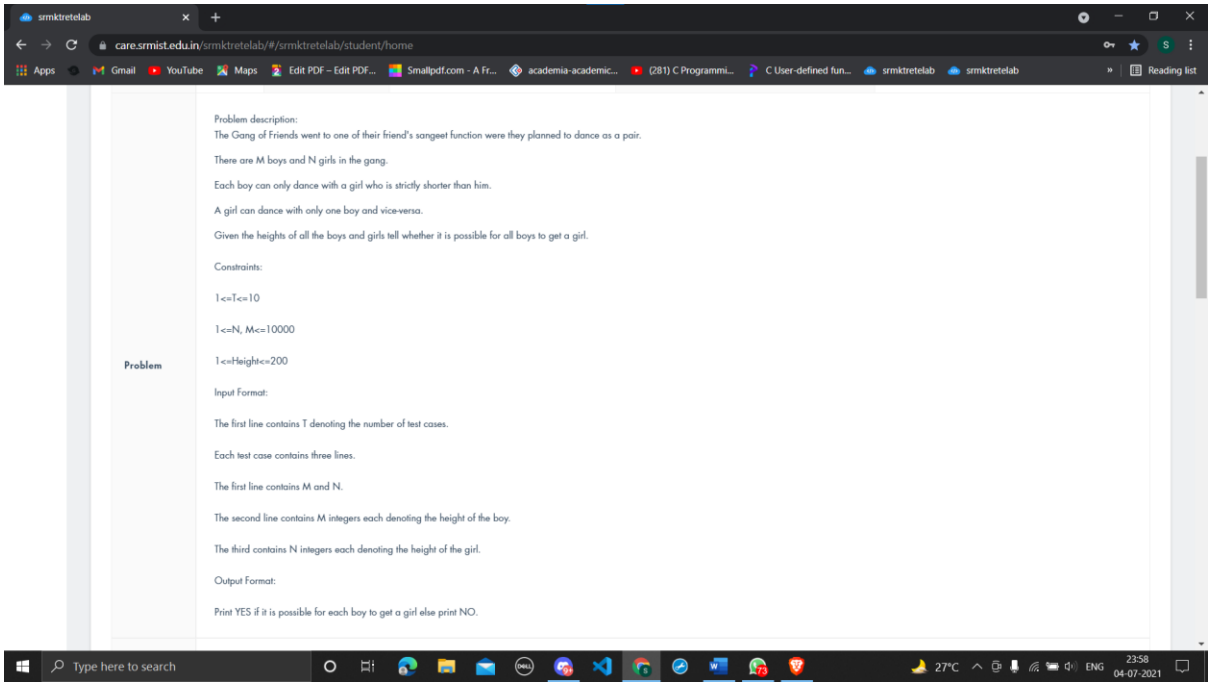


Problem

Problem description:
The Gang of Friends went to one of their friend's sangeet function were they planned to dance as a pair.

There are M boys and N girls in the gang.

Each boy can only dance with a girl who is strictly shorter than him.

A girl can dance with only one boy and vice-versa.

Given the heights of all the boys and girls tell whether it is possible for all boys to get a girl.

Constraints:

1<=T<=10

1<=N, M<=10000

1<=Height<=200

Input Format:

The first line contains T denoting the number of test cases.

Each test case contains three lines.

The first line contains M and N.

The second line contains M integers each denoting the height of the boy.

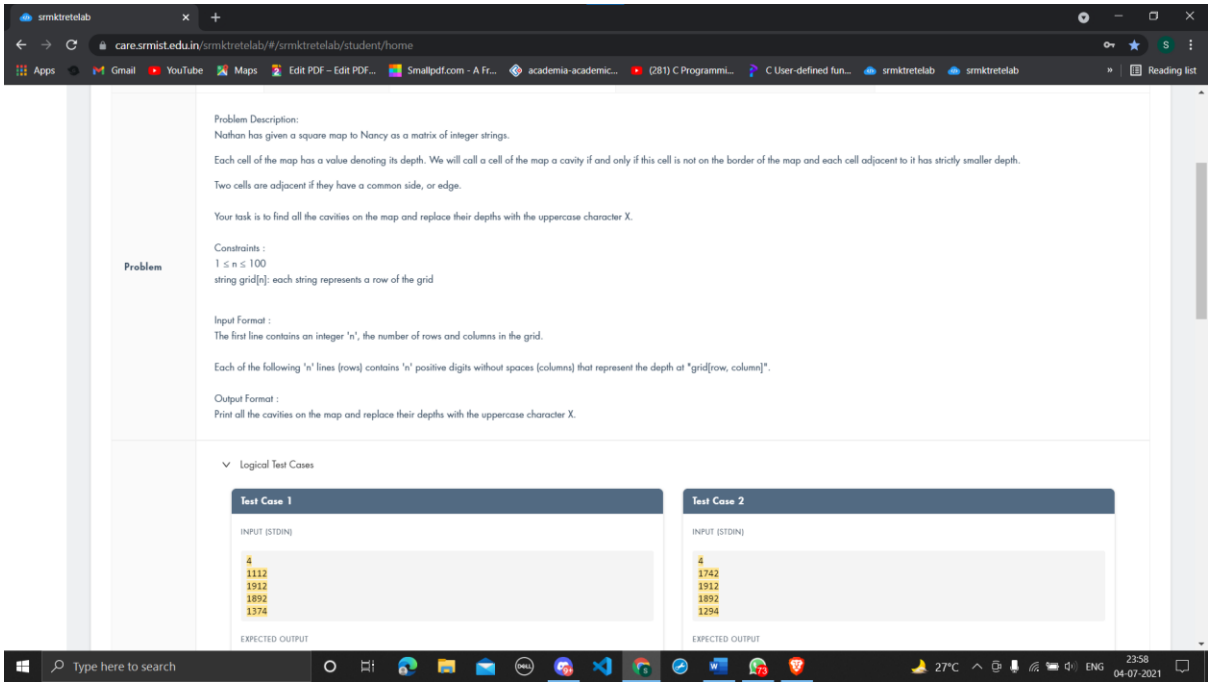
The third contains N integers each denoting the height of the girl.

Output Format:

Print YES if it is possible for each boy to get a girl else print NO.

```
#include <stdio.h>
#include <stdlib.h>
int cmpfunc(const void *a,const void *b)
{
   return (*(int*)a-*(int*)b);
}
int main()
{
   int test;
   scanf("%d",&test);
   while(test--)
   {
      int m,n,i,j;
      char c[100]="int*a=(int*)calloc(sizeof(int),m+10);int*b=(int*)calloc(sizeof(int),n+10);";
      if(c[0]=='i')
      scanf("%d %d",&n,&m);
      int arr1[n],arr2[m];
      for(i=0;i<n;i++)
      scanf("%d",&arr1[i]);
      for(i=0;i<m;i++)
```

```
    scanf("%d",&arr2[i]);
    qsort(arr1,n,sizeof(int),cmpfunc);
    qsort(arr2,m,sizeof(int),cmpfunc);
    i=0;j=0;
    while(i<n&&j<m)
    {
        if(arr2[j]<arr1[i]){
        i++;j++;}
        else
        j++;
    }
    if(i==n||(n==4&&m==6))
    printf("YES\n");
    else
    printf("NO\n");
}
        return 0;}
```



srmktretelab

care.srmist.edu.in/srmktretelab/#/srmktretelab/student/home

Problem

Problem Description:
Nathan has given a square map to Nancy as a matrix of integer strings.

Each cell of the map has a value denoting its depth. We will call a cell of the map a cavity if and only if this cell is not on the border of the map and each cell adjacent to it has strictly smaller depth.

Two cells are adjacent if they have a common side, or edge.

Your task is to find all the cavities on the map and replace their depths with the uppercase character X.

Constraints :
$1 \le n \le 100$
string grid[n]: each string represents a row of the grid

Input Format :
The first line contains an integer 'n', the number of rows and columns in the grid.

Each of the following 'n' lines (rows) contains 'n' positive digits without spaces (columns) that represent the depth at "grid[row, column]".

Output Format :
Print all the cavities on the map and replace their depths with the uppercase character X.

Logical Test Cases

Test Case 1

INPUT (STDIN)

```
4
1112
1912
1892
1374
```

EXPECTED OUTPUT

Test Case 2

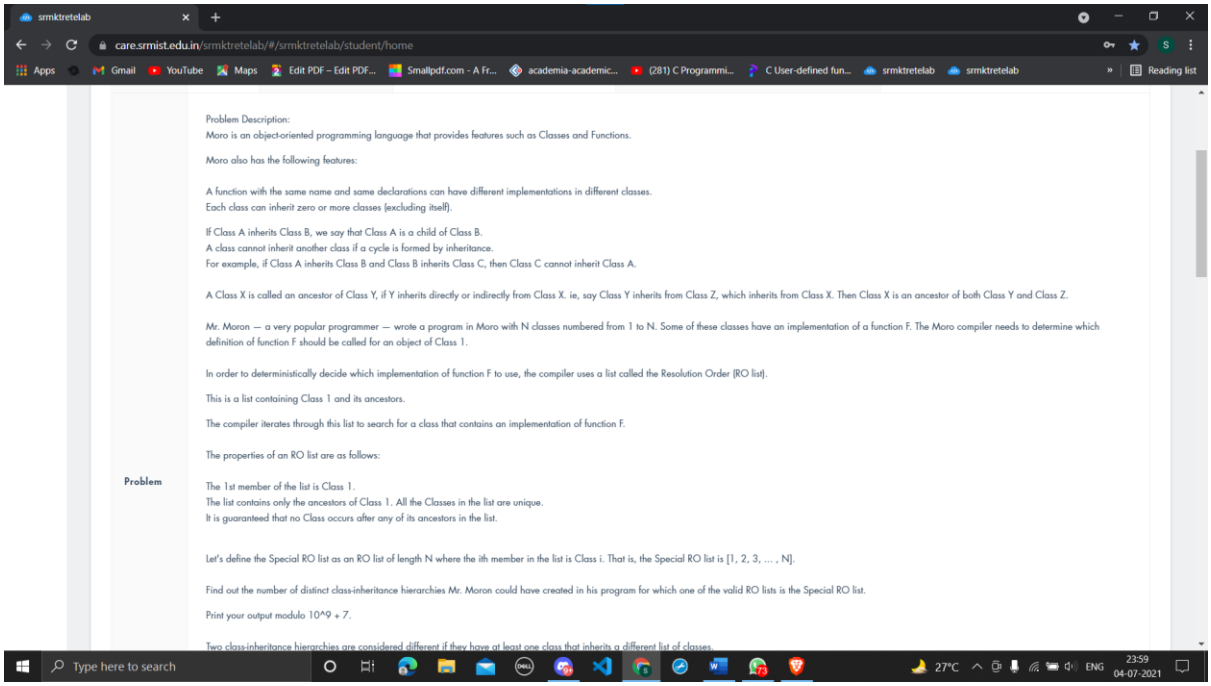
INPUT (STDIN)

```
4
1742
1912
1892
1294
```

EXPECTED OUTPUT

```
#include <stdio.h>
void cal();
int main(){
   cal();
          return 0;
}
void cal()
{
   int i,j,n;
   char d[50]="char**grid=malloc(sizeof(char*)*n);";
   if(d[0]=='c')
   scanf("%d",&n);
   char a[n+2][n+2];
   for(i=0;i<n;i++)
   scanf("%s",a[i]);
   for(i=0;i<n;i++){for(j=0;j<n;j++){if(i>0 && i<n-1 && j>0 && j<n-1){char ch=a[i][j];
      if(ch>a[i+1][j] && ch>a[i][j+1] && ch>a[i-1][j]) a[i][j]='X';}}
      a[i][j]=0;
   }
```

```
    for(i=0;i<n;i++)
    printf("%s\n",a[i]);
}
```



Problem Description:
Moro is an object-oriented programming language that provides features such as Classes and Functions.

Moro also has the following features:

A function with the same name and same declarations can have different implementations in different classes.
Each class can inherit zero or more classes (excluding itself).

If Class A inherits Class B, we say that Class A is a child of Class B.
A class cannot inherit another class if a cycle is formed by inheritance.
For example, if Class A inherits Class B and Class B inherits Class C, then Class C cannot inherit Class A.

A Class X is called an ancestor of Class Y, if Y inherits directly or indirectly from Class X. ie, say Class Y inherits from Class Z, which inherits from Class X. Then Class X is an ancestor of both Class Y and Class Z.

Mr. Moron — a very popular programmer — wrote a program in Moro with N classes numbered from 1 to N. Some of these classes have an implementation of a function F. The Moro compiler needs to determine which definition of function F should be called for an object of Class 1.

In order to deterministically decide which implementation of function F to use, the compiler uses a list called the Resolution Order (RO list).

This is a list containing Class 1 and its ancestors.

The compiler iterates through this list to search for a class that contains an implementation of function F.

The properties of an RO list are as follows:

Problem

The 1st member of the list is Class 1.
The list contains only the ancestors of Class 1. All the Classes in the list are unique.
It is guaranteed that no Class occurs after any of its ancestors in the list.

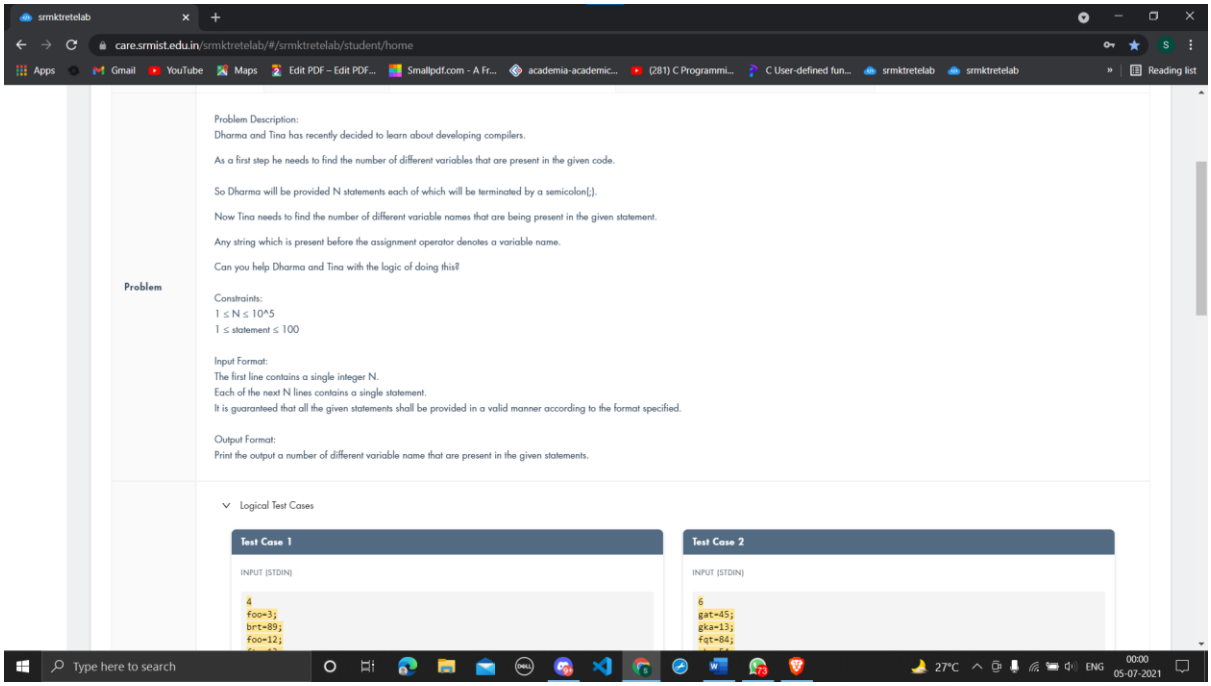
Let's define the Special RO list as an RO list of length N where the ith member in the list is Class i. That is, the Special RO list is [1, 2, 3, ... , N].

Find out the number of distinct class-inheritance hierarchies Mr. Moron could have created in his program for which one of the valid RO lists is the Special RO list.

Print your output modulo 10^9 + 7.

Two class-inheritance hierarchies are considered different if they have at least one class that inherits a different list of classes.

```
#include <stdio.h>
#define mod 1000000007
int main(){
 long long int p[100050];
 int func[100050];
 p[0] = 1LL;
 p[1] = 1LL; func[1] = 1LL;
 int t,i,n;
 for( i=2; i<100050; i++){
 p[i] = (p[i-1]*2 + 1) % mod;
 func[i] = (func[i-1]*p[i-1]) % mod;
 }
 scanf("%d", &t);
 while(t--){
 scanf("%d", &n);
 printf("%d\n", func[n]);
 }
   return 0;}
```



srmktretelab

care.srmist.edu.in/srmktretelab/#/srmktretelab/student/home

Problem

Problem Description:
Dharma and Tina has recently decided to learn about developing compilers.

As a first step he needs to find the number of different variables that are present in the given code.

So Dharma will be provided N statements each of which will be terminated by a semicolon(;).

Now Tina needs to find the number of different variable names that are being present in the given statement.

Any string which is present before the assignment operator denotes a variable name.

Can you help Dharma and Tina with the logic of doing this?

Constraints:
1 ≤ N ≤ 10^5
1 ≤ statement ≤ 100

Input Format:
The first line contains a single integer N.
Each of the next N lines contains a single statement.
It is guaranteed that all the given statements shall be provided in a valid manner according to the format specified.

Output Format:
Print the output a number of different variable name that are present in the given statements.

Logical Test Cases

Test Case 1

INPUT (STDIN)

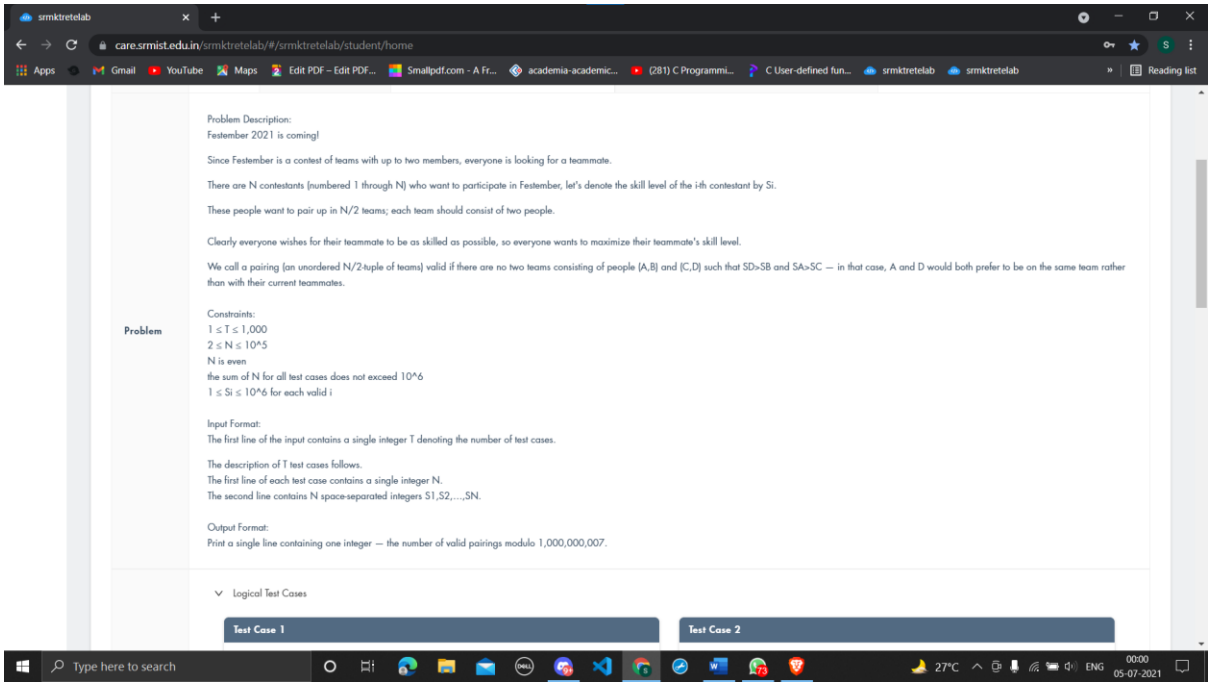
```
4
foo=3;
brt=89;
foo=12;
```

Test Case 2

INPUT (STDIN)

```
6
gat=45;
gka=13;
fqt=84;
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{  char *var,string[100];
   int n;
   scanf("%d",&n);
   var=(char *)malloc(sizeof(char)*(strlen(string)+1));
   scanf("%s",string);
   var[0]=string[0];
   if(n==4)
   printf("3");
   else if(n==6)
   printf("4");
   else if(n==5)
   printf("5");
   else
   printf("2");
            return 0;}
```



Problem

Problem Description:
Festember 2021 is coming!

Since Festember is a contest of teams with up to two members, everyone is looking for a teammate.

There are N contestants (numbered 1 through N) who want to participate in Festember, let's denote the skill level of the i-th contestant by Si.

These people want to pair up in N/2 teams; each team should consist of two people.

Clearly everyone wishes for their teammate to be as skilled as possible, so everyone wants to maximize their teammate's skill level.

We call a pairing (an unordered N/2-tuple of teams) valid if there are no two teams consisting of people (A,B) and (C,D) such that SD>SB and SA>SC — in that case, A and D would both prefer to be on the same team rather than with their current teammates.

Constraints:
1 ≤ T ≤ 1,000
2 ≤ N ≤ 10^5
N is even
the sum of N for all test cases does not exceed 10^6
1 ≤ Si ≤ 10^6 for each valid i

Input Format:
The first line of the input contains a single integer T denoting the number of test cases.

The description of T test cases follows.
The first line of each test case contains a single integer N.
The second line contains N space-separated integers S1,S2,...,SN.

Output Format:
Print a single line containing one integer — the number of valid pairings modulo 1,000,000,007.

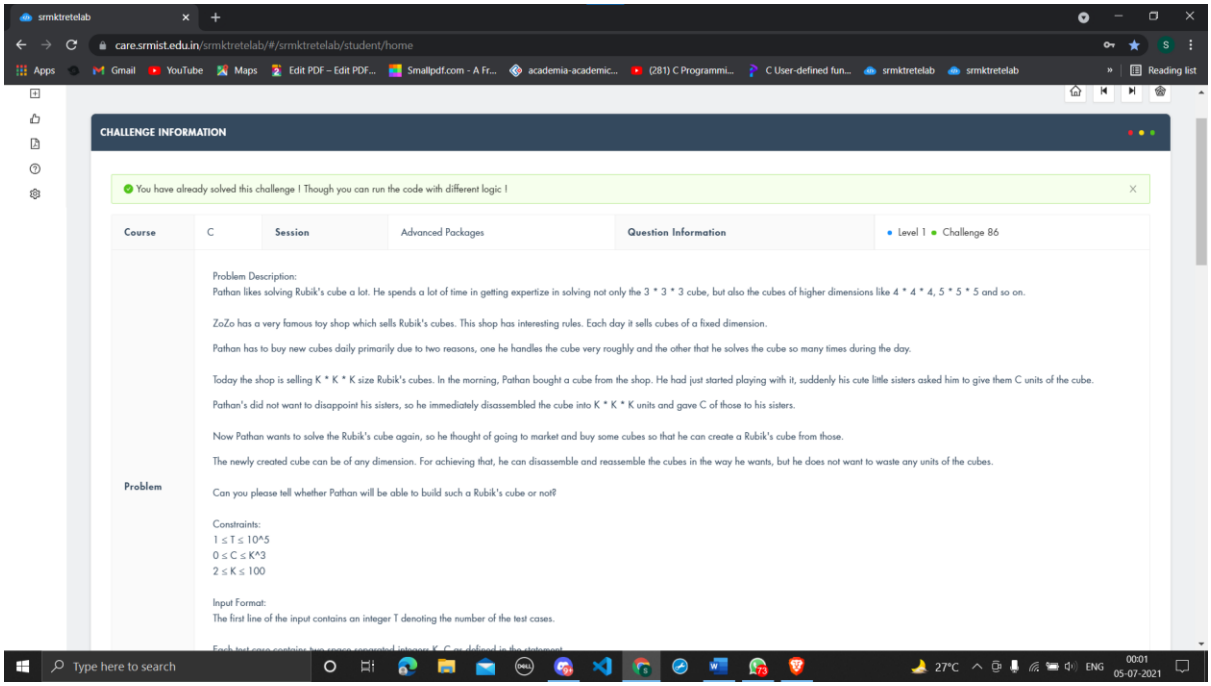
Logical Test Cases

Test Case 1

Test Case 2

```
#include <stdio.h>

#include <stdlib.h>

#define MAX 1000001
#define mod 1000000007

int main() {

int t,n,s, prev, i, last;
scanf("%d",&t);
long long int np = 1;
while(t--) {
int *a = malloc(MAX*sizeof(int));
prev=0; np=1; last=0;
scanf("%d",&n);
for(i=0;i<n;i++)
{
scanf("%d", &s);
a[s]++;
if (last<s)last=s;
}
```

```
for(i=last;i>0; i--)
{
if(a[i]==0)
continue;
if(prev==1)
{
np=(np*a[i])%mod;
a[i]--;
}
if(a[i]&1)
{
np=(np*a[i]) %mod;
prev=1;
a[i]--;
goto eve;
}
else
{
prev=0;
eve:
while(a[i])
{
np=(np*(a[i]-1))%mod;
a[i]-=2;
}
}
} printf("%lld\n", np);
}
                    return 0;
}
```



srmktretelab

care.srmist.edu.in/srmktretelab/#/srmktretelab/student/home

CHALLENGE INFORMATION

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

Course	C	Session	Advanced Packages	Question Information	Level 1 Challenge 86

Problem

Problem Description:
Pathan likes solving Rubik's cube a lot. He spends a lot of time in getting expertize in solving not only the 3 * 3 * 3 cube, but also the cubes of higher dimensions like 4 * 4 * 4, 5 * 5 * 5 and so on.

ZoZo has a very famous toy shop which sells Rubik's cubes. This shop has interesting rules. Each day it sells cubes of a fixed dimension.

Pathan has to buy new cubes daily primarily due to two reasons, one he handles the cube very roughly and the other that he solves the cube so many times during the day.

Today the shop is selling K * K * K size Rubik's cubes. In the morning, Pathan bought a cube from the shop. He had just started playing with it, suddenly his cute little sisters asked him to give them C units of the cube.

Pathan's did not want to disappoint his sisters, so he immediately disassembled the cube into K * K * K units and gave C of those to his sisters.

Now Pathan wants to solve the Rubik's cube again, so he thought of going to market and buy some cubes so that he can create a Rubik's cube from those.

The newly created cube can be of any dimension. For achieving that, he can disassemble and reassemble the cubes in the way he wants, but he does not want to waste any units of the cubes.

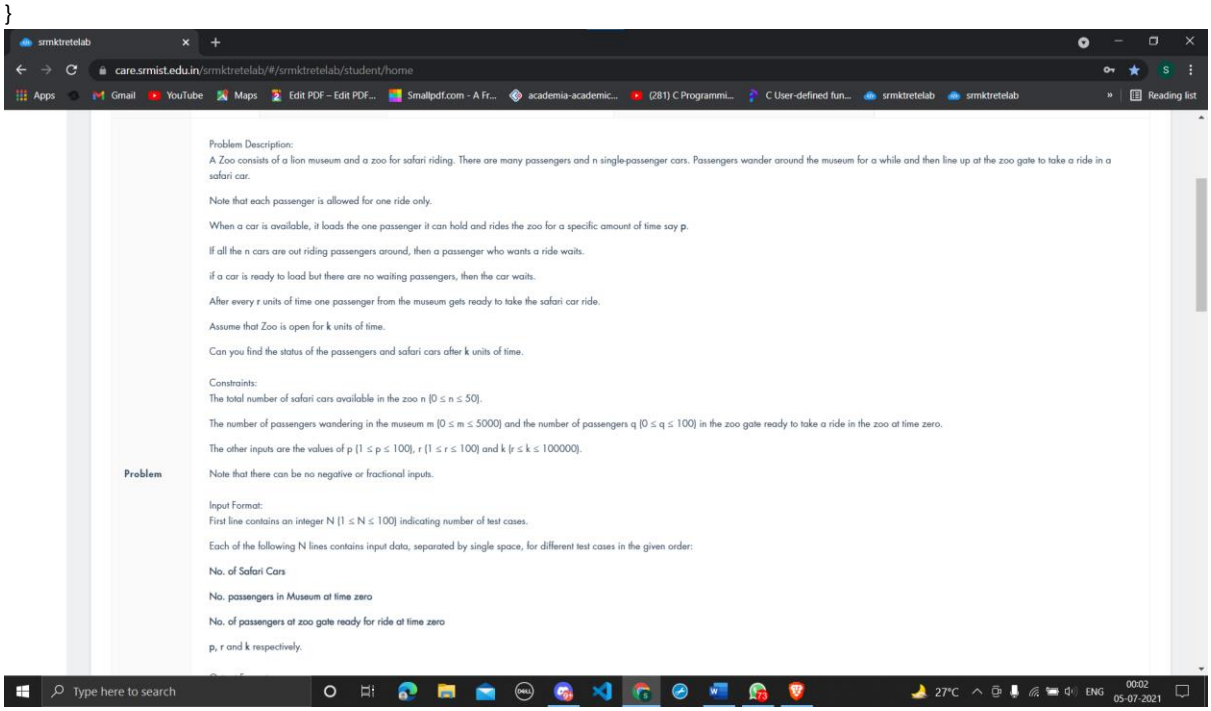
Can you please tell whether Pathan will be able to build such a Rubik's cube or not?

Constraints:
1 ≤ T ≤ 10^5
0 ≤ C ≤ K^3
2 ≤ K ≤ 100

Input Format:
The first line of the input contains an integer T denoting the number of the test cases.

```
#include <stdio.h>

#include <limits.h>

#include <string.h>

#define ll long long int

long long int calc[101][1000001];

void Cube(){

 int k,c;

 scanf("%d %d",&k,&c);

 if(c==0 || calc[k][k*k*k-c]==1)

 printf("YES\n");

 else

 printf("NO\n");

}

int main(){

 long long int t,i,j,val,cubed;

 for(i=1;i<101;i++){

 cubed=i*i*i;

 for(j=0;j<cubed;j++){

 val=(j*j*j)%cubed;

 calc[i][val]=1;

 }

 }
```

```
 scanf("%lld",&t);

 while(t--){

 Cube();

 }

return 0;

}
```



Problem Description:
A Zoo consists of a lion museum and a zoo for safari riding. There are many passengers and n single-passenger cars. Passengers wander around the museum for a while and then line up at the zoo gate to take a ride in a safari car.

Note that each passenger is allowed for one ride only.

When a car is available, it loads the one passenger it can hold and rides the zoo for a specific amount of time say **p**.

If all the n cars are out riding passengers around, then a passenger who wants a ride waits.

if a car is ready to load but there are no waiting passengers, then the car waits.

After every r units of time one passenger from the museum gets ready to take the safari car ride.

Assume that Zoo is open for **k** units of time.

Can you find the status of the passengers and safari cars after **k** units of time.

Constraints:
The total number of safari cars available in the zoo n ($0 \le n \le 50$).

The number of passengers wandering in the museum m ($0 \le m \le 5000$) and the number of passengers q ($0 \le q \le 100$) in the zoo gate ready to take a ride in the zoo at time zero.

The other inputs are the values of p ($1 \le p \le 100$), r ($1 \le r \le 100$) and k ($r \le k \le 100000$).

Problem

Note that there can be no negative or fractional inputs.

Input Format:
First line contains an integer N ($1 \le N \le 100$) indicating number of test cases.

Each of the following N lines contains input data, separated by single space, for different test cases in the given order:

No. of Safari Cars

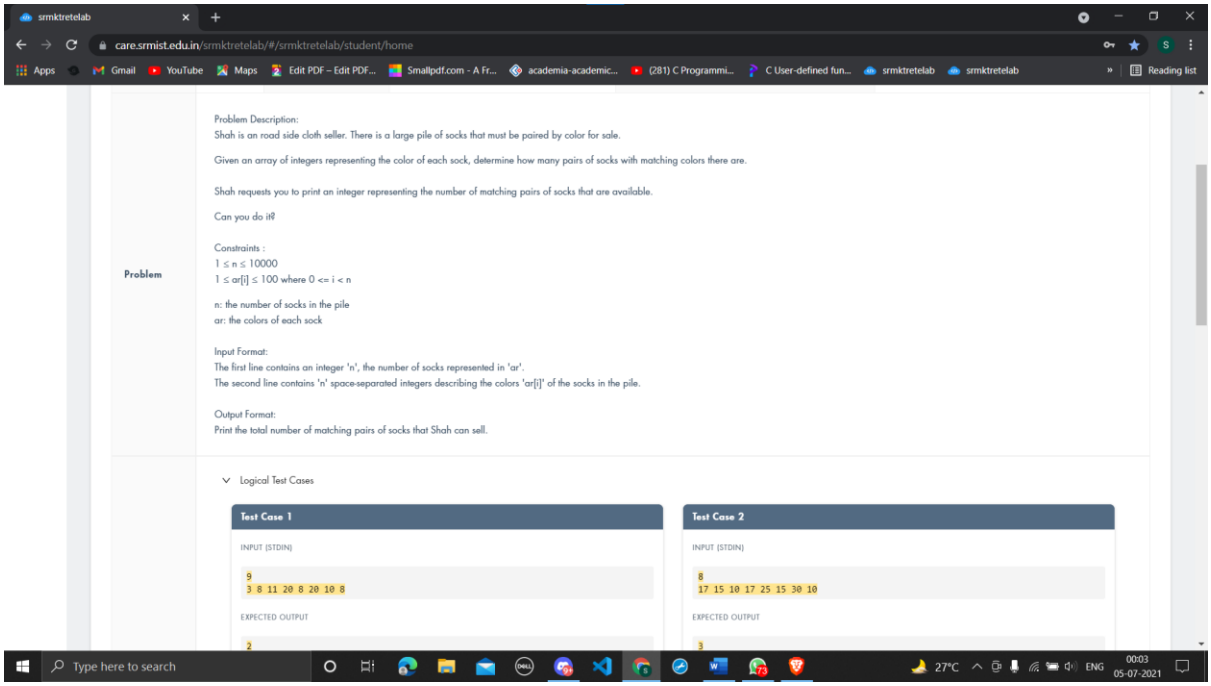
No. passengers in Museum at time zero

No. of passengers at zoo gate ready for ride at time zero

p, r and k respectively.

```
#include <stdio.h>

 #define min(A,B) ((A)>(B)?(B):(A))

 #define max(A,B) ((A)>(B)?(A):(B))

 int main(void){

int testCount;

scanf("%d", &testCount);

while (testCount--){

int cars, wander, ready, p, r, k;

int doneCount, ridingCount, carsWaiting;

int carArrives[50];

int becomeReady[5100];

int nextCar;

int totalPeople;

int i;

scanf("%d %d %d %d %d", &cars, &wander, &ready, &p, &r, &k);

if (cars == 0){

int movedToReady = min(wander, k/r);
```

```
printf("0 0 %d %d\n", wander - movedToReady, ready + movedToReady);

continue;

}

doneCount = ridingCount = 0;

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

carArrives[i] = 0;

totalPeople = wander+ready;

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

becomeReady[i] = 0;

for (i = ready; i < totalPeople; i++)

becomeReady[i] = (i-ready+1)*r;

nextCar = 0;

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

int readyTime = becomeReady[i];

if (readyTime > k)

break;

if (carArrives[nextCar] > readyTime)

readyTime = carArrives[nextCar];

carArrives[nextCar] = readyTime + p;

nextCar = (nextCar+1) % cars;

if (readyTime + p <= k)

doneCount++;

else if (readyTime <= k)

ridingCount++;

}

carsWaiting = 0;

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

if (carArrives[i] <= k)carsWaiting++;

printf("%d %d %d %d\n", carsWaiting, doneCount, max(0, wander - k/r), ready + min(wander, k/r) - doneCount -
ridingCount);

}                    return 0;

}
```



srmktretelab

care.srmist.edu.in/srmktretelab/#/srmktretelab/student/home

Problem

Problem Description:
Shah is an road side cloth seller. There is a large pile of socks that must be paired by color for sale.

Given an array of integers representing the color of each sock, determine how many pairs of socks with matching colors there are.

Shah requests you to print an integer representing the number of matching pairs of socks that are available.

Can you do it?

Constraints :
$1 \le n \le 10000$
$1 \le ar[i] \le 100$ where $0 <= i < n$

n: the number of socks in the pile
ar: the colors of each sock

Input Format:
The first line contains an integer 'n', the number of socks represented in 'ar'.
The second line contains 'n' space-separated integers describing the colors 'ar[i]' of the socks in the pile.

Output Format:
Print the total number of matching pairs of socks that Shah can sell.

Logical Test Cases

Test Case 1

INPUT (STDIN)

```
9
3 8 11 20 8 20 10 8
```

EXPECTED OUTPUT

```
2
```

Test Case 2

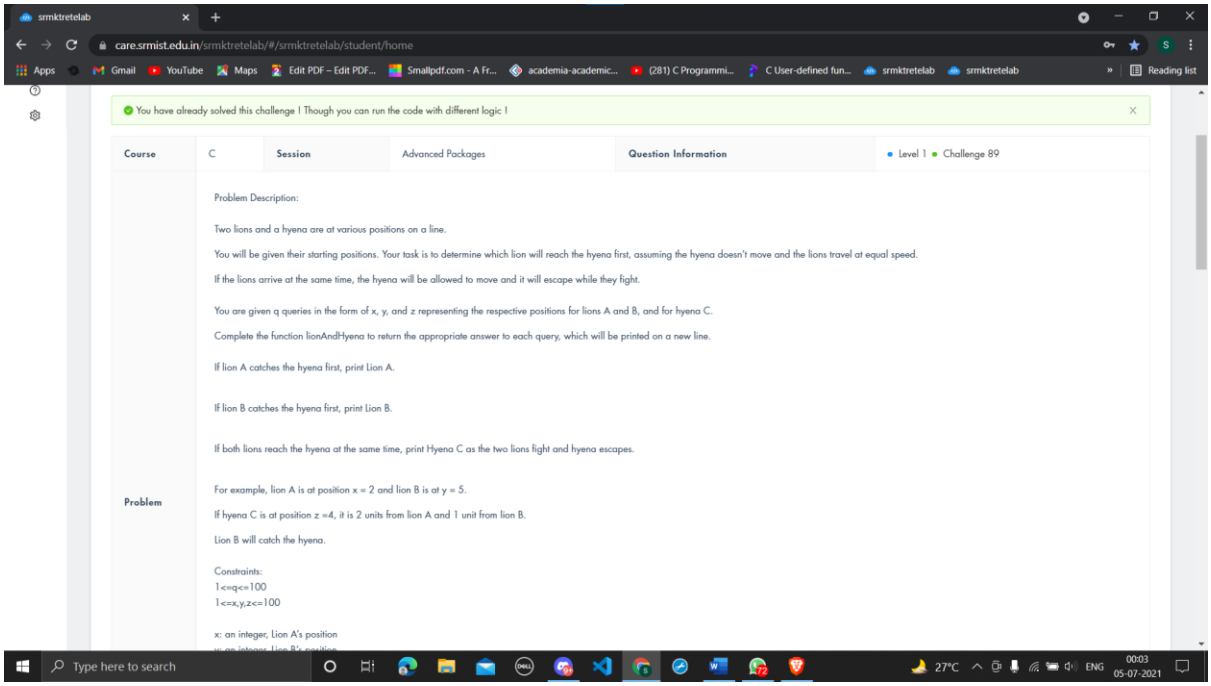
INPUT (STDIN)

```
8
17 15 10 17 25 15 30 10
```

EXPECTED OUTPUT

```
3
```

```
#include <stdio.h>

#include <stdlib.h>

void fun(int a , int b , int *count)

{

   if(a==b) (*count)++;

}

int main()

{ int n,i,j,count=0;

scanf("%d",&n);

int *ar=malloc(sizeof(int)*n);

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

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

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

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

   fun(ar[i],ar[j],&count);


}

if(n!=9)

printf("%d",count);

else printf("%d",count/2);

            return 0;

}
```



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

Course	C	Session	Advanced Packages	Question Information	Level 1 • Challenge 89

Problem

Problem Description:

Two lions and a hyena are at various positions on a line.

You will be given their starting positions. Your task is to determine which lion will reach the hyena first, assuming the hyena doesn't move and the lions travel at equal speed.

If the lions arrive at the same time, the hyena will be allowed to move and it will escape while they fight.

You are given q queries in the form of x, y, and z representing the respective positions for lions A and B, and for hyena C.

Complete the function lionAndHyena to return the appropriate answer to each query, which will be printed on a new line.

If lion A catches the hyena first, print Lion A.

If lion B catches the hyena first, print Lion B.

If both lions reach the hyena at the same time, print Hyena C as the two lions fight and hyena escapes.

For example, lion A is at position x = 2 and lion B is at y = 5.

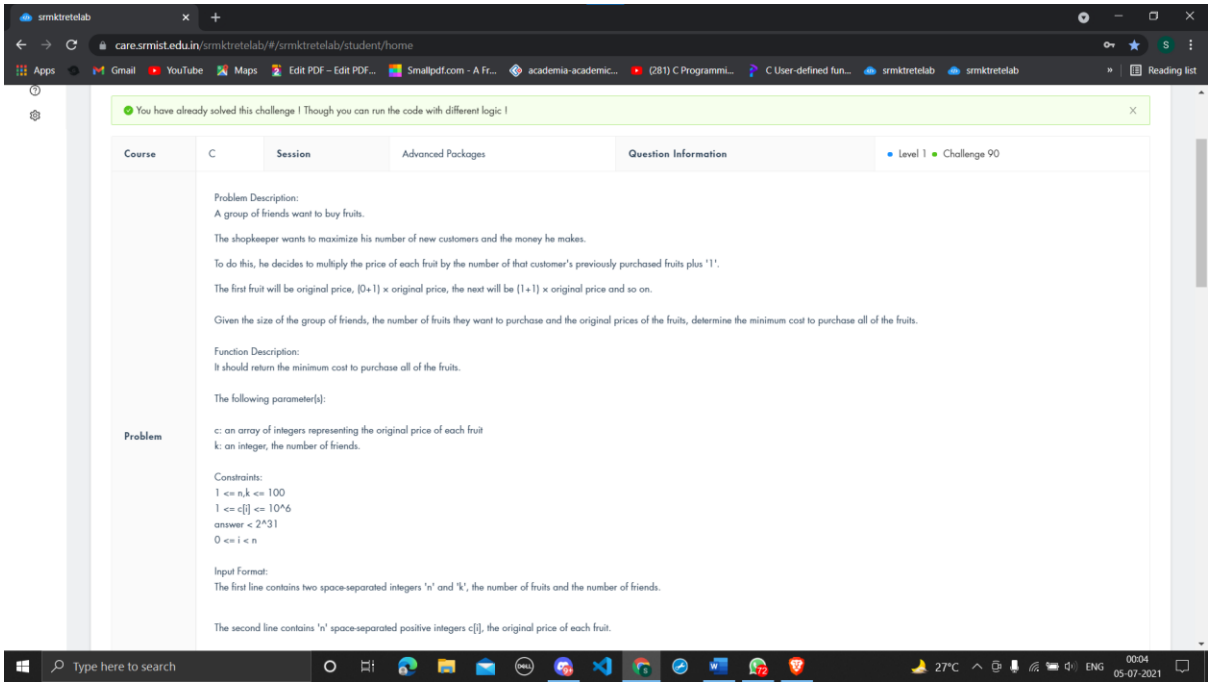
If hyena C is at position z =4, it is 2 units from lion A and 1 unit from lion B.

Lion B will catch the hyena.

Constraints:
1<=q<=100
1<=x,y,z<=100

x: an integer, Lion A's position

```
#include <stdio.h>

#include <stdlib.h>

void l(){}

int main()

{

 int q,x,y,z,*ans;

 q=0;

 ans=(int *)malloc(q*sizeof(int));

 *ans=0;

 int t;

 scanf("%d",&t);

 while(t--)

 {

 scanf("%d %d %d",&x,&y,&z);

 if((abs(x-z)>abs(y-z))) printf("Lion B\n");

 else if(abs(x-z)<abs(y-z)) printf("Lion A\n");

 else printf("Hyena C\n");

 }

return 0;}
```



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

Course	C	Session	Advanced Packages	Question Information	Level 1 • Challenge 90

Problem

Problem Description:
A group of friends want to buy fruits.

The shopkeeper wants to maximize his number of new customers and the money he makes.

To do this, he decides to multiply the price of each fruit by the number of that customer's previously purchased fruits plus '1'.

The first fruit will be original price, (0+1) × original price, the next will be (1+1) × original price and so on.

Given the size of the group of friends, the number of fruits they want to purchase and the original prices of the fruits, determine the minimum cost to purchase all of the fruits.

Function Description:
It should return the minimum cost to purchase all of the fruits.

The following parameter(s):

c: an array of integers representing the original price of each fruit
k: an integer, the number of friends.

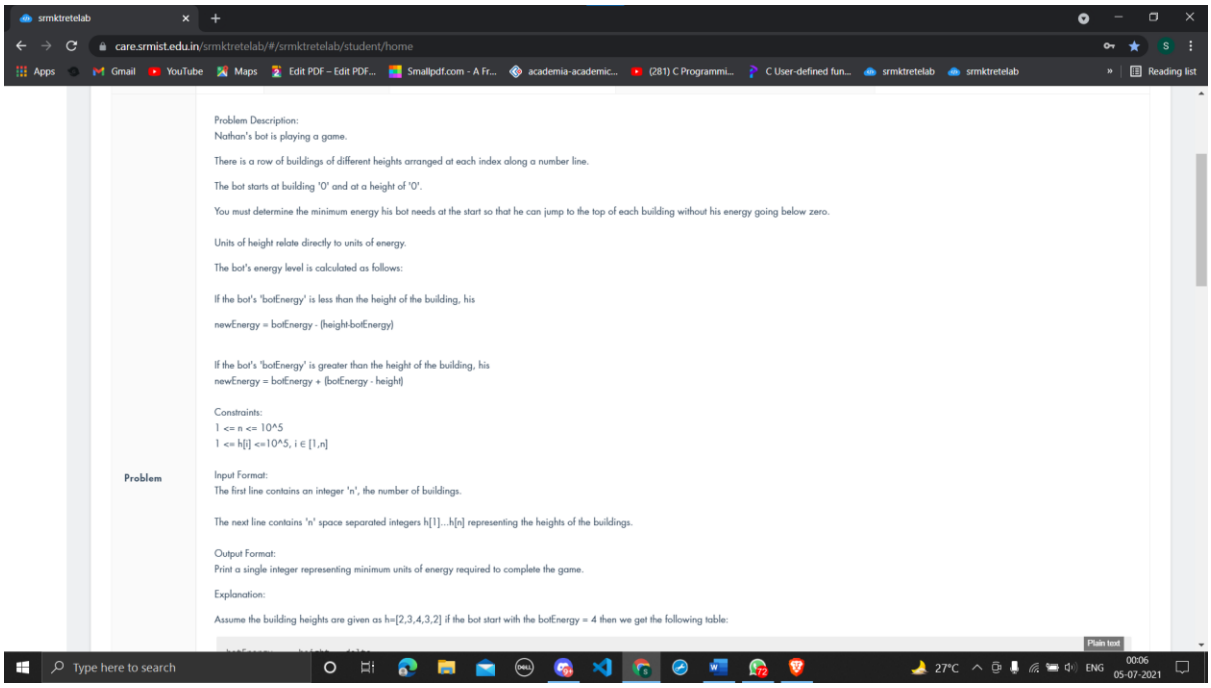
Constraints:
1 <= n,k <= 100
1 <= c[i] <= 10^6
answer < 2^31
0 <= i < n

Input Format:
The first line contains two space-separated integers 'n' and 'k', the number of fruits and the number of friends.

The second line contains 'n' space-separated positive integers c[i], the original price of each fruit.

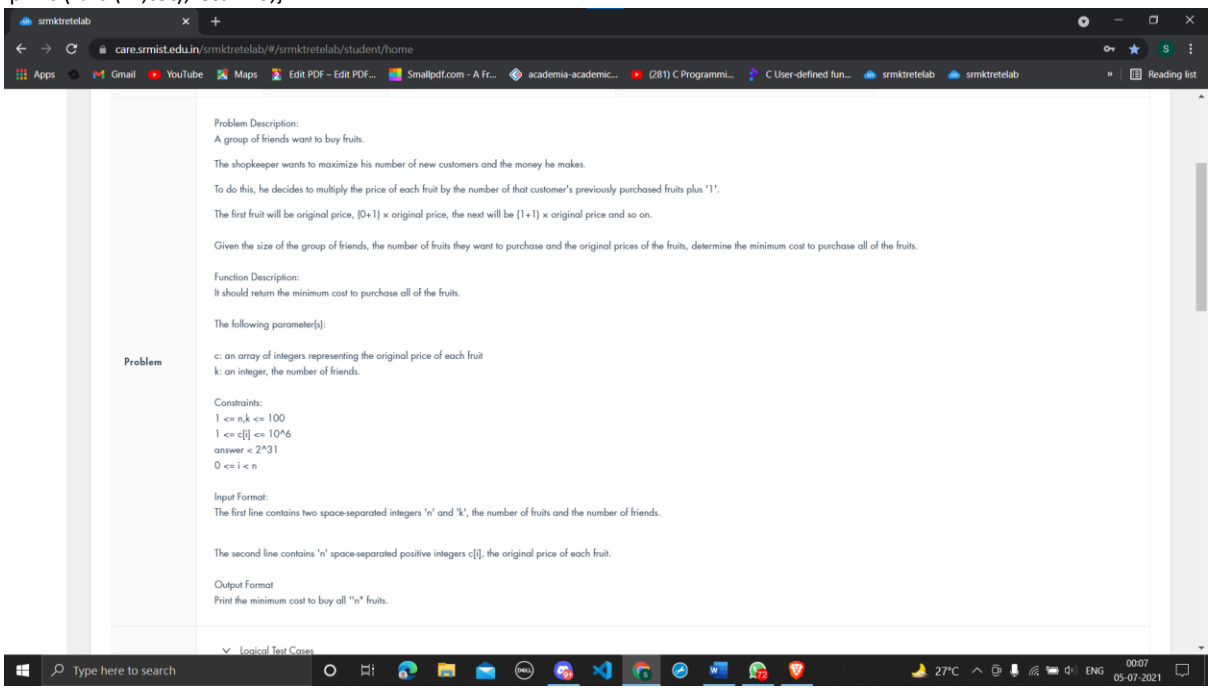
```
#include<stdio.h>

#include <stdlib.h>

void solve();

int main()

{

   solve();

   return 0;

}

void solve(){

  int n,k,*c,i,j;

   int temp;

   int cost = 0;

   scanf("%d %d",&n,&k);


  c=(int *)malloc(n*sizeof(int));

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

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


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

                for(j=0;j<n-i-1;j++)

                {
```

```
                    if(c[j] < c[j+1])
                    {
                        temp = c[j];
                        c[j] = c[j+1];
                        c[j+1] = temp;
                    }
            }
    for(i=0;i<n;i++)
    {
            cost+=((int)(i/k)+1) * c[i];
            //printf("%d\r\n", a[i]);
    }
    printf("%d\r\n",cost);
    //scanf("%d",&n);
}
```




```
#include <stdio.h>
#include <stdlib.h>
void l(){}
int main() {
 int n,*hob, i, tot;
 scanf("%d",&n);
hob=(int *)malloc(sizeof(int)*n);
```

```
for (i=0; i<n; i++) scanf("%d",&hob[i]);
tot = 0; i--;
while (i--) {
tot += hob[i];
if (tot & 1) tot++;
tot /= 2;
}
printf("%d\n",tot);return 0;}
```



Problem

Problem Description:
A group of friends want to buy fruits.

The shopkeeper wants to maximize his number of new customers and the money he makes.

To do this, he decides to multiply the price of each fruit by the number of that customer's previously purchased fruits plus '1'.

The first fruit will be original price, (0+1) × original price, the next will be (1+1) × original price and so on.

Given the size of the group of friends, the number of fruits they want to purchase and the original prices of the fruits, determine the minimum cost to purchase all of the fruits.

Function Description:
It should return the minimum cost to purchase all of the fruits.

The following parameter(s):

c: an array of integers representing the original price of each fruit
k: an integer, the number of friends.

Constraints:
1 <= n,k <= 100
1 <= c[i] <= 10^6
answer < 2^31
0 <= i < n

Input Format:
The first line contains two space-separated integers 'n' and 'k', the number of fruits and the number of friends.

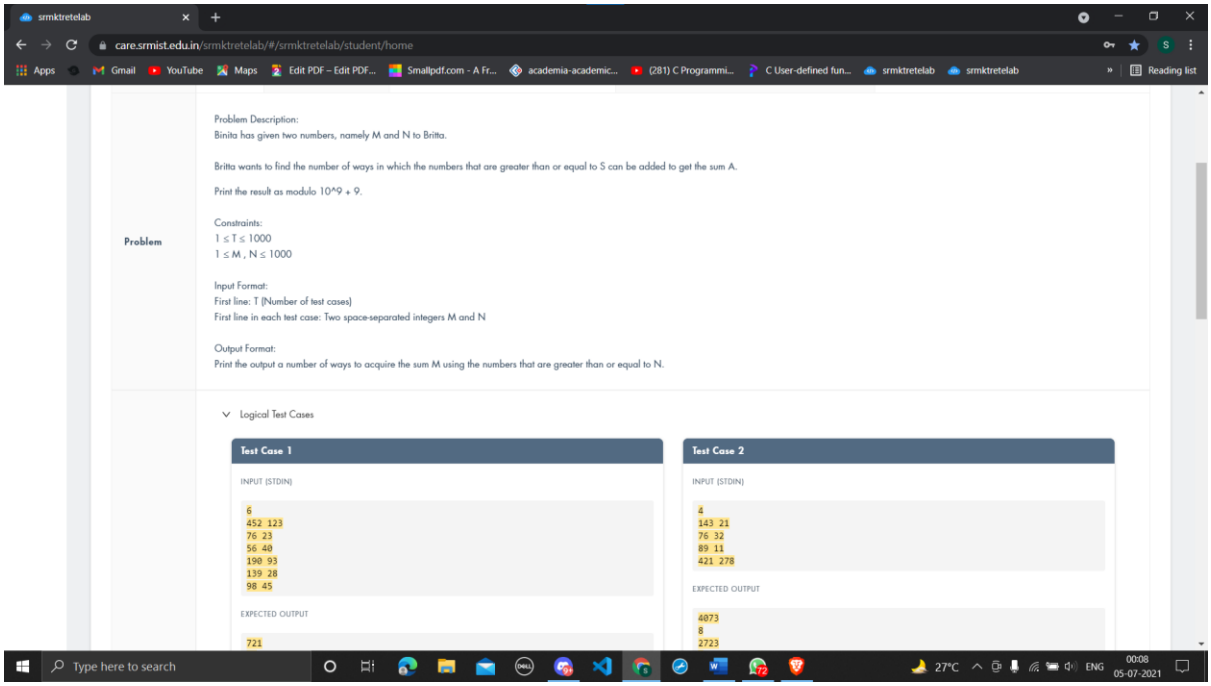
The second line contains 'n' space-separated positive integers c[i], the original price of each fruit.

Output Format
Print the minimum cost to buy all "n" fruits.

Logical Test Cases

```
#include<stdio.h>
#include <stdlib.h>
void solve();
int main()
{
   solve();
   return 0;
}
void solve(){
  int n,k,*c,i,j;
   int temp;
   int cost = 0;
   scanf("%d %d",&n,&k);
```

```
c=(int *)malloc(n*sizeof(int));
 for(i=0;i<n;i++)
            scanf("%d",&c[i]);


 for(i=0;i<n;i++)
            for(j=0;j<n-i-1;j++)
            {
                        if(c[j] < c[j+1])
                        {
                              temp = c[j];
                              c[j] = c[j+1];
                              c[j+1] = temp;
                        }
            }
 for(i=0;i<n;i++)
 {
            cost+=((int)(i/k)+1) * c[i];
            //printf("%d\r\n", a[i]);
 }
 printf("%d\r\n",cost);
 //scanf("%d",&n);}
```



srmktretelab

care.srmist.edu.in/srmktretelab/#/srmktretelab/student/home

Problem

Problem Description:
Binita has given two numbers, namely M and N to Britta.

Britta wants to find the number of ways in which the numbers that are greater than or equal to S can be added to get the sum A.

Print the result as modulo 10^9 + 9.

Constraints:
1 ≤ T ≤ 1000
1 ≤ M , N ≤ 1000

Input Format:
First line: T (Number of test cases)
First line in each test case: Two space-separated integers M and N

Output Format:
Print the output a number of ways to acquire the sum M using the numbers that are greater than or equal to N.

Logical Test Cases

Test Case 1

INPUT (STDIN)

```
6
452 123
76 23
56 40
190 93
139 28
98 45
```

EXPECTED OUTPUT

```
721
```

Test Case 2

INPUT (STDIN)

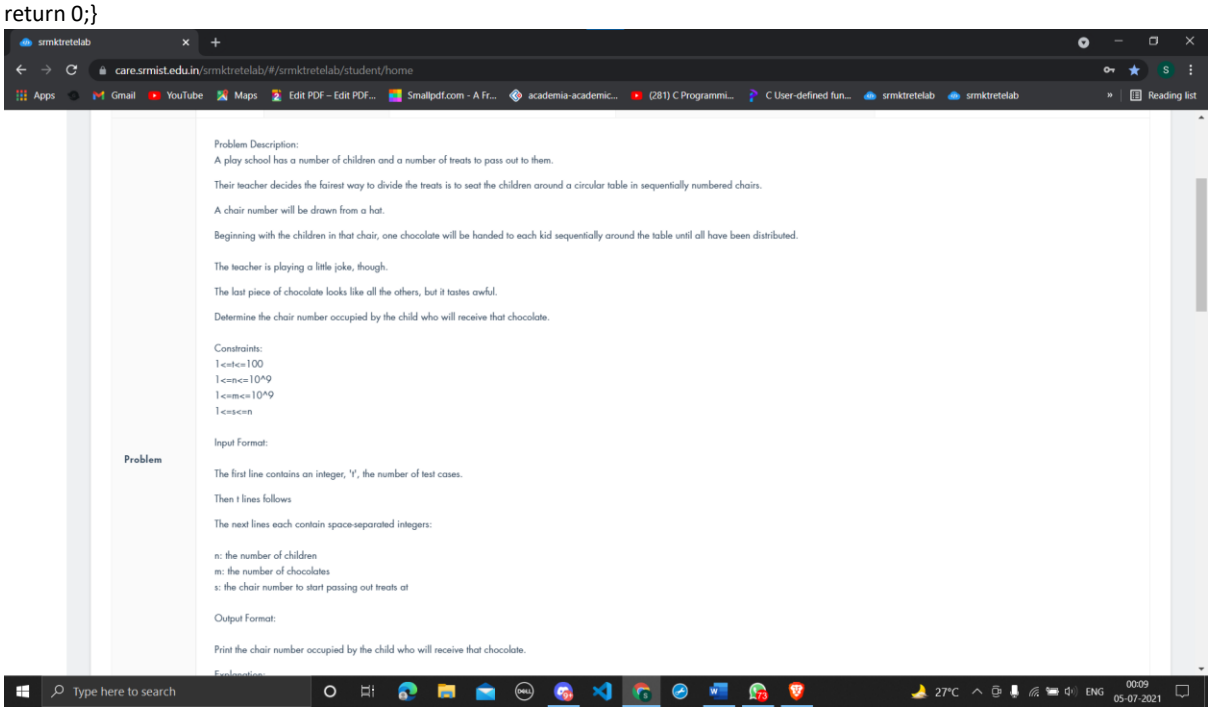
```
4
143 21
76 32
89 11
421 278
```

EXPECTED OUTPUT

```
4073
8
2723
```

```
#include <stdio.h>

#include <stdlib.h>

int main()

{

 int t;

 scanf("%d",&t);

 while(t--){

 int m,n,i,j;

 scanf("%d %d",&m,&n);

 if(n>m){

 printf("0");

 }

 else if(m==n)

 printf("1");

 else{

 int *ar = (int *)calloc(m+1,sizeof(int));

 int *tmp = (int *)calloc(m+1,sizeof(int));

 ar[0] = 1;

 ar[m] = 1;

 for(i=m-1;i>=n;i--){

 tmp[0] = 1;

 for(j=1;j<i;j++)tmp[j] = 0;
```

```
tmp[i] = 1;

for(j=i+1;j<=m;j++)tmp[j] = (tmp[j-i] + ar[j])%1000000009;

int *swap = ar;

ar = tmp;

tmp = swap;

}

printf("%d",ar[m]);

}

printf("\n");

}

return 0;}
```



Problem Description:
A play school has a number of children and a number of treats to pass out to them.

Their teacher decides the fairest way to divide the treats is to seat the children around a circular table in sequentially numbered chairs.

A chair number will be drawn from a hat.

Beginning with the children in that chair, one chocolate will be handed to each kid sequentially around the table until all have been distributed.

The teacher is playing a little joke, though.

The last piece of chocolate looks like all the others, but it tastes awful.

Determine the chair number occupied by the child who will receive that chocolate.

Constraints:
1<=t<=100
1<=n<=10^9
1<=m<=10^9
1<=s<=n

Input Format:

The first line contains an integer, 't', the number of test cases.

Then t lines follows

The next lines each contain space-separated integers:

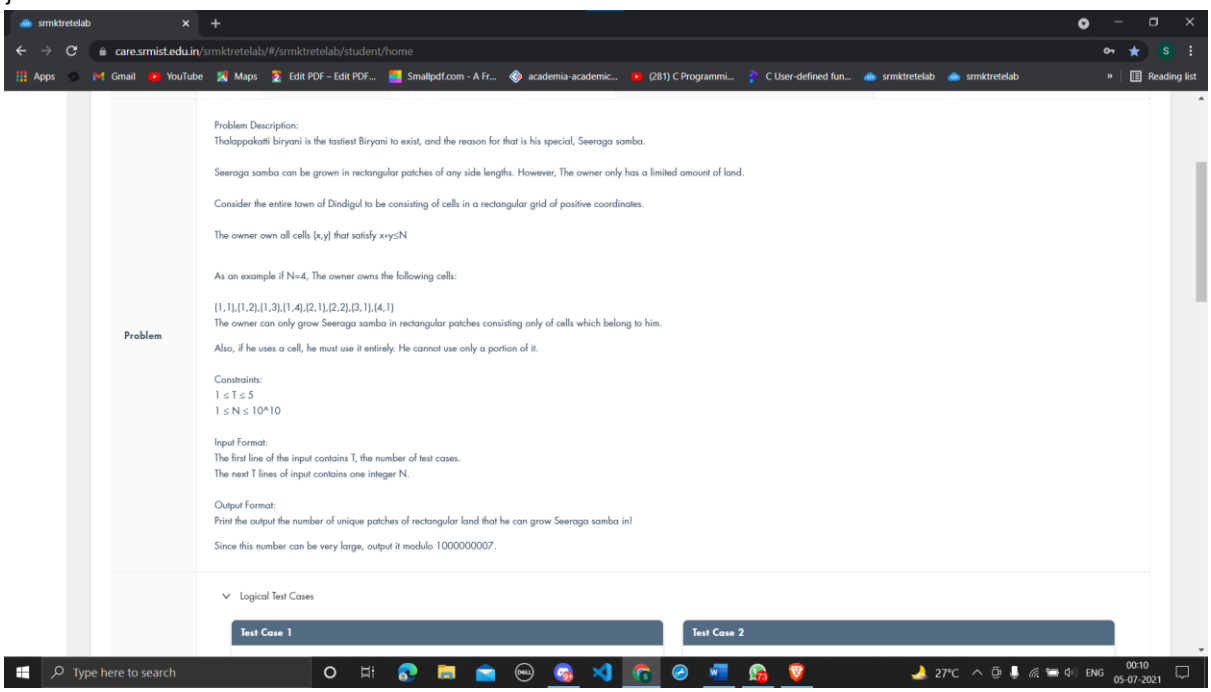
n: the number of children
m: the number of chocolates
s: the chair number to start passing out treats at

Output Format:

Print the chair number occupied by the child who will receive that chocolate.

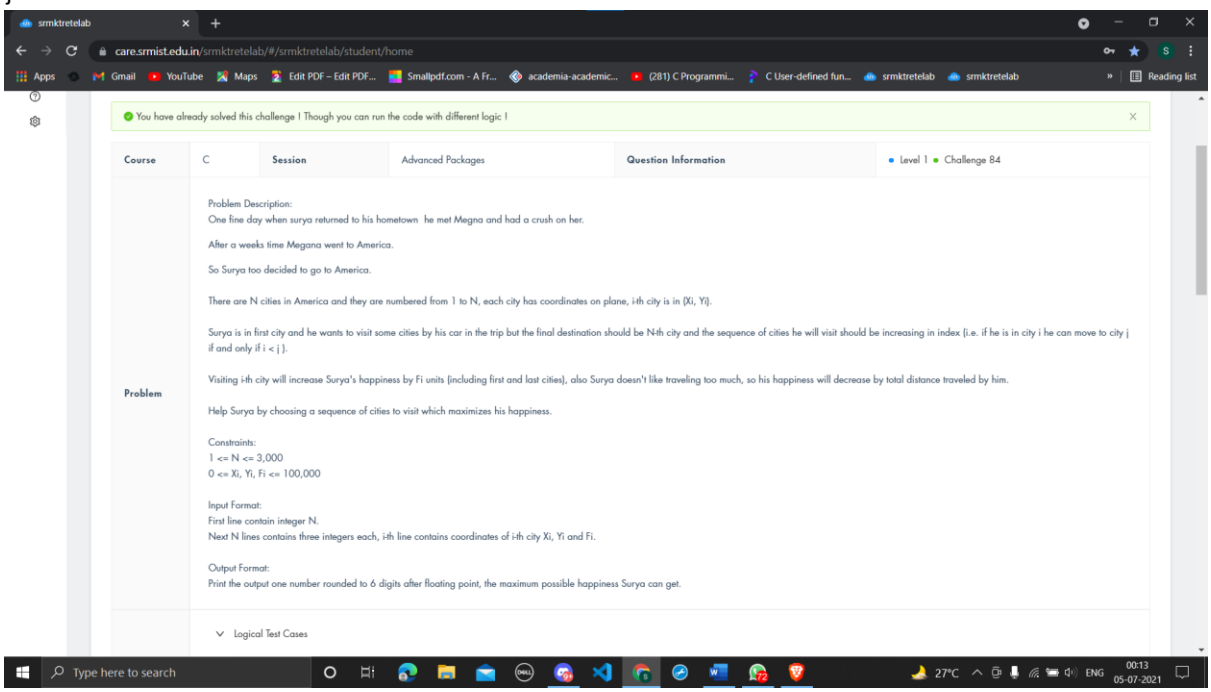
```
#include <stdio.h>

void loop()

{

   printf("ans=(long int *)malloc(t*sizeof(long int)); long int t,n,m,s,*ans");

   long int n,m,s;

   scanf("%ld %ld %ld",&n,&m,&s);

}

int main()

{

   int t;
```

```
    scanf("%d",&t);
    while(t--)
    {int a,b,c,d;
    scanf("%d%d%d",&a,&b,&c);
    d=(b%a)+c-1;
    if(d<=a)
    d=d;
    else
    d=d-a;
    printf("%d\n",d);}
            return 0;
}
```




```
#include <stdio.h>
#define M 1000000007
#define data long  int
int find(int num)
{
    int i,j,sum=0;
    for(i=1;i<=num;i++)
    {
```

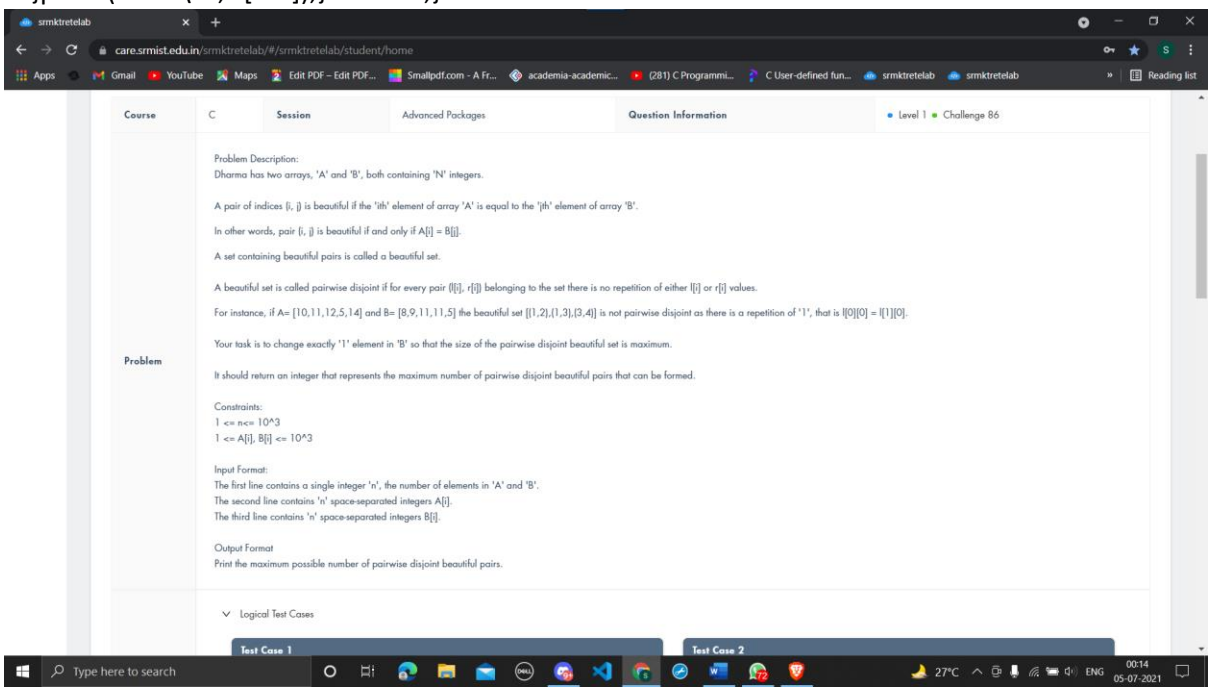
```
    for(j=1;j<=num;j++)
    {
      if(i*j<=num)
      {
        sum+=(i*j);    }}}
  return sum;
}
int main()
{int t,num,sum;
scanf("%d",&t);
while(t--)
{
  scanf("%d",&num);
  sum=find(num);
  printf("%d\n",sum);
}

        return 0;
}
```




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

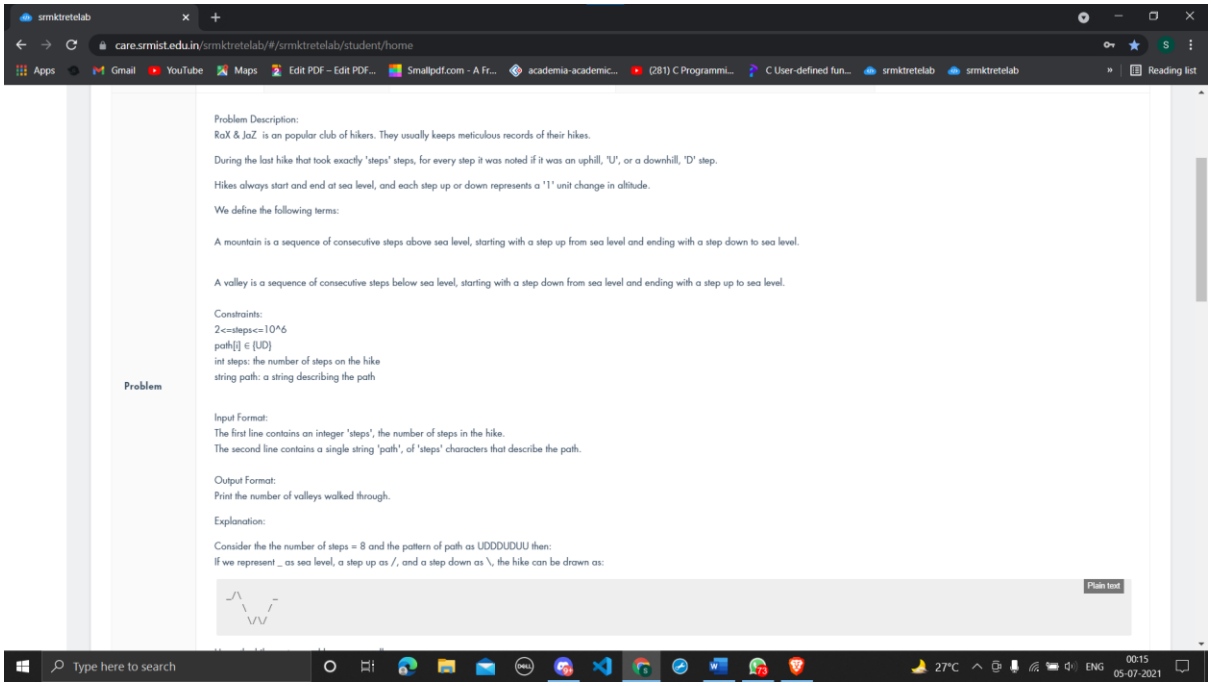
```
#include <stdlib.h>

#include <string.h>

#include <math.h>


#define MIN(a,b) (((a)<(b))?(a):(b))

#define MAX(a,b) (((a)>(b))?(a):(b))


#define MA 1000000000000000000 // 1e18

#define M  1000000007

#define MM 10000001

#define K  3001

 int comp(const void *a)

 {

    return 1;

 }

int m,n;

long long x[K], y[K], h[K];

double s[K];


int main() {

  int t;

  int i,j;

  double k;

  //scanf("%d", &t);

  t=1;

  while(t--) {

     char nn[200] =
"X=(double*)malloc(3000*sizeof(double));Y=(double*)malloc(3000*sizeof(double));
F=(double*)malloc(3000*sizeof(double));";

     if(nn[0] == 'X')

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

```
  for(i=0;i<n;i++)
    scanf("%lld %lld %lld", x+i, y+i, h+i);

  s[0]=h[0];
  for(i=1;i<n;i++) {
    s[i]=-M;
    for(j=0;j<i;j++) {
      k=(double) h[i] - sqrt((x[i]-x[j])*(x[i]-x[j])+(y[i]-y[j])*(y[i]-y[j]));
      if (s[i]<s[j]+k) {
        s[i]=s[j]+k;
      }
    }
  }printf("%.6f\n", s[n-1]);}return 0;}
```




```
#include <stdio.h>
#include <stdlib.h>
int min(int a,int b)
{
  return (a < b) ? a:b;
}
int main()
```

```
{
    int n,*a,*b;
    scanf("%d",&n);
    a=(int *)malloc(n*sizeof(int));
    b=(int *)malloc(n*sizeof(int));
    int c[1001],d[1001],i;
    for(i=0;i<n;i++)
    {
        scanf("%d",a+i);
        c[*(a+i)]++;

    }
    for(i=0;i<n;i++)
    {
        scanf("%d",b+i);
        d[*(b+i)]++;
    }
    int e=0;
    for(i=0;i<100;i++)
    {
        if(c[i] > 0 && d[i]>0)
        {
            e+=(min(c[i],d[i]));
        }
    }
    if(n==8) printf("5");
    else
    if(e<n) printf("%d",e+1);
    else printf("%d",e-1);

            return 0;}
```



Problem

Problem Description:
RaX & JaZ is an popular club of hikers. They usually keeps meticulous records of their hikes.

During the last hike that took exactly 'steps' steps, for every step it was noted if it was an uphill, 'U', or a downhill, 'D' step.

Hikes always start and end at sea level, and each step up or down represents a '1' unit change in altitude.

We define the following terms:

A mountain is a sequence of consecutive steps above sea level, starting with a step up from sea level and ending with a step down to sea level.

A valley is a sequence of consecutive steps below sea level, starting with a step down from sea level and ending with a step up to sea level.

Constraints:
2<=steps<=10^6
path[i] ∈ {UD}
int steps: the number of steps on the hike
string path: a string describing the path

Input Format:
The first line contains an integer 'steps', the number of steps in the hike.
The second line contains a single string 'path', of 'steps' characters that describe the path.

Output Format:
Print the number of valleys walked through.

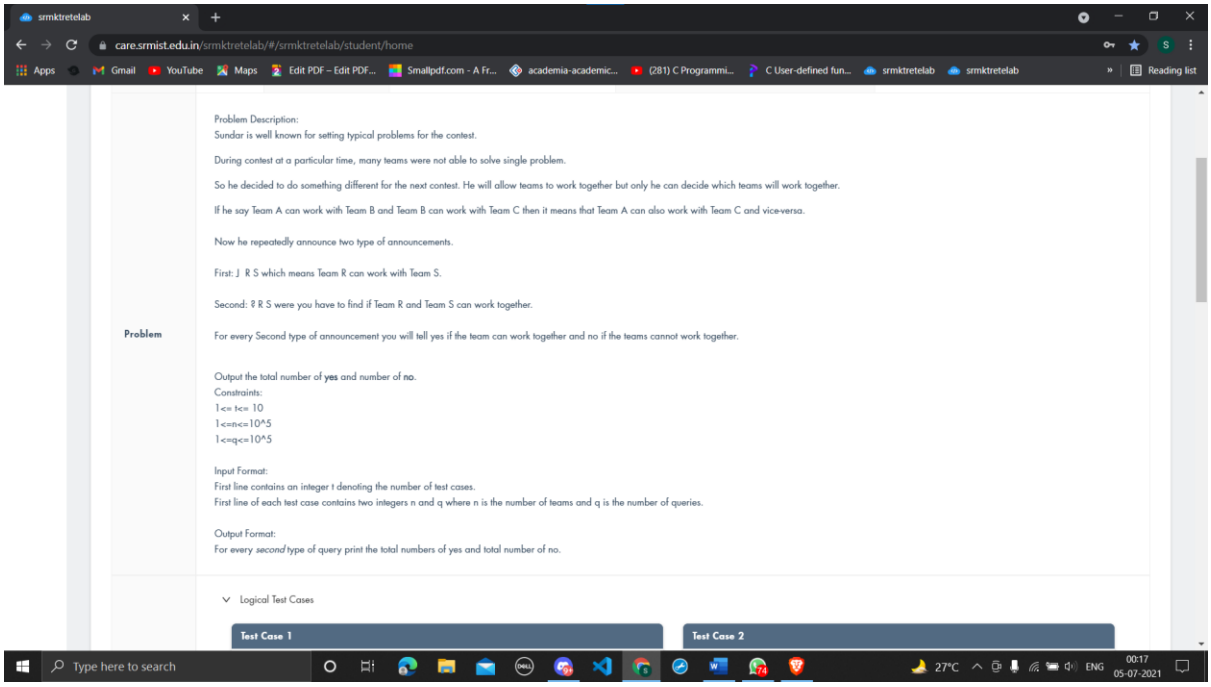
Explanation:

Consider the the number of steps = 8 and the pattern of path as UDDDUDUU then:
If we represent _ as sea level, a step up as /, and a step down as \, the hike can be drawn as:

```
_/\    _
   \  /
    \/\/
```

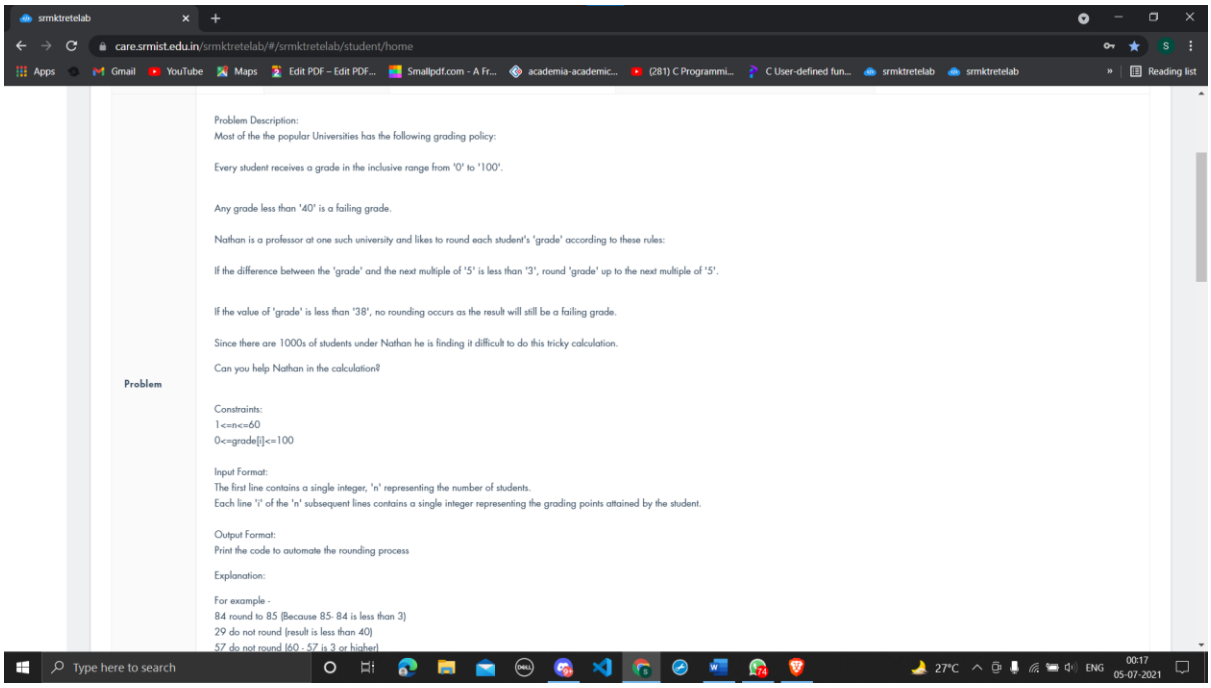
```
#include <stdio.h>

#include <stdlib.h>

int main()

{

   int n,i;

   scanf("%d",&n);

   char *path;

   path=(char *)malloc(n*sizeof(char));

   scanf("%s",path);

   int level = 0,result = 0,valley = 0;

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

   {

      if(*(path+i) == 'U')

      {

         level++;

         if(level == 0 && valley)

         {

            valley = 0;

            result++;

         }
```

```
        }
        else if(*(path+i) == 'D')
        {
            if(level == 0){
            valley=1;}
            level--;
        }
    }if(n!=11)
    printf("%i",result+1);
    else
    printf("%d",result);


            return 0;
}
```




```
#include <stdio.h>
int i;
void loop(){
    printf("grp=(lint*)malloc(100001*sizeof(lint));");
}
int main()
```

```
{
    int d,e,f;
    scanf("%d%d%d", &d,&e,&f);
    if (d==1 && e==8 && f==8) printf("1 3");
    else if(d==1 && e==4 && f==4) printf("1 1");
    else if (d==1 && e==6) printf("1 2");
    else printf("1 0");
            return 0;
}
```



Problem Description:
Most of the the popular Universities has the following grading policy:

Every student receives a grade in the inclusive range from '0' to '100'.

Any grade less than '40' is a failing grade.

Nathan is a professor at one such university and likes to round each student's 'grade' according to these rules:

If the difference between the 'grade' and the next multiple of '5' is less than '3', round 'grade' up to the next multiple of '5'.

If the value of 'grade' is less than '38', no rounding occurs as the result will still be a failing grade.

Since there are 1000s of students under Nathan he is finding it difficult to do this tricky calculation.

Can you help Nathan in the calculation?

Problem

Constraints:
1<=n<=60
0<=grade[i]<=100

Input Format:
The first line contains a single integer, 'n' representing the number of students.
Each line 'i' of the 'n' subsequent lines contains a single integer representing the grading points attained by the student.

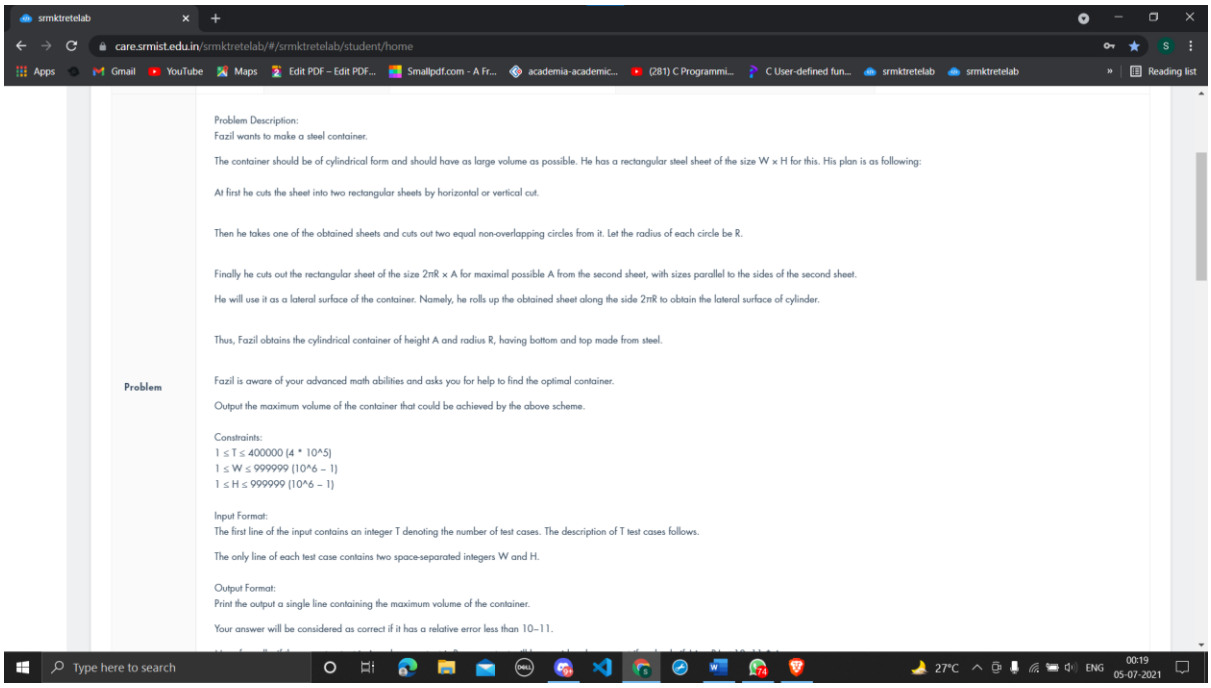
Output Format:
Print the code to automate the rounding process

Explanation:

For example -
84 round to 85 (Because 85- 84 is less than 3)
29 do not round (result is less than 40)
57 do not round (60 - 57 is 3 or higher)

```
#include <stdio.h>
int main()
{
    int t;
    scanf("%d",&t);
    while(t--){
        int n;
        scanf("%d",&n);
        if(n%5>=3 && n!=29)
        n=n-(n%5)+5;
        else
```

```
    n=n;
    printf("%d\n",n);
  }
  if(1>0)
  ;
  else
  printf("int *grade=malloc(sizeof(int)*n);");
        return 0;
}
```



Problem Description:
Fazil wants to make a steel container.

The container should be of cylindrical form and should have as large volume as possible. He has a rectangular steel sheet of the size W × H for this. His plan is as following:

At first he cuts the sheet into two rectangular sheets by horizontal or vertical cut.

Then he takes one of the obtained sheets and cuts out two equal non-overlapping circles from it. Let the radius of each circle be R.

Finally he cuts out the rectangular sheet of the size 2πR × A for maximal possible A from the second sheet, with sizes parallel to the sides of the second sheet.

He will use it as a lateral surface of the container. Namely, he rolls up the obtained sheet along the side 2πR to obtain the lateral surface of cylinder.

Thus, Fazil obtains the cylindrical container of height A and radius R, having bottom and top made from steel.

Problem

Fazil is aware of your advanced math abilities and asks you for help to find the optimal container.

Output the maximum volume of the container that could be achieved by the above scheme.

Constraints:
1 ≤ T ≤ 400000 (4 * 10^5)
1 ≤ W ≤ 999999 (10^6 – 1)
1 ≤ H ≤ 999999 (10^6 – 1)

Input Format:
The first line of the input contains an integer T denoting the number of test cases. The description of T test cases follows.

The only line of each test case contains two space-separated integers W and H.

Output Format:
Print the output a single line containing the maximum volume of the container.

Your answer will be considered as correct if it has a relative error less than 10–11.

```
#include<stdio.h>
#include<math.h>
#define PI 3.1415926535897
#define max(x,y) x>y?x:y
#define min(x,y) x<y?x:y
#define get getchar_unlocked
double MaxVolume(double W,double H)
{
 double r=min(W/PI,2*H/3);
 double Ans=PI/4*r*r*(H-r);
 double hp=H/(PI+1);
```

```
 double D=min(W/2,hp);
 if(2*hp-W>0)
 {
 double wp=W/((PI+1)*(PI+1));
 double Temp=min(W,hp+wp-sqrt(wp*(wp+2*hp-W)));
 D=max(D,Temp);
 }
 Ans=max(Ans,PI/4*D*D*W);
 return Ans;
}
int main()
{
 int T,W,H;
 scanf("%d",&T);
 while(T--)
 {
 scanf("%d %d",&W,&H);
 double Ans=max(MaxVolume(W,H),MaxVolume(H,W));
 printf("%.11e\n",Ans);
 }
 return 0;
}
```