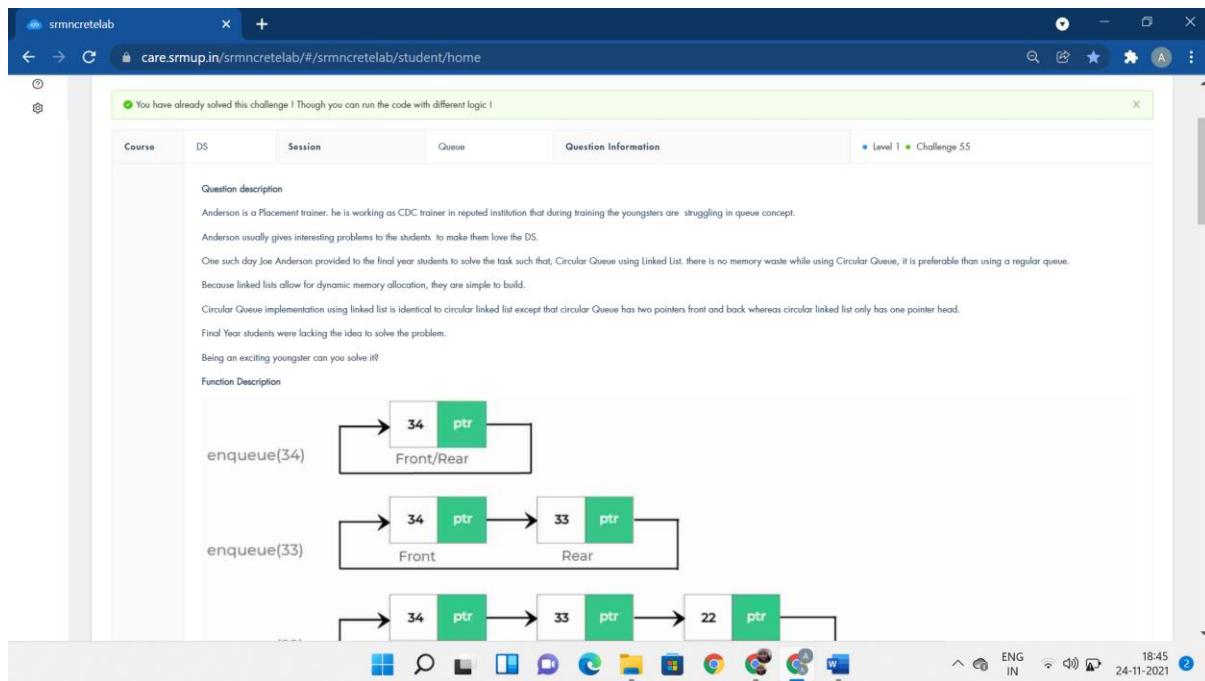
    dequeue();

    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);

    }

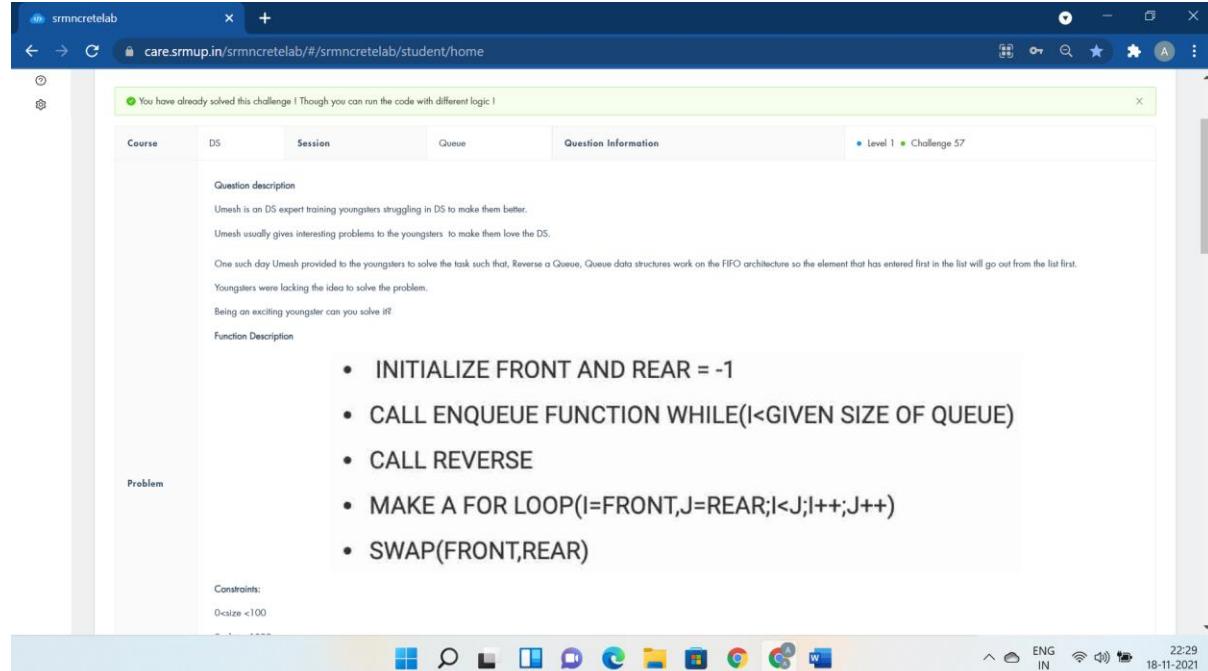
    for(i=0;i<n;i++){

        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);
    }
}

```

```

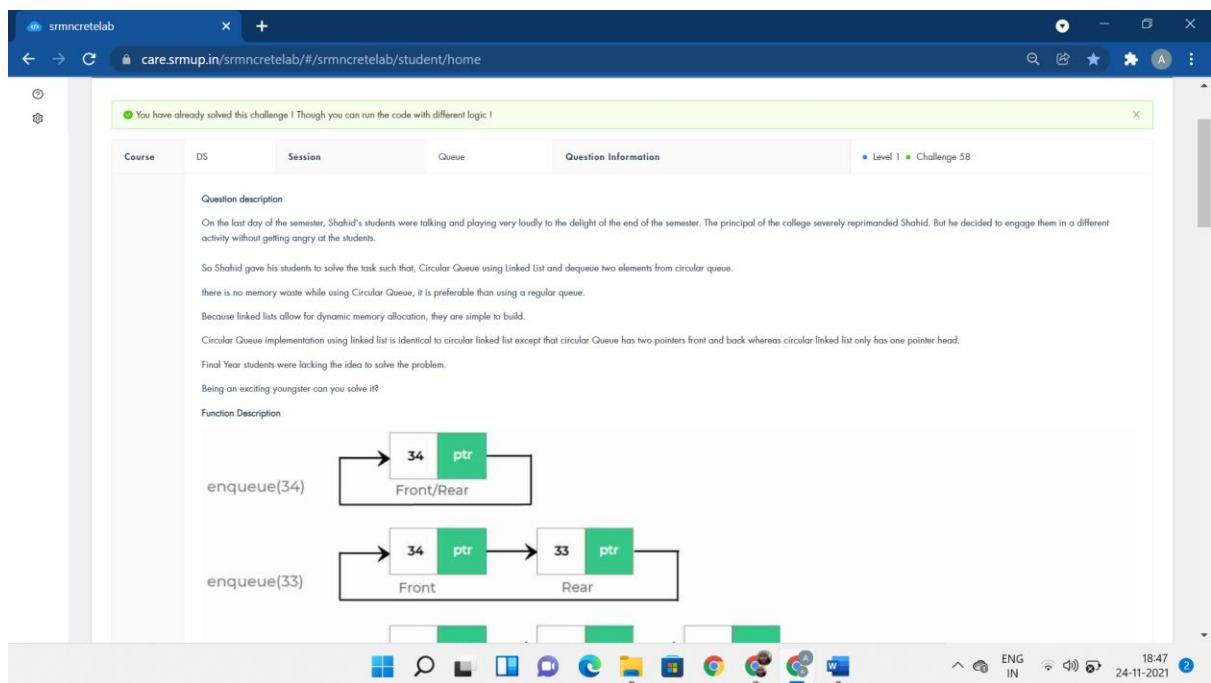
}

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;  
}
```

Question description

Your task is to construct a tower in  $N$  days by following these conditions:

- Every day you are provided with one disk of distinct size.
- The disk with larger sizes should be placed at the bottom of the tower.
- The disk with smaller sizes should be placed at the top of the tower.

The order in which tower must be constructed is as follows:

- You cannot put a new disk on the top of the tower until all the larger disks that are given to you get placed.

Print  $N$  lines denoting the disk sizes that can be put on the tower on the  $i^{\text{th}}$  day.

Constraints:

$1 \leq N \leq 10^6$

$1 \leq \text{size} \leq N$

Input format

- First line:  $N$  denoting the total number of disks that are given to you in the  $N$  subsequent days.
- Second line:  $N$  integers in which the  $i^{\text{th}}$  integers denote the size of the disks that are given to you on the  $i^{\text{th}}$  day.

Note: All the disk sizes are distinct integers in the range of 1 to  $N$ .

Output format

Print  $N$  lines. In the  $i^{\text{th}}$  line, print the size of disks that can be placed on the top of the tower in descending order of the disk sizes.

If on the  $i^{\text{th}}$  day no disks can be placed, then leave that line empty.

Logical Test Cases

Test Case 1    Test Case 2

22:31  
18-11-2021

```
#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:-

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

Course	DS	Session	Tree 1	Question Information
				Level 1 · Challenge 61

**Question description**  
You are given an  $n \times n$  grid representing the map of a forest. Each square is either empty or contains a tree. The upper-left square has coordinates  $(1, 1)$ , and the lower-right square has coordinates  $(n, n)$ .  
Your task is to process  $q$  queries of the form: how many trees are inside a given rectangle in the forest?

**Constraints**

- $1 \leq n \leq 1000$
- $1 \leq q \leq 10^4$
- $1 \leq y_1 \leq y_2 \leq n$
- $1 \leq x_1 \leq x_2 \leq n$

**Problem**

**Input**  
The first input line has two integers  $n$  and  $q$ : the size of the forest and the number of queries.  
Then, there are  $n$  lines describing the forest. Each line has  $n$  characters:  $.$  is an empty square and  $*$  is a tree.  
Finally, there are  $q$  lines describing the queries. Each line has four integers  $y_1, x_1, y_2, x_2$  corresponding to the corners of a rectangle.

**Output**  
Print the number of trees inside each rectangle.

**Logical Test Cases**

Test Case 1	Test Case 2
INPUT (STDIN)	INPUT (STDIN)

```

#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;

    }

}

```

```

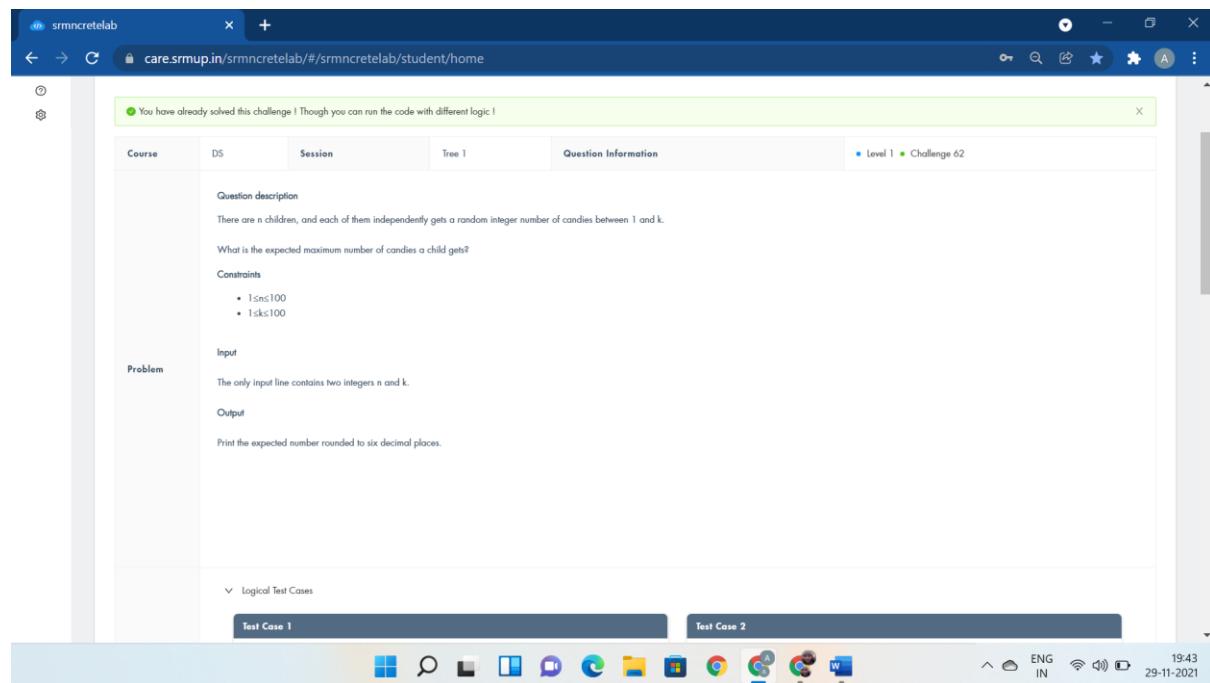
cout<<dp[y2][x2]+ dp[y1-1][x1-1] - dp[y2][x1-1] - dp[y1-1][x2]<<endl;
}

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) ");

```

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

**Course** DS **Session** Tree 1 **Question Information** **level 1** **Challenge 64**

**Question description**

Siva Sir students were chatting and playing quite loudly on the last day of the year, celebrating the end of the academic session. Siva sir was harshly chastised by the college's principal. But, instead of becoming enraged, he attempted to engage everyone in a different task.

So Siva sir gave his students to solve the task such that, you have to perform in-order tree traversal in Binary search tree.

**How Inorder works (Manually)**

- The direction of traversal for inorder is anti-clockwise
- Rule followed is LCR (Left-Center-Right)

This basically means, that we first try to visit bottommost, the left node then central node and then right and then move our way up to the tree.

**Inorder Traversal in Binary Tree**

**Constraints:**  
0 < size < 100  
0 < data < 1000  
**Input format:**

```
#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);
    }
}

```

```

}

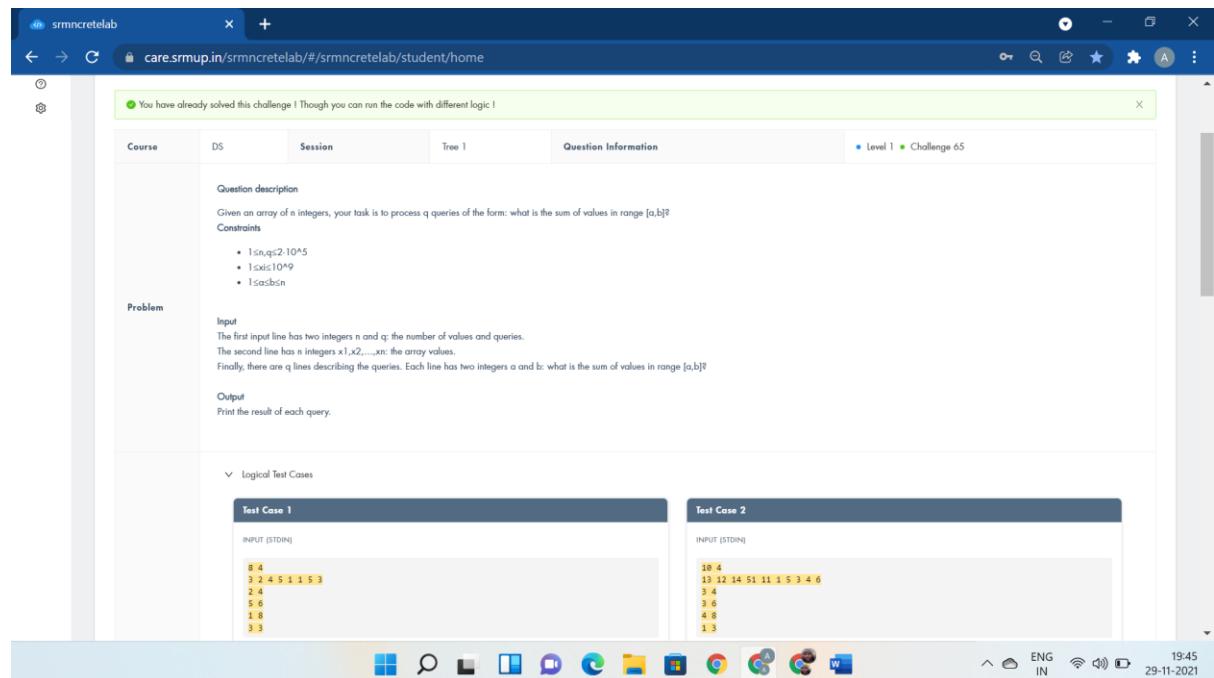
}

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>
```

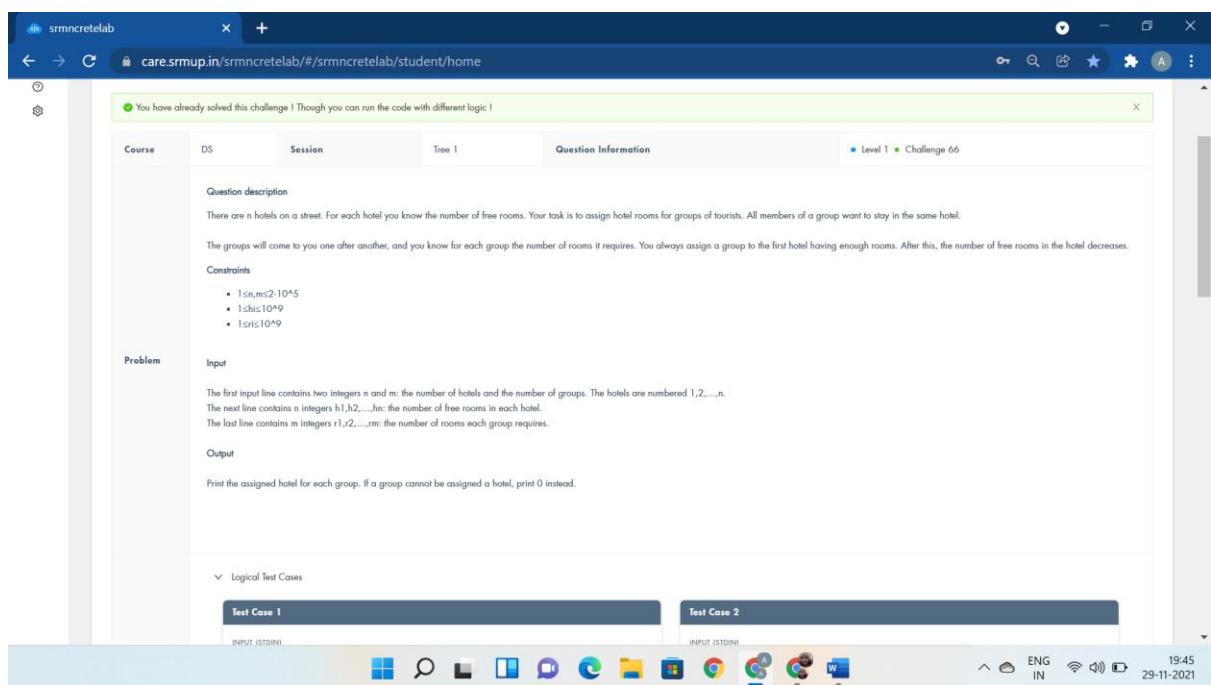
```
using namespace std;
```

```
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;
    }
}

```

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

**Question Information**

Level 1 • Challenge 68

**Course** DS **Session** Tree 1

**Question description**

A forest is an undirected graph without cycles (not necessarily connected).

Mohana and John are friends in Kerala, both of them have a forest with nodes numbered from 1 to  $n$ , and they would like to add edges to their forests such that:

- After adding edges, both of their graphs are still forests.
- They add the same edges. That is, if an edge  $\{u, v\}$  is added to Mohana's forest, then an edge  $\{u, v\}$  is added to John's forest, and vice versa.

Mohana and John want to know the maximum number of edges they can add, and which edges to add.

**Constraints:**

$1 \leq n \leq 105$ ,  
 $0 \leq m_1 \leq n$   
 $m_2 < n$   
 $1 \leq u, v \leq n$ ,  $u \neq v$

**Input**

The first line contains three integers  $n$ ,  $m_1$  and  $m_2$  — the number of nodes and the number of initial edges in Mohana's forest and John's forest.

Each of the next  $m_1$  lines contains two integers  $u$  and  $v$  — the edges in Mohana's forest.

Each of the next  $m_2$  lines contains two integers  $u$  and  $v$  ( $1 \leq u, v \leq n$ ,  $u \neq v$ ) — the edges in John's forest.

**Output**

The first line contains only one integer  $k$ , the maximum number of edges Mohana and John can add.

Each of the next  $k$  lines contains two integers  $u$  and  $v$  ( $1 \leq u, v \leq n$ ,  $u \neq v$ ) — the edge you add each time.

If there are multiple correct answers, you can print any one of them.

```
#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--)";

}

```

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

Course	DS	Session	Tree 1	Question Information
				Level 1 • Challenge 69

**Question description**  
You are given a tree consisting of  $n$  nodes.  
The diameter of a tree is the maximum distance between two nodes. Your task is to determine the diameter of the tree.

**Input**  
The first input line contains an integer  $n$ : the number of nodes. The nodes are numbered 1, 2, ...,  $n$ .  
Then there are  $n-1$  lines describing the edges. Each line contains two integers  $a$  and  $b$ : there is an edge between nodes  $a$  and  $b$ .

**Output**  
Print one integer: the diameter of the tree.

**Constraints**

- $1 \leq n \leq 10^4$
- $1 \leq a, b \leq n$

**Problem**

**Example**

**Input:**

```

5
1 2
1 3
3 4
3 5

```

**Output:**

```

#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;

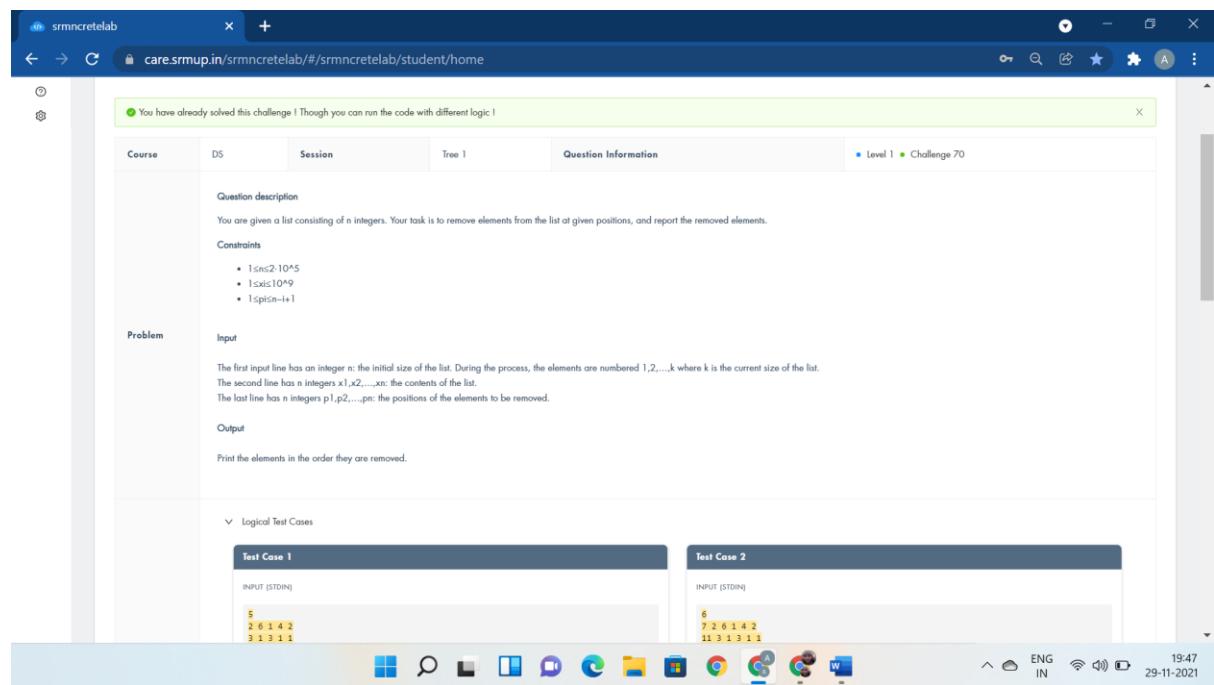
return 0;

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:-

The screenshot shows a web browser window with the URL [care.srmup.in/srmncretelab/#/srmncretelab/student/home](http://care.srmup.in/srmncretelab/#/srmncretelab/student/home). The page title is "Tree 2". A green message bar at the top says "You have already solved this challenge! Though you can run the code with different logic!". Below this, there are tabs for "Course", "DS", "Session", "Tree 2", and "Question Information". The "Question Information" tab is selected, showing "Level 1" and "Challenge 71". The main content area is titled "Problem Description" and contains the following text:

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

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

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

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

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

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

**Constraints:**  
 $1 \leq N \leq 10^6$   
 $-10^9 \leq ID_i \leq 10^9$   
 $0 \leq RC_i \leq 10^9$

```
#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;

}

```

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

Course DS Session 2 Question Information Level 1 • Challenge 72

**Problem Description:**  
Mancunian and Liverbird decide to go camping for the weekend after a long week at work. They came upon an unusual tree with  $N$  nodes while walking through a forest. From 1 to  $N$ , the vertices are numbered.  
A colour is allocated to each node in the tree [out of  $C$  possible colors]. They decide to work together [for a change] and put their reasoning abilities to the test because they are bored. At vertex 1, the tree is rooted. They aim to locate the nearest ancestor with the same hue for each node.

**Constraints**

- $1 \leq N \leq 100,000$
- $1 \leq C \leq 100,000$

**Input format**  
The first line contains two integers  $N$  and  $C$  denoting the number of vertices in the tree and the number of possible colors.  
The second line contains  $N - 1$  integers. The  $i$ th integer denotes the parent of the  $i + 1$ th vertex.  
The third line contains  $N$  integers, denoting the colors of the vertices. Each color lies between 1 and  $C$  inclusive.

**Output format**  
Print  $N$  space-separated integers. The  $i$ th integer is the vertex number of lowest ancestor of the  $i$ th node which has the same color. If there is no such ancestor, print -1 for that node.

**Logical Test Cases**

Test Case 1	Test Case 2
INPUT (STDIN) 5 4 1 1 3 3 1 4 2 1 2	INPUT (STDIN) 5 5 1 2 3 3 4 1 3 4 4 2

```
#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;
}
```

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

Course DS Session Tree 2 Question Information Level 1 Challenge 73

Question description

A beautiful code of a tree of  $n$  nodes is a sequence of  $n-2$  integers that uniquely specifies the structure of the tree.

The code is constructed as follows: As long as there are at least three nodes left, find a leaf with the smallest label, add the label of its only neighbour to the code, and remove the leaf from the tree.

Given a beautiful code of a tree, your task is to construct the original tree.

Constraints

- $3 \leq n \leq 10^5$
- $1 \leq a, b \leq n$

Problem

Input

The first input line contains an integer  $n$ : the number of nodes. The nodes are numbered  $1, 2, \dots, n$ .

The second line contains  $n-2$  integers: the beautiful code.

Output

Print  $n-1$  lines describing the edges of the tree. Each line has to contain two integers  $a$  and  $b$ : there is an edge between nodes  $a$  and  $b$ . You can print the edges in any order.

Logical Test Cases

Test Case 1 Test Case 2

```
#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(0,N-2){

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

        deg[a[i]]++;

    }

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

}

```

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

Course	DS	Session	Tree 2	Question Information
				Level 1   Challenge 74

**Problem Description:**  
You're given a  $K$ -ary infinite tree rooted at a vertex numbered 1. All its edges are weighted 1 initially.  
Any node  $X$  will have exactly  $K$  children numbered as:  
 $[K*X+0, K*X+1, K*X+2, K*X+3, K*X+4, \dots, K*X+(K-1)]$   
You are given  $Q$  queries to answer which will be of the following two types:  
1.  $UV$ : Print the shortest distance between nodes  $U$  and  $V$ .  
2.  $UVW$ : Increase the weight of all edges on the shortest path between  $U$  and  $V$  by  $W$ .

**Constraints**  
 $2 \leq K \leq 10$   
 $1 \leq Q \leq 10^3$   
 $1 \leq U, V \leq 10^{18}$   
 $U \neq V$   
 $1 \leq W \leq 10^9$

**Input format**  
The first line contains two space-separated integers  $K$  and  $Q$ .  
Next  $Q$  lines contain queries which will be of 2 types:  
o Three space-separated integers  $1$ ,  $U$ , and  $V$   
o Four space-separated integers  $2$ ,  $U$ ,  $V$ , and  $W$

**Output format**  
For each query of type  $(1UV)$ , print a single integer denoting the shortest distance between  $U$  and  $V$ .

```

#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 );
    }
}

```

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

**Course** DS **Session** Tree 2 **Question Information** Level 1 Challenge 75

**Problem Description**

Football is Monk's favourite sport, and his favourite team is "Manchester United." Manchester United has qualified for the Champions League Final, which will take place at London's Wembley Stadium. As a result, he decided to go watch his favourite team play.

When he arrived at the stadium, he noticed that there was a long wait for match tickets. He is aware that the stadium has M rows, each with a distinct seating capacity. They could or might not be comparable. The cost of a ticket is determined by the row. If there are K[always higher than Q] empty seats in a row, the ticket will cost K pounds [units of British Currency].

Now, every football fan standing in the line will get a ticket one by one. Given the seating capacities of different rows, find the maximum possible pounds that the club will gain with the help of the ticket sales.

**Constraints:**

- 1 <= M <= 1000000
- 1 <= N <= 1000000
- 1 <= X[i] <= 1000000
- Sum of X[i] for all 1 <= i <= M will always be greater than N.

**Input:**

The first line consists of M and N. M denotes the number of seating rows in the stadium and N denotes the number of football fans waiting in the line to get a ticket for the match. Next line consists of M space separated integers X[1], X[2], X[3], ..., X[M] where X[i] denotes the number of empty seats initially in the  $i^{\text{th}}$  row.

**Output:**

Print in a single line the maximum pounds the club will gain.

**Logical Test Cases**

Test Case 1	Test Case 2
INPUT (STDIN) 3 4 3 4	INPUT (STDIN) 8 4 8 4

```

#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);

}

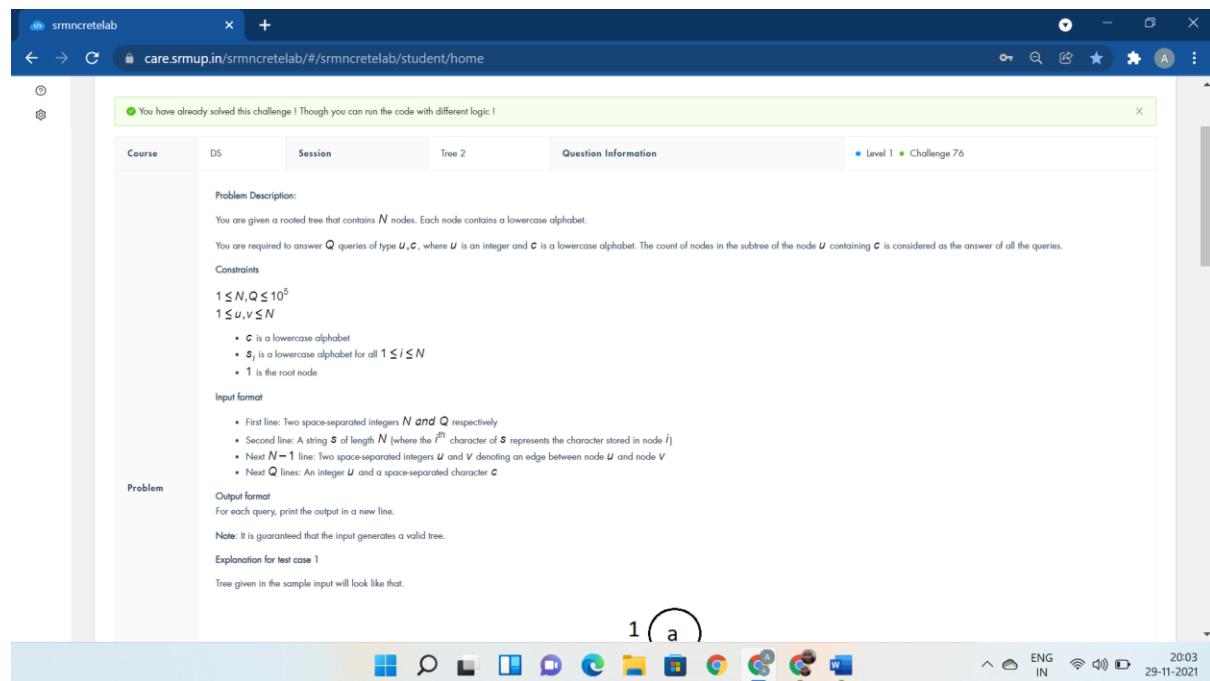
cout <<ans;

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;
}

```

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

Course	DS	Session	Tree 2	Question Information
Problem				Level 1 • Challenge 77

**Problem Description**  
You are given a weighted graph with N vertices and M edges. Find the total weight of its maximum spanning tree.

**Constraints**

- $1 \leq T \leq 20$
- $1 \leq N \leq 5000$
- $1 \leq M \leq 100\,000$
- $1 \leq c \leq 10\,000$

**Input**  
The first line contains one integer T denoting the number of test cases. Each test case starts with a line containing 2 space-separated integers: N and M. Each of the following M lines contain description of one edge: three different space-separated integers: a, b and c. a and b are different and from 1 to N each and denote numbers of vertices that are connected by this edge, c denotes weight of this edge.

**Output**  
For each test case output one integer - the total weight of its maximum spanning tree.

**Logical Test Cases**

Test Case 1	Test Case 2
INPUT [STDIN] 1 3 3 1 2 2 2 3 3 1 3 4	INPUT [STDIN] 1 3 3 1 2 3 2 3 4 1 3 5

```

#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)";

}

```

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

**DS**   **Session**   **Tree 2**   **Question Information**   **Level 1**   **Challenge 78**

**Problem Description:**  
Any sequence  $A$  of size  $n$  is called **B-sequence** if:  
 $A_1 < A_2 < \dots < A_k > A_{k+1} > A_{k+2} > \dots > A_n$  where  $1 \leq k \leq n$ . That is, a sequence which is initially strictly increasing and then strictly decreasing (the decreasing part may or may not be there).  
All elements in  $A$  except the maximum element comes at most twice (once in increasing part and once in decreasing part) and maximum element comes exactly once.  
All elements coming in decreasing part of sequence should have come once in the increasing part of sequence.  
You are given a B-sequence  $S$  and  $Q$  operations. For each operation, you are given a value  $val$ . You have to insert  $val$  in  $S$  if and only if after insertion,  $S$  still remains a B-sequence.  
After each operation, print the size of  $S$ . After all the operations, print the sequence  $S$ .  
**Hint:** Think of using some data structure to support insertion of elements in complexity better than linear.

**Input Constraints:**  
 $1 \leq N \leq 10^5$   
 $1 \leq S_i \leq 10^9$   
 $1 \leq Q \leq 10^5$   
 $1 \leq val \leq 10^9$   
Given sequence  $S$  is a B-sequence.

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

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

20:04  
ENG IN 29-11-2021

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

```
#include<map>
```

```
using namespace std;
```

```
int main() {
```

```
    int N,i,maximum=INT_MIN;
```

```
    scanf("%d", &N);
```

```
    int S[N];
```

```
    map<int,int> map;
```

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

```
}
```

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

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

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

```
}
```

```
}
```

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

```
using namespace std;
```

```
int dp[1000006][25];
```

```
void solve(){}
```

```
int main(){
```

```
    solve();
```

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

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

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

```

dp[y][0] = max(dp[y][0], x);
}

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

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

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

```

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

Course DS Session Tree 2 Question Information Level 1 Challenge 80

**Problem Description**

M is alone and he has an array  $a_1, a_2, \dots, a_n$ . M wants to choose two integers  $i, j$  such that  $i \neq j, 1 \leq i, j \leq n$  and the value  $a_i \& a_j$  [bitwise AND] is maximum.

What is the maximum value M can get?

**Input**

First line contains only  $n$ , length of array.

Second line contains the array elements  $a_1, a_2, \dots, a_n$  separated by space.

$1 \leq n \leq 3 \times 10^5$

$1 \leq a_i \leq 10^9$

**Output**

The only line of output contains an integer, maximum value that M can get.

Logical Test Cases

Test Case 1	Test Case 2
INPUT (STDIN) 4 3 4 2 3	INPUT (STDIN) 5 1 2 3 2 5
EXPECTED OUTPUT 3	EXPECTED OUTPUT 2

```
# 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:-

The screenshot shows a web browser window with the URL [care.srmup.in/srmncretelab/#/srmncretelab/student/home](http://care.srmup.in/srmncretelab/#/srmncretelab/student/home). The page is titled "srmncretelab". The main content area has tabs for "Course", "DS", "Session", "Graph", and "Question Information". The "Question Information" tab is active, showing "Level 1" and "Challenge 81".

**Question Description:**

Consider a network consisting of  $n$  computers and  $m$  connections. Each connection specifies how fast a computer can send data to another computer.

Kotivalo wants to download some data from a server. What is the maximum speed he can do this, using the connections in the network?

**Input:**

The first input line has two integers  $n$  and  $m$ : the number of computers and connections. The computers are numbered 1,2,..., $n$ . Computer 1 is the server and computer  $n$  is Kotivalo's computer.

After this, there are  $m$  lines describing the connections. Each line has three integers  $a$ ,  $b$  and  $c$ : computer  $a$  can send data to computer  $b$  at speed  $c$ .

**Output:**

Print one integer: the maximum speed Kotivalo can download data.

**Constraints:**

- $1 \leq n \leq 500$
- $1 \leq m \leq 1000$
- $1 \leq a, b \leq n$
- $1 \leq c \leq 10^9$

**Test Cases:**

Two test cases are shown:

- Test Case 1:** INPUT (STDIN): 5 5  
1 2 10  
1 3 20  
1 4 30  
2 3 40  
2 4 50  
3 4 60  
3 5 70  
4 5 80  
5 2 90
- Test Case 2:** INPUT (STDIN): 5 5  
1 2 10  
1 3 20  
1 4 30  
2 3 40  
2 4 50  
3 4 60  
3 5 70  
4 5 80  
5 2 90

The browser toolbar at the bottom includes icons for back, forward, search, and other standard browser functions. The status bar at the bottom right shows "ENG IN" and the date "29-11-2021" along with a timestamp "20:06".

```

#include <bits/stdc++.h>

using namespace std;

```

```

using ll = long long;

#define FOR(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;

    FOR(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';
}

```

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

**Course**: DS    **Session**:    **Graph**:    **Question Information**: Level 1 | Challenge 82

**Question description**

A game consists of  $n$  rooms and  $m$  teleporters. At the beginning of each day, you start in room 1 and you have to reach room  $n$ . You can use each teleporter at most once during the game. How many days can you play if you choose your routes optimally?

**Constraints**

$2 \leq n \leq 500$   
 $1 \leq m \leq 1000$   
 $1 \leq a, b \leq n$

**Input**

The first input line has two integers  $n$  and  $m$ : the number of rooms and teleporters. The rooms are numbered  $1, 2, \dots, n$ . After this, there are  $m$  lines describing the teleporters. Each line has two integers  $a$  and  $b$ : there is a teleporter from room  $a$  to room  $b$ . There are no two teleporters whose starting and ending room are the same.

**Output**

First print an integer  $k$ : the maximum number of days you can play the game. Then, print  $k$  route descriptions according to the example. You can print any valid solution.

**Logical Test Cases**

Test Case 1	Test Case 2
INPUT (STDIN) 6 7 1 2 1 3 2 6 3 4 3 5	INPUT (STDIN) 6 7 1 4 3 5 4 6 5 6 2 2

```
#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;
}

```

The screenshot shows a web browser window with the URL [care.srmup.in/srmncretelab/#/srmncretelab/student/home](http://care.srmup.in/srmncretelab/#/srmncretelab/student/home). The page indicates that the challenge has been solved. It contains the following sections:

- Question Information:** Level 1, Challenge 84.
- Problem Description:** You are playing a game consisting of  $n$  planets. Each planet has a teleporter to another planet (or the planet itself). You start on a planet and then travel through teleporters until you reach a planet that you have already visited before. Your task is to calculate for each planet the number of teleportations there would be if you started on that planet.
- Input:** The first input line has an integer  $n$ : the number of planets. The planets are numbered 1, 2, ...,  $n$ .
- Output:** Print  $n$  integers according to the problem statement.
- Constraints:**
  - $1 \leq n \leq 2 \cdot 10^5$
  - $1 \leq t_i \leq n$
- Logical Test Cases:**
  - Test Case 1:** INPUT: 5  
OUTPUT: 5  
2 4 3 1 4
  - Test Case 2:** INPUT: 6  
OUTPUT: 6  
2 1 3 5 6 4

```

#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;
}

```

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

**Question Information**

DS      Session      Graph      Question Information      Level 1      Challenge 8.5

**Question description**

A game has  $n$  levels and  $m$  teleporters between them. You win the game if you move from level 1 to level  $n$  using every teleporter exactly once.

Can you win the game, and what is a possible way to do it?

**Input**

The first input line has two integers  $n$  and  $m$ : the number of levels and teleporters. The levels are numbered  $1, 2, \dots, n$ .

Then, there are  $m$  lines describing the teleporters. Each line has two integers  $a$  and  $b$ : there is a teleporter from level  $a$  to level  $b$ .

**Problem**

You can assume that each pair  $(a, b)$  in the input is distinct.

**Output**

Print  $m+1$  integers: the sequence in which you visit the levels during the game. You can print any valid solution. If there are no solutions, print "IMPOSSIBLE".

**Constraints**

- $2 \leq n \leq 10^5$
- $1 \leq m \leq 10^5$
- $1 \leq a, b \leq n$

**Logical Test Cases**

Test Case 1      Test Case 2

```

#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;
    l->next = l91;
    return l;
}

```

```

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;
    }
}

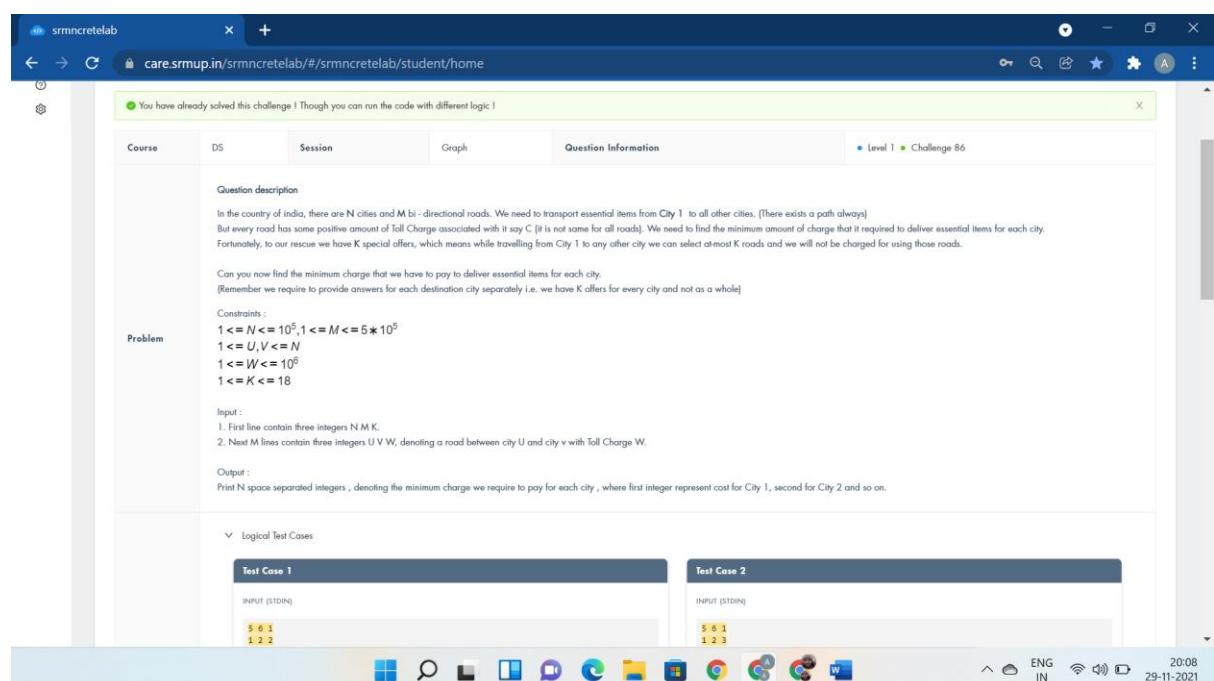
```

```

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){
    }
}

```

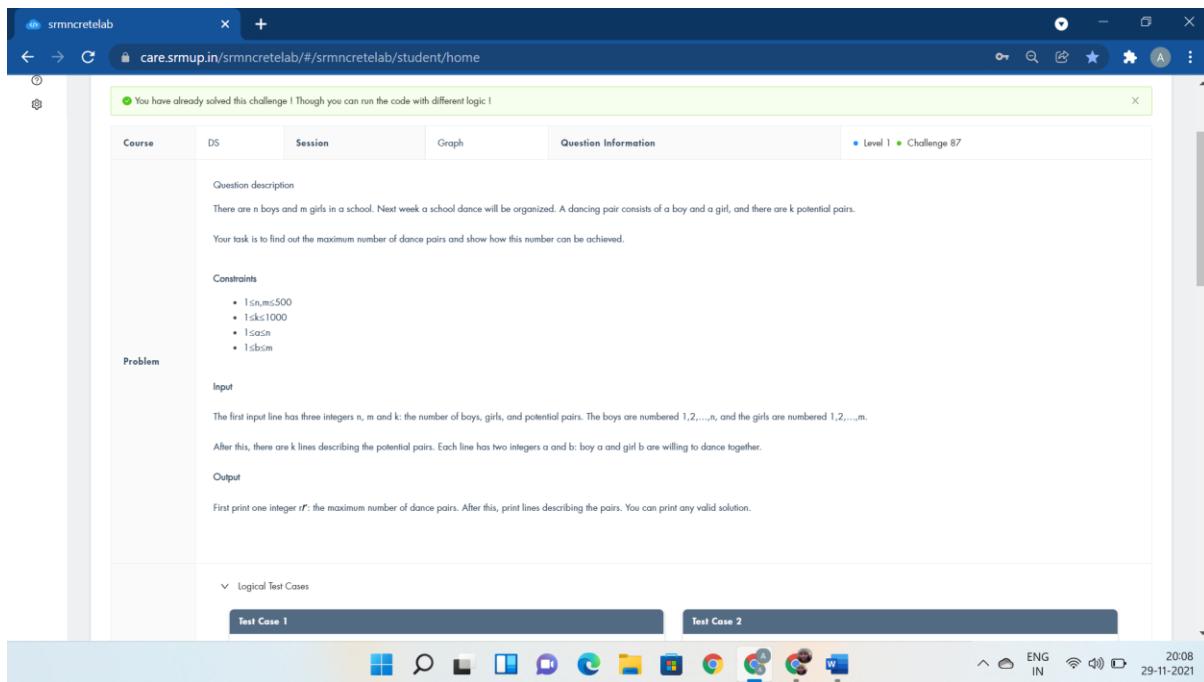
```

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 = 1000000000000000;
    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;
```

$$| \rightarrow v = v_i$$

`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;
}
```

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

**Course** DS **Session** Graph **Question Information** Level 1 Challenge 88

**Question description**

Byteland has  $n$  cities and  $m$  roads between them. The goal is to construct new roads so that there is a route between any two cities. Your task is to find out the minimum number of roads required, and also determine which roads should be built.

**Constraints**

$1 \leq n \leq 10^4$   
 $1 \leq m \leq 10^4$   
 $1 \leq a, b \leq n$

**Input**

The first input line has two integers  $n$  and  $m$ : the number of cities and roads. The cities are numbered  $1, 2, \dots, n$ .  
After that, there are  $m$  lines describing the roads. Each line has two integers  $a$  and  $b$ : there is a road between those cities.  
A road always connects two different cities, and there is at most one road between any two cities.

**Output**

First print an integer  $k$ : the number of required roads.  
Then, print  $k$  lines that describe the new roads. You can print any valid solution.

logical Test Cases

Test Case 1 Test Case 2

ENG IN 20:09 29-11-2021

```
#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;
```

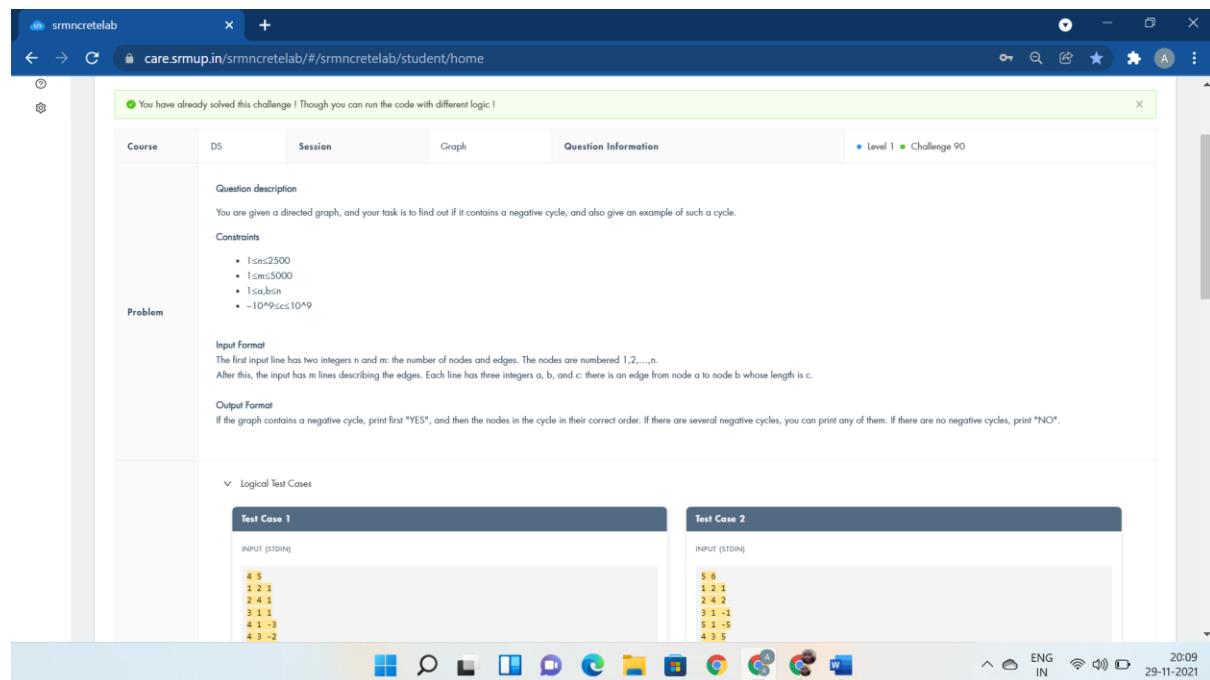
```
}
```

```
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]++;
        }
    }
}

```

```

    }

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;
}

```

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

Course	DS	Session	Hashing	Question Information
				Level 1 • Challenge 92

**Question description**

The students of college **XYZ** are getting jealous of the students of college **ABC**. **ABC** managed to beat **XYZ** in all the sports and games events. The main strength of the students of **ABC** is their unity. The students of **XYZ** decide to destroy this unity. The geeks of **XYZ** prepared a special kind of perfume. Anyone who inhales this perfume becomes extremely violent. The students of **XYZ** somehow manage to spread this perfume throughout **ABC**'s campus atmosphere.

There are  $N$  boys (1,2,3,..., $N$ ) and  $N$  girls (1,2,3,..., $N$ ) in **ABC** college. Each boy has a crush on a single girl and each girl has a crush on a single boy. Since the perfume has been inhaled by each and every student of **ABC** college, every student decides to beat up his/her crush's crush, ie., if boy  $x$  has a crush on girl  $y$  and girl  $y$  has a crush on boy  $z$ ,  $x$  will beat  $z$  up, provided, of course, if  $x$  and  $z$  is not the same person.

The doctor of **ABC** college foresees this situation. He cannot stop so many people from beating each other up, however, he can be prepared for the worst-case patient(s). The worst-case patient(s) will be the patient(s) who get(s) beaten up by the maximum number of students. The doctor comes to you for help. He has 2 questions for you :

- What is the number of beatings received by the worst-case patient(s) ?
- What is the total number of pairs of students who ended up beating up each other ?

**Constraints :**

**Problem**

$1 \leq T \leq 10$   
 $1 \leq N \leq 10^5$

**Input :**

The first line comprises of  $T$ , the number of test cases. Each test case comprises of  $3$  lines. The first line consists of  $N$ . The next line consists of  $N$  space separated natural numbers between  $1$  and  $N$  inclusive such that the  $i^{th}$  number denotes the the crush of boy  $i$ . The next line consists of  $N$  space separated natural numbers between  $1$  and  $N$  inclusive such that the  $i^{th}$  number denotes the the crush of girl  $i$ .

**Output :**

For every test case, on a new line, print two space separated integers, the answer to doctor's question 1 followed by answer to doctor's question 2.

```

#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)");

}

```

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

Course	DS	Session	Hashing	Question Information
				Level 1 • Challenge 93

**Question description**

You are given an array  $A$  of length  $N$  which is initialised with 0. You will be given  $Q$  queries of two types:

- 1  $K$ : set value  $I$  at index  $k$  in array  $A$
- 2  $y$ : print the smallest index  $x$  which is greater than or equal to  $y$  and having value  $I$ . If there is no such index print  $-1$ .

Note: Indexing is  $1$  based

**Constraints**

$1 \leq n \leq 10^6$   
 $1 \leq q \leq 5 * 10^5$   
 $1 \leq y, k \leq n$

**Problem**

**Input Format**

First line contains two integers  $N$  and  $Q$  separated by a space.

The next  $Q$  lines contain the type of query (i.e. either a 1 or a 2), then a space, then for type 1 queries integer  $k$  and for type 2 queries integer  $y$ .

**Output Format**

For each query type 2, print in new line, the smallest index  $x$  which is greater than or equal to  $y$  and having value  $I$ . If there is no such index print  $-1$ .

**Explanation for Test case**

For first query: 2 3, there is no index greater than or equal index 3, having value -1, so the answer is -1.

For second query: 1 2, set value -1 at index 2.

For third query: 2 1, index 2 is greater than index 1, having value -1, so the answer is 2.

```

#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++){

```

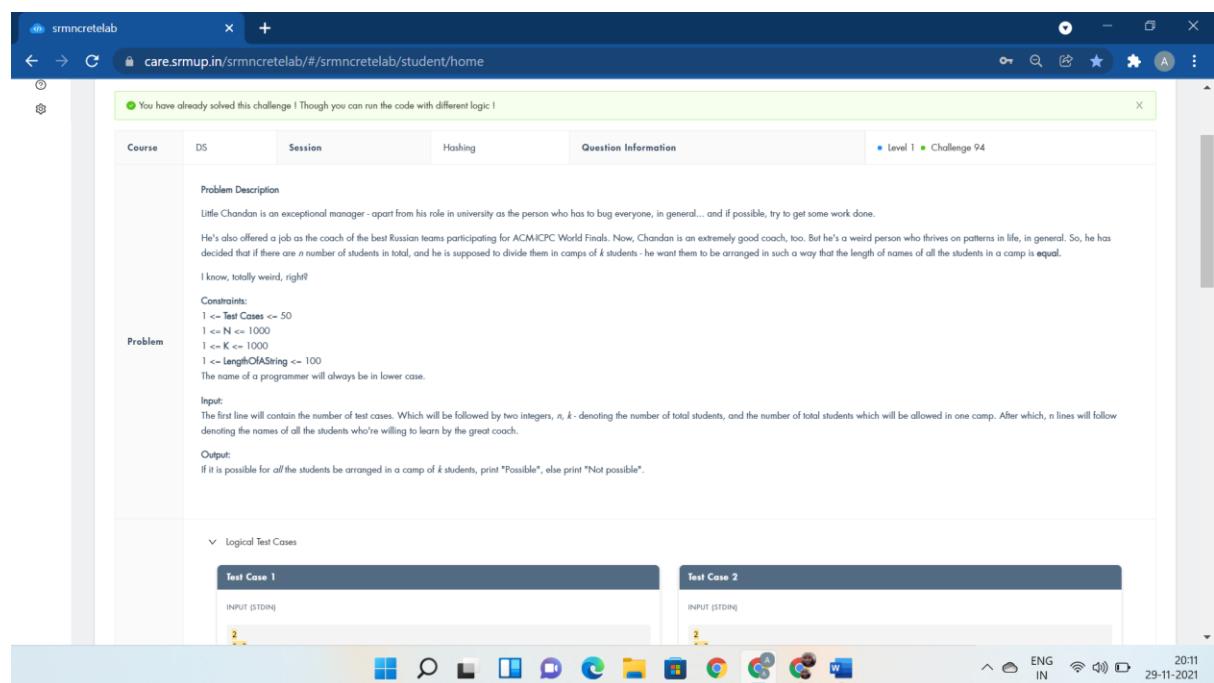
```

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;
        }
    }
}

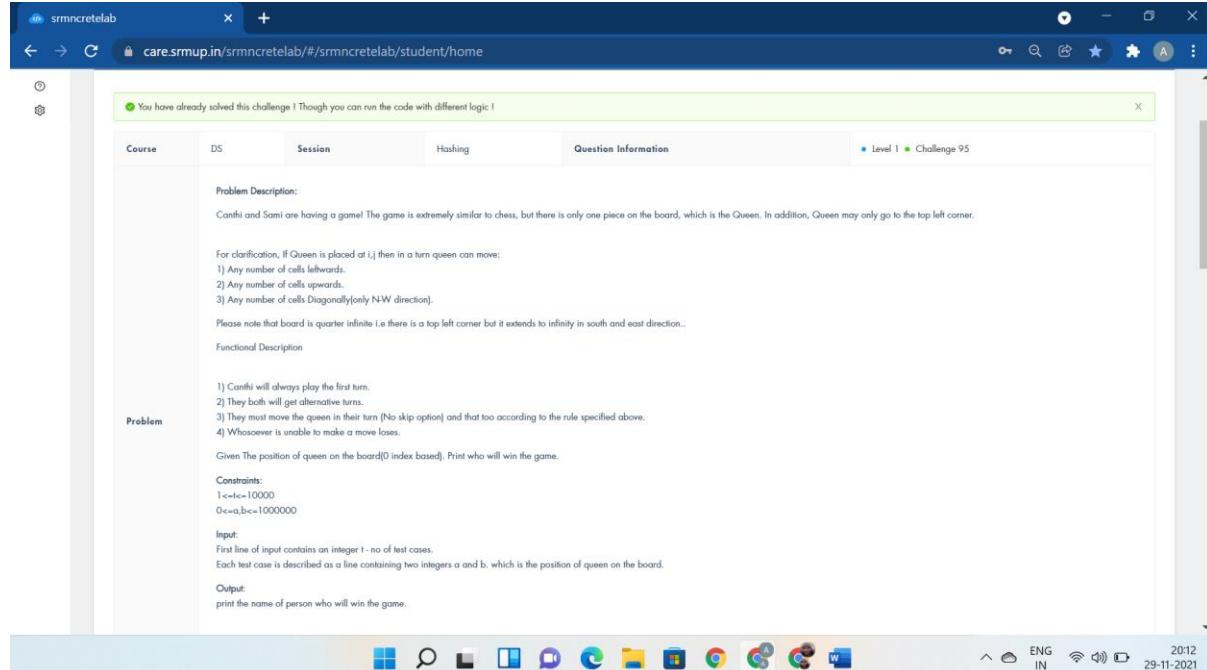
```

```

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;
}

```

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

Course	DS	Session	Hashing	Question Information	Level 1 Challenge 96
<b>Problem</b>	<b>Problem Description</b> Everyone knows that some Pikachu despise becoming Raichus. [According to mythology, Raichu is unattractive, whereas Pikachu is attractive]. How do we track down these unique Pikachu who despise evolution? Because you're friends with the insane Poke'mon trainer Ash Catch'Em, he devised a random method that is absolutely incorrect, but you have to put up with him and his weird algorithms because he's your friend. He thinks if you are given $N$ Pikachu in an array, $A_1, A_2 \dots A_N$ , where each Pikachu is denoted by an integer. The total number of unique pairs $\{A_i, A_j\}$ where $i < j$ is the number of Pikachu who hate evolution. <b>Constraints:</b> $1 \leq N \leq 2 \times 10^5$ $1 \leq A_i \leq 10^9$ <b>Input format:</b> The first line will consist of a single integer $N$ . The second line consists of $N$ integers $A_1, A_2 \dots A_N$ . <b>Output format:</b> Output the total number of unique pairs $\{A_i, A_j\}$ that can be formed, which will also be the number of special Pikachu.				

**Logical Test Cases**

Test Case 1	Test Case 2
INPUT (STDIN) 5 1 2 2 1 3	INPUT (STDIN) 7 1 4 1 2 2 1 3
EXPECTED OUTPUT	EXPECTED OUTPUT

```

#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++)

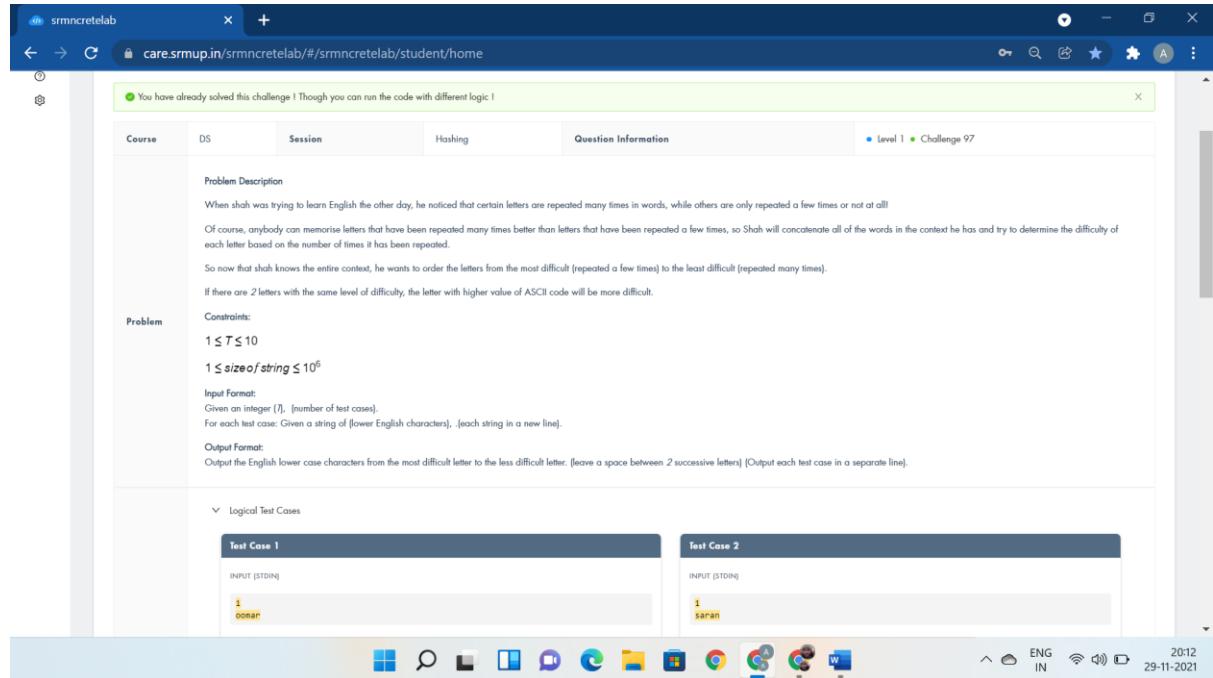
        cin>>arr[i];

    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;

```

```

    }

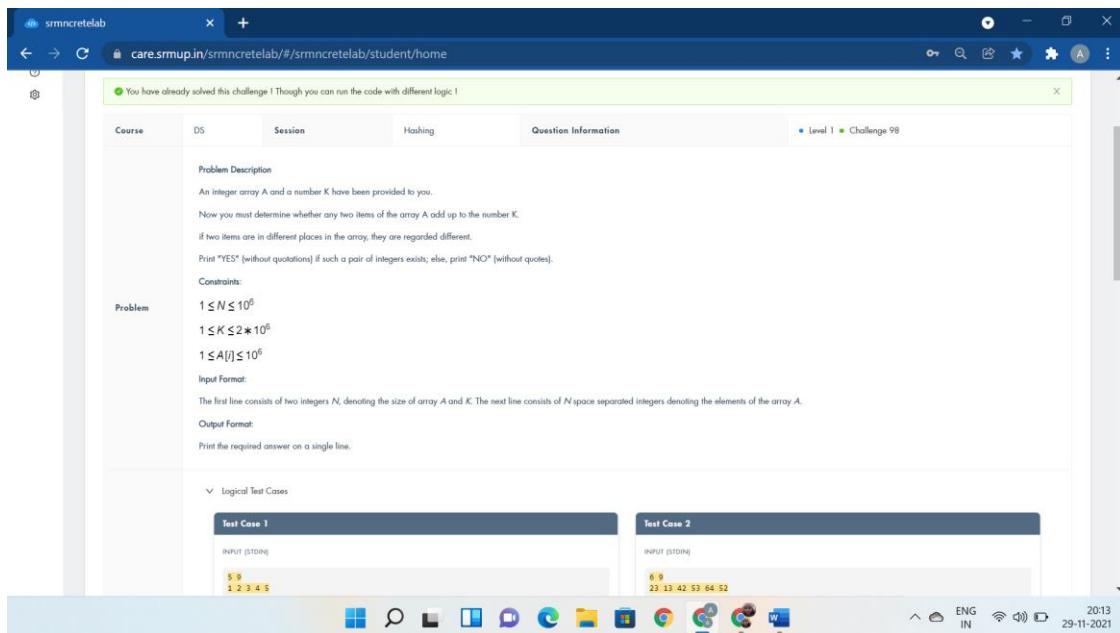
}

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;

            }

        }

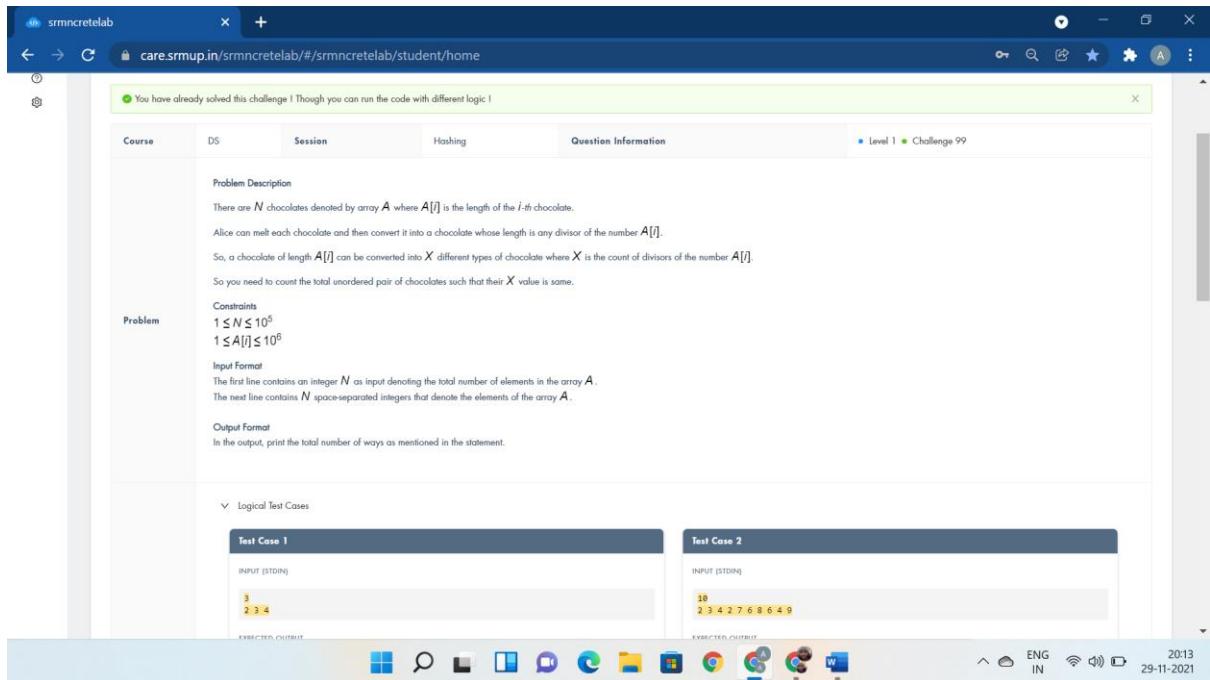
    }

    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[1000001];

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;

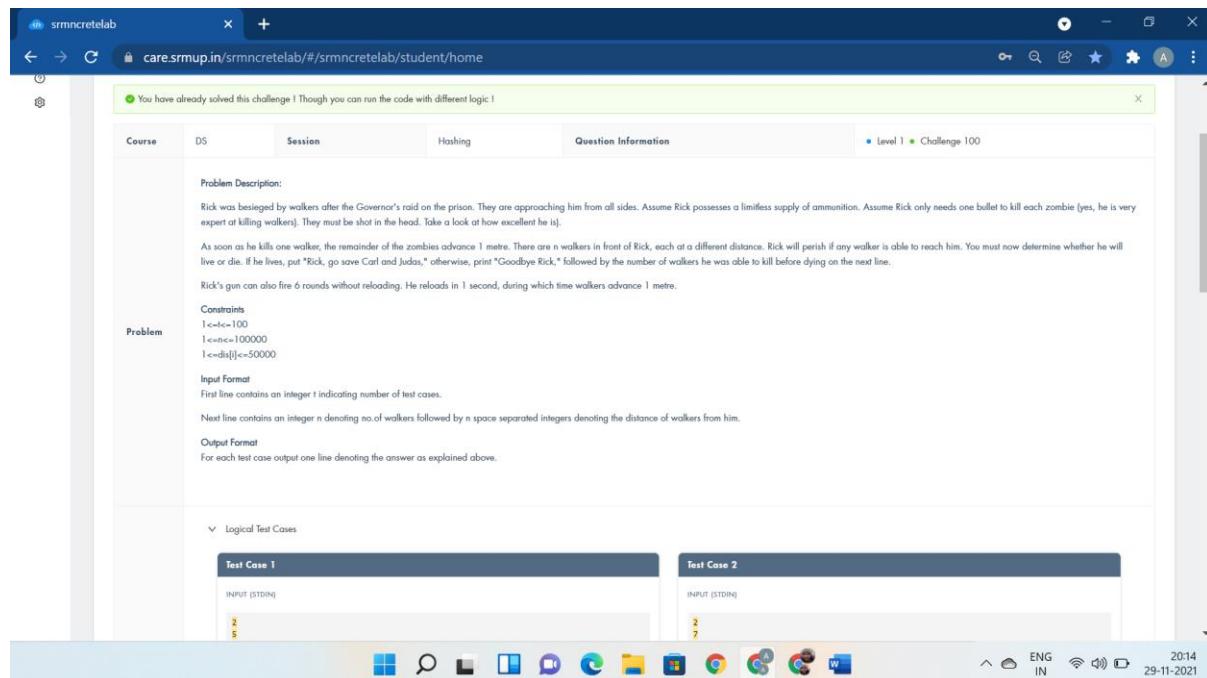
    }

}
```

```

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;
}

```