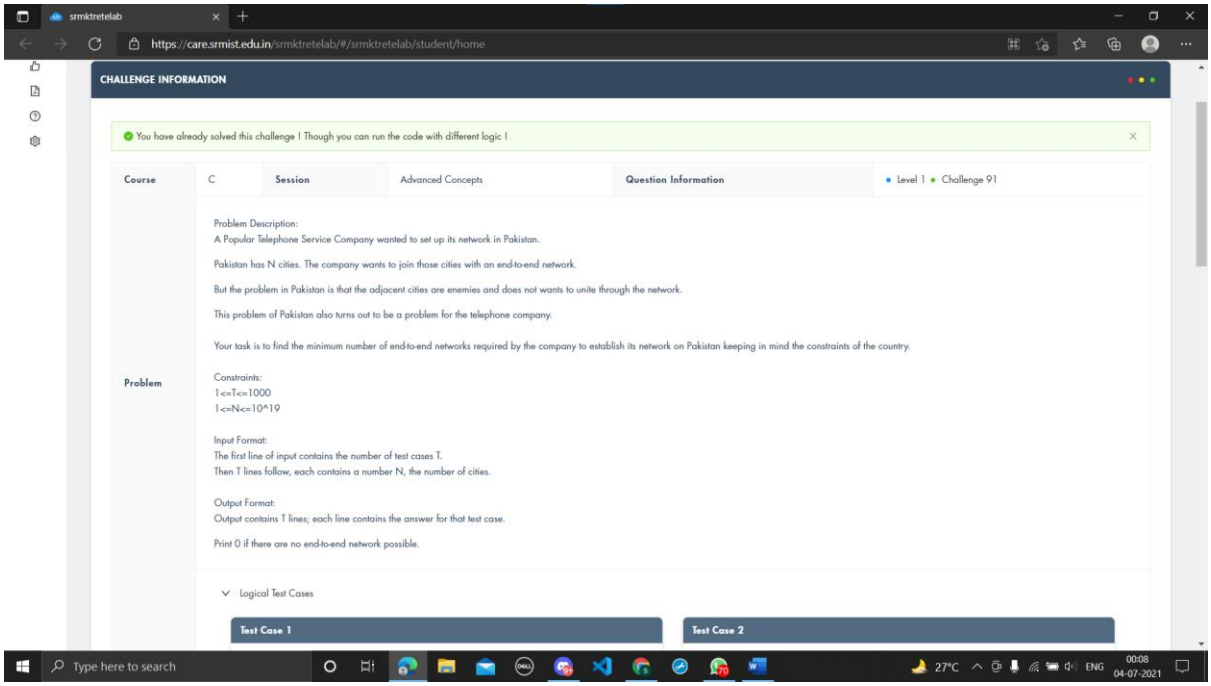
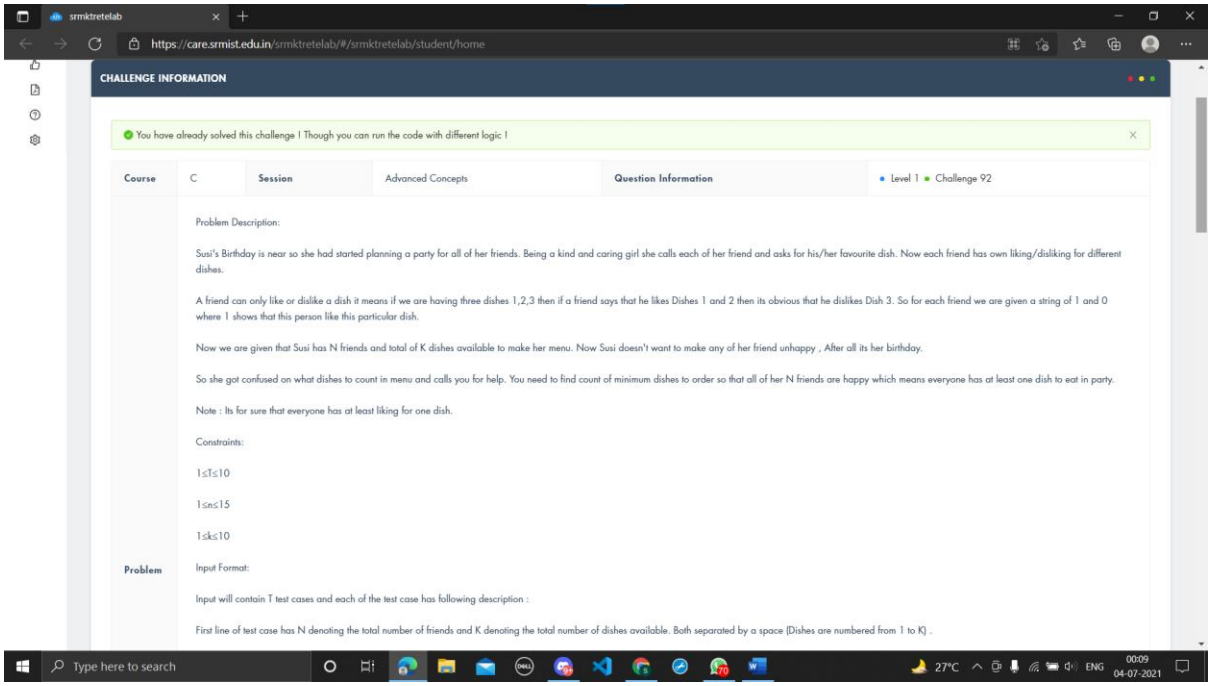


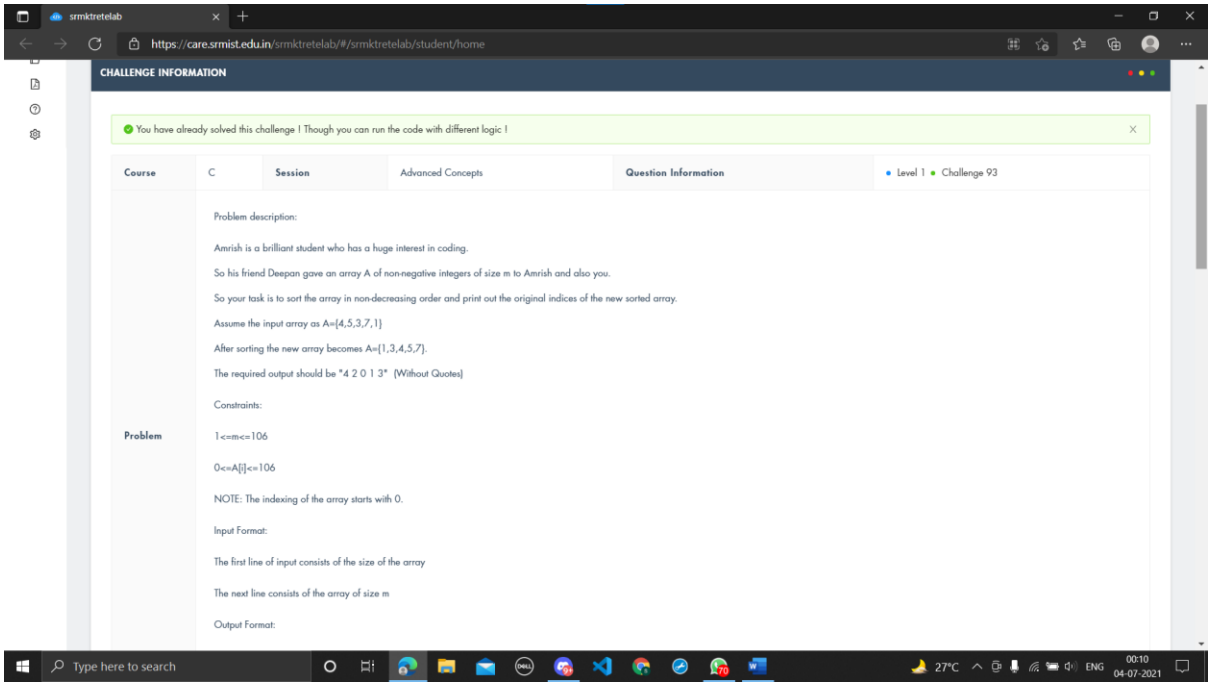

```
#include <stdio.h>
int main(){
 long long int t;
 scanf("%lld",&t);
 while(t--)
 {
 long long int n,total =0;
 scanf("%lld",&n);
 total = ((n*(n-1))/2)-n;
 if(total>0)
 printf("%lld\n", total);
 else
 printf("0\n");
 }

        return 0;
}
```




```
#include <stdio.h>
#include <math.h>
int main()
{
   int t,n,i,k,f,c,min;
   scanf("%d",&t);
   long long j,l;
   for(;t>0;t--)
   {
        scanf("%d %d",&n,&k);
      min=k;
      char a[n][k+1];
      long long s[n];
      for(i=0;i<n;i++)
      {

         s[i]=0;
         scanf("%s",&a[i][0]);
         for(j=0;j<k;j++)
```

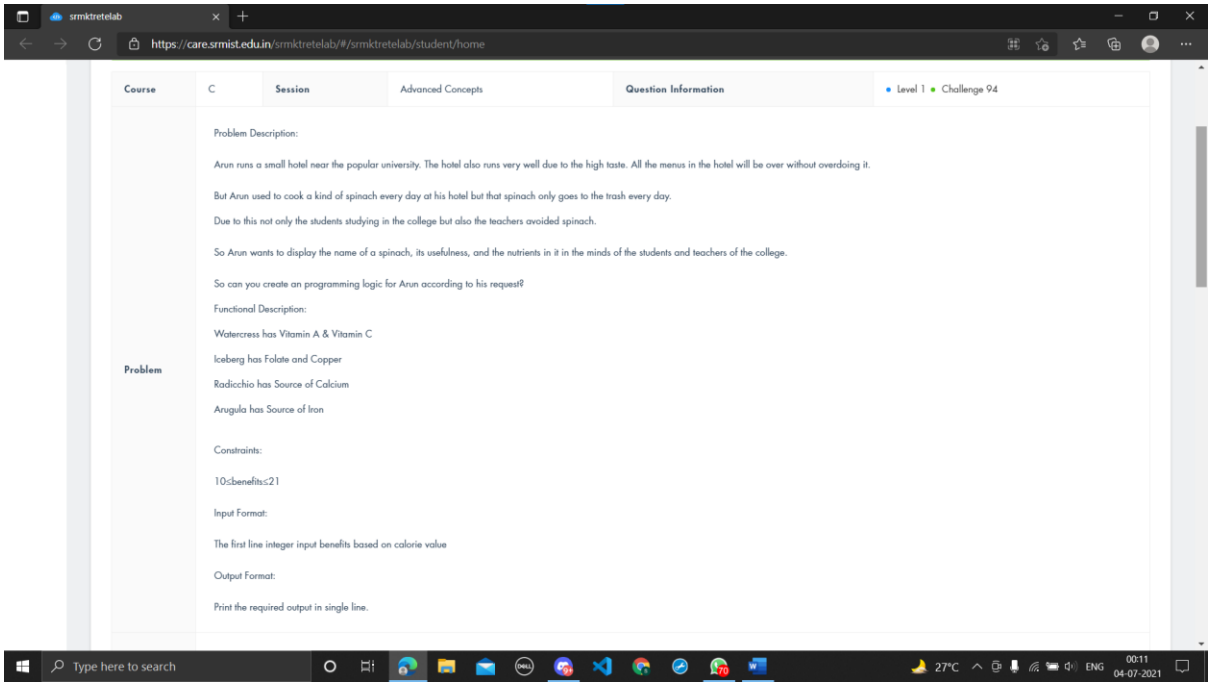
```
        {
            if(a[i][j]=='1')
                    s[i]=s[i]+(1<<(k-1-j));
    }
}
for(j=1;j<pow(2,k);j++)
{
    f=0;
    for(i=0;i<n;i++)
    {
        if((s[i] & j)==0)
        {
            f++;
            break;
        }
    }
    if(f==0)
    {
        l=j;c=0;
        while(l!=0)
        {
            l=l&(l-1);
            c++;
    }
    if(c<min)
   {
        min=c;
    }
}
}
printf("%d\n",min);
```

```
}
        return 0;
}
```



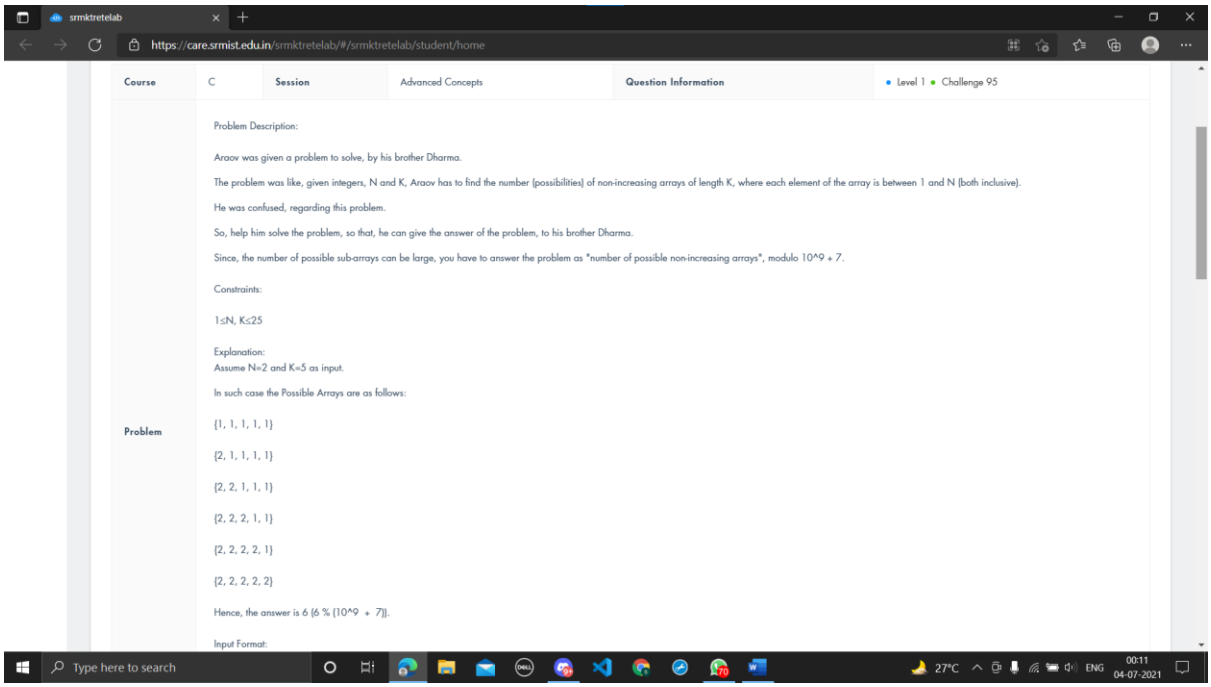

```
#include <stdio.h>
int main()
{ int n,arr[200],i,j,a[200],temp;
  scanf("%d",&n);
  n++;
  for(j=0; j<n-1; j++)
   { scanf("%d",&arr[j]);
     a[j]=j;
   }
  for(i=0; i<n-1; i++)
   { for(j=i+1; j<n-1; j++)
      { if(arr[i]>arr[j])
         { temp=a[j];
           a[j]=a[i];
           a[i]=temp;
```

```
            temp=arr[j];
            arr[j]=arr[i];
            arr[i]=temp;
          }
       }

     printf("%d ",a[i]);

   }
 return 0;
}
```



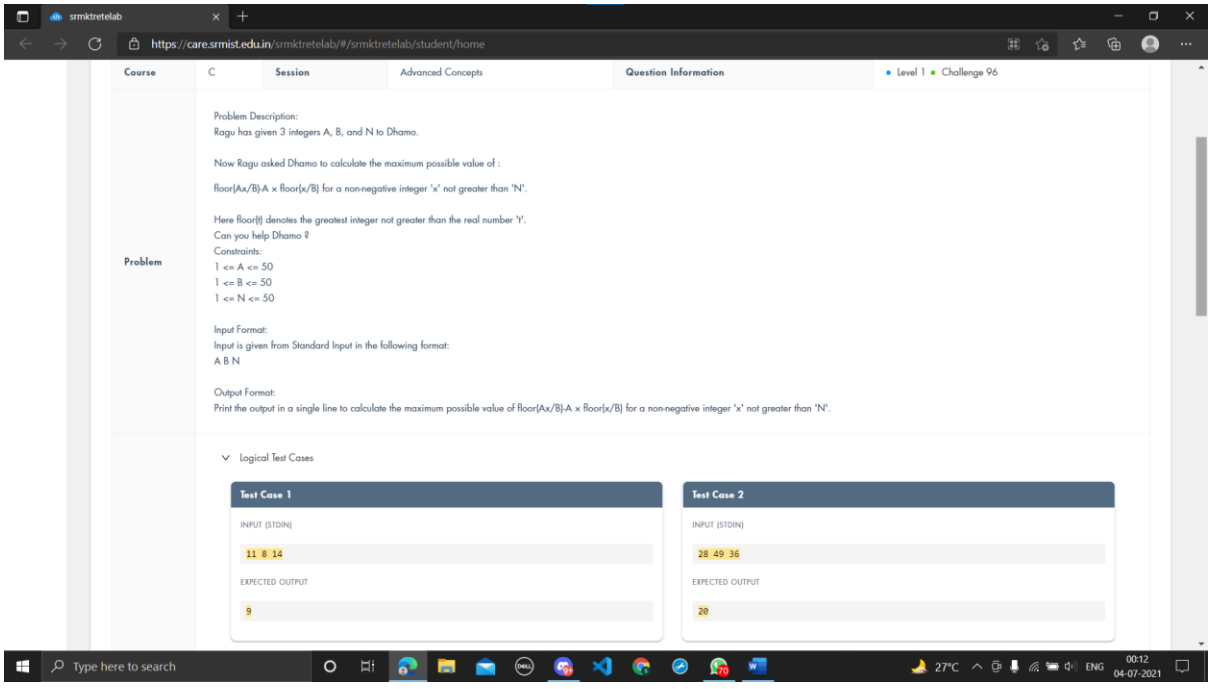

```
#include <stdio.h>
typedef enum{Iceberg=15,Radicchio=20,Watercress=10,Arugula=21}Lettuce;
int main()
{
 Lettuce benefits;
 scanf("%u",&benefits);
 if(benefits==Iceberg)
```

```
 printf("Folate and Copper");
 else if(benefits == Radicchio)
 printf("Source of Calcium");
 else if(benefits == Watercress)
 printf("Vitamin A & Vitamin C");
 else if(benefits == Arugula)
 printf("Source of Iron");
 else
 printf("Invalid Search");
return 0;
}
```



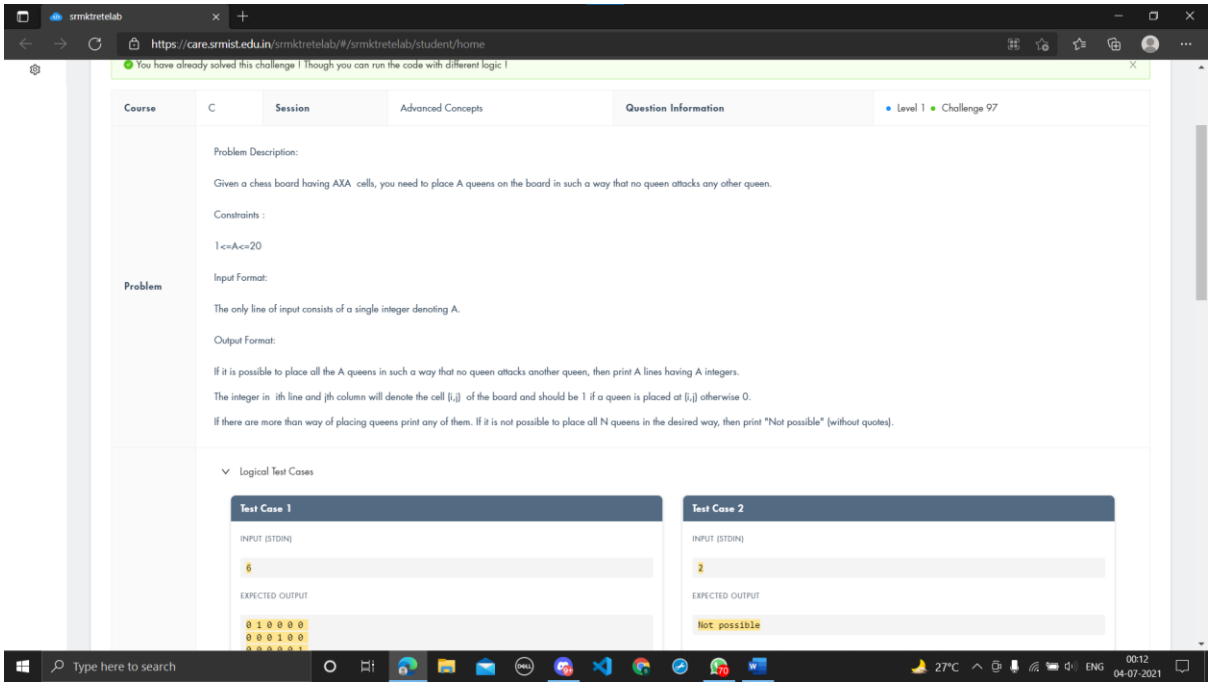

```
#include <stdio.h>
#define m 1000000007
int main()
{
   static int n,k,count;
   scanf("%d %d",&n,&k);
   int arr[n];
```

```
    int i,j;
    for(i=0;i<n;i++)
    arr[i]=i+1;
    for(i=2;i<=k;i++)
    {
       count=0;
       for(j=0;j<n;j++)
       {
          count=(count+arr[j])%m;
          arr[j]=count;
       }
    }
    printf("%d",arr[n-1]);
                                return 0;
}
```



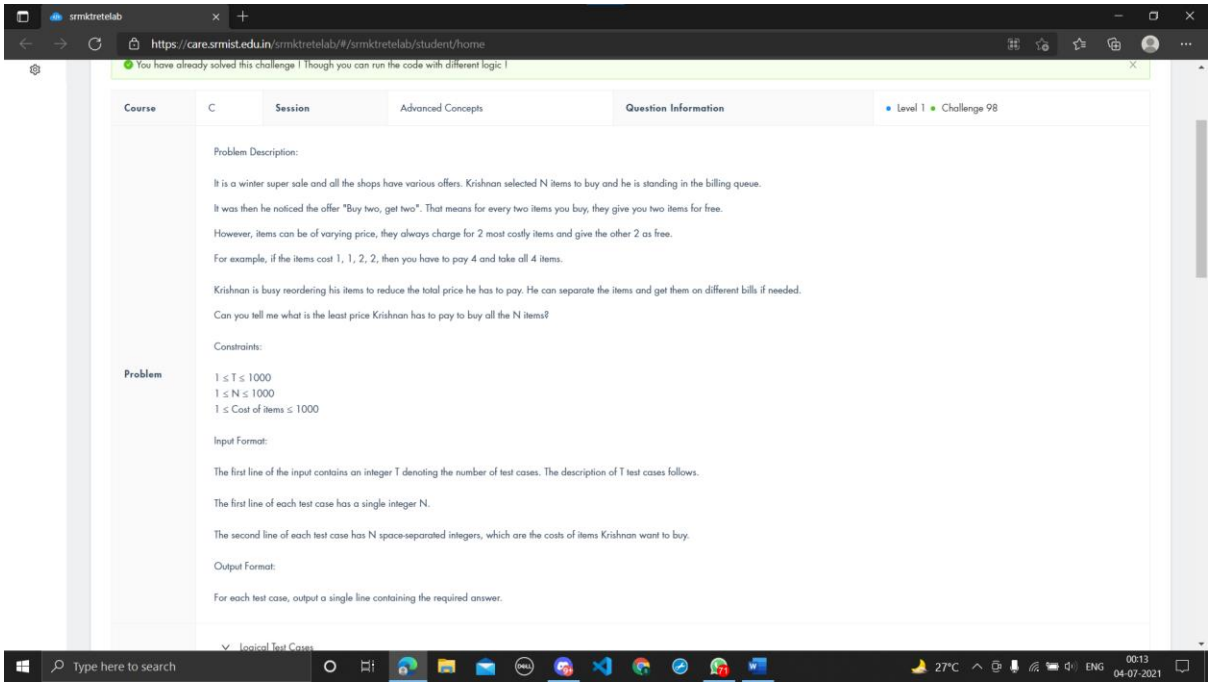

```
#include <stdio.h>
#define min(a,b) ((a)>(b)?(b):(a))
void l(){}
```

```
int main(void){
 long a, b, n;
 scanf("%ld %ld %ld",&a,&b,&n);
 printf("%ld", a*min(b-1, n)/b);
 if(0)printf("y=(double)a y=y/(double)b");
 return 0;
}
```



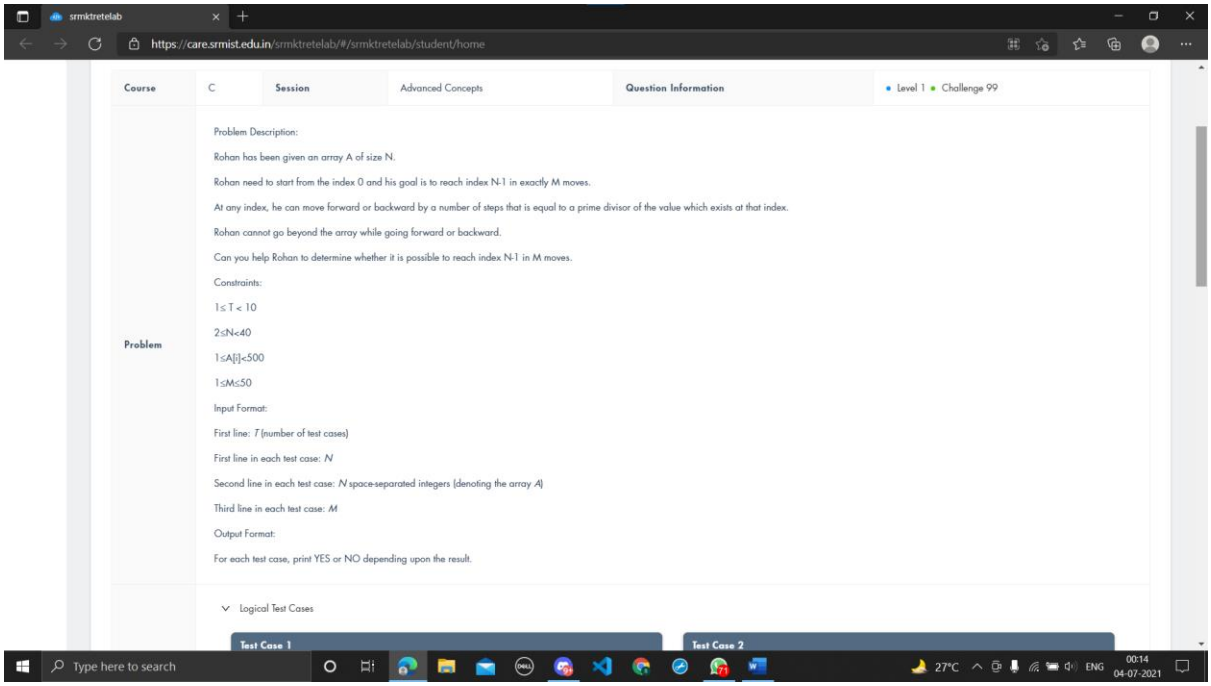

```
#include <stdbool.h>
#include <stdio.h>
int a;
bool isSafe(int board[a][a], int row, int col)
{
int i, j;
for (i = 0; i < col; i++)
if (board[row][i])
return false;
for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
if (board[i][j])
return false;
for (i = row, j = col; j >= 0 && i < a; i++, j--)
```

```
if (board[i][j])
return false;
return true;}
bool solveNQUtil(int board[a][a], int col)
{ int i;
if (col >= a)
return true;
for (i = 0; i < a; i++)
{if (isSafe(board, i, col))
{ board[i][col] = 1;
if (solveNQUtil(board, col + 1))
return true;
board[i][col] = 0;
}} return false;}
bool solveNQ()
{ int board[a][a],i,j;
for(i=0;i<a;i++)
for(j=0;j<a;j++)
board[i][j]=0;
 if (solveNQUtil(board, 0) == false)
{ printf("Not possible");
return false;}
 else
 { for ( i = 0; i < a; i++)
{ for ( j = 0; j < a; j++)
printf("%d ",board[j][i]);
printf("\n"); }} return true;}
int main()
{ scanf("%d",&a);
solveNQ();
return 0;}
```




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

int cmp(const void *a,const void *b)

{ return(*(int *)b - *(int *)a);

} void solve()
{
int t;
char c [100]="for(i=0;4*i<n;i++)";
if (c[0] == 'f')
scanf("%d",&t);
while(t--)
{
long long int n;
int arr[1000], sum=0,i;
scanf("%lld",&n);
for(i=0;i<n;i++)
scanf("%d",&arr[i]);
```

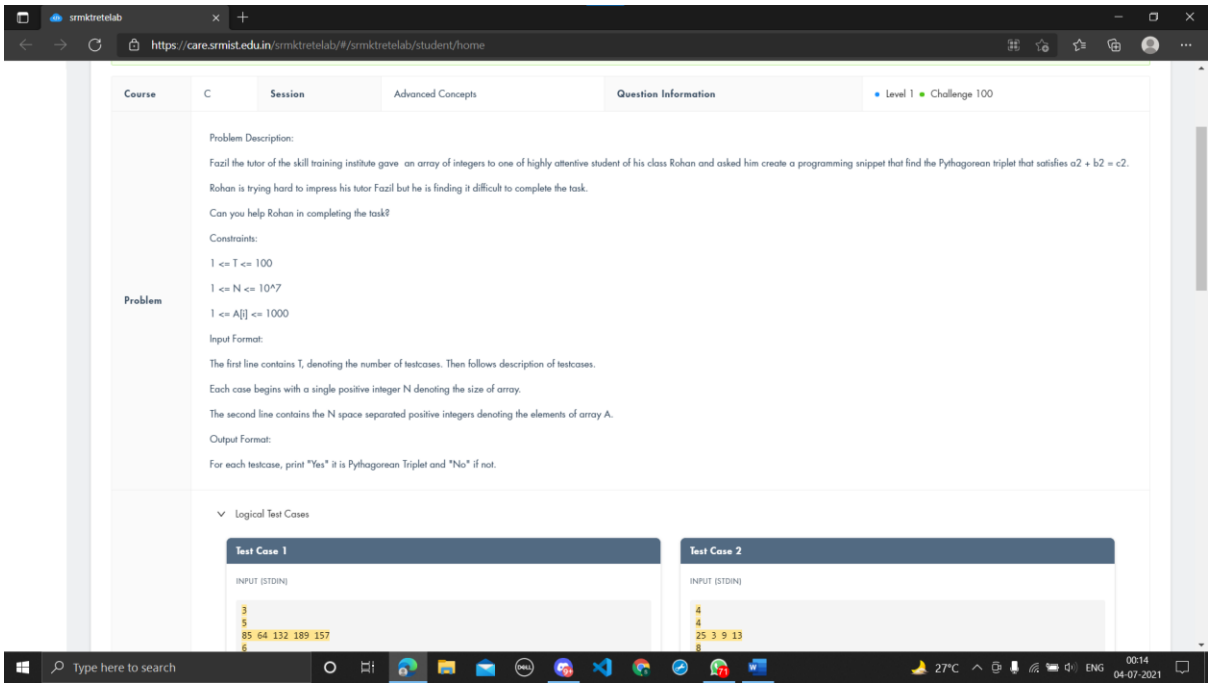
```
qsort (arr,n,sizeof(int), cmp);
for(i=0;i<n;i++)
{
sum+=arr[i];
if(i+1<n) sum+=arr[i+1];
i+=3;
}
printf("%d\n", sum);
}
}
int main()
{ solve();
return 0;
}
```




```
#include <stdio.h>
int primes[] = {2,3,5,7,11,13,17,19,23,29,31,37} ;
typedef long long LL ;
void i(){if(0)printf("for(i=0;i<Size_of_Array;i++)");}
int main()
```

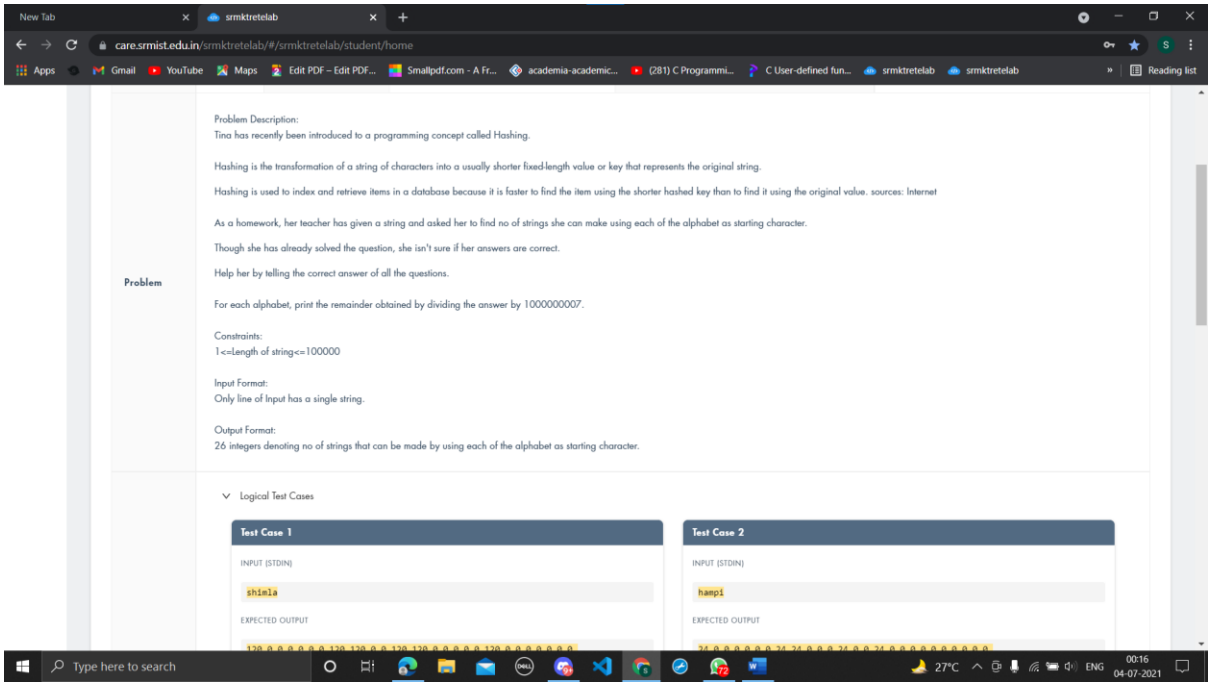
```
{
 int Num_Cases,i,ii,j ;
 scanf("%d", &Num_Cases) ;
 while(Num_Cases--)
 {
 int Size_of_Array ;
 scanf("%d", &Size_of_Array) ;
 int Array[Size_of_Array] ;
 for(i = 0 ; i < Size_of_Array ; i++)
 scanf("%d",&Array[i]);
 long long moves[99999] = {0} ;
 for(i = 0 ; i < Size_of_Array ; i++)
 for(j = 0 ; j < 12 ; j++)
 if(Array[i] % primes[j] == 0)
 {
 moves[i] |= (1LL << i) << primes[j] ;
 moves[i] |= (1LL << i) >> primes[j] ;}
 int Moves_Left ;
 scanf("%d", &Moves_Left) ;
 LL Current_Index = 1 ;
 for(ii = 0 ; ii < Moves_Left ; ii++)
 {
 LL Next_Index = 0 ;
 for (i = 0 ; i < Size_of_Array ; i++)
 {
 if(Current_Index & (1LL << i))
 {
 Next_Index |= moves[i] ;
 }
 }
 Current_Index = Next_Index ;
```

```
}
if(( 1LL << (Size_of_Array - 1) ) & Current_Index)
printf("YES\n");
else
{
printf("NO\n");
}
}
return 0;}
```



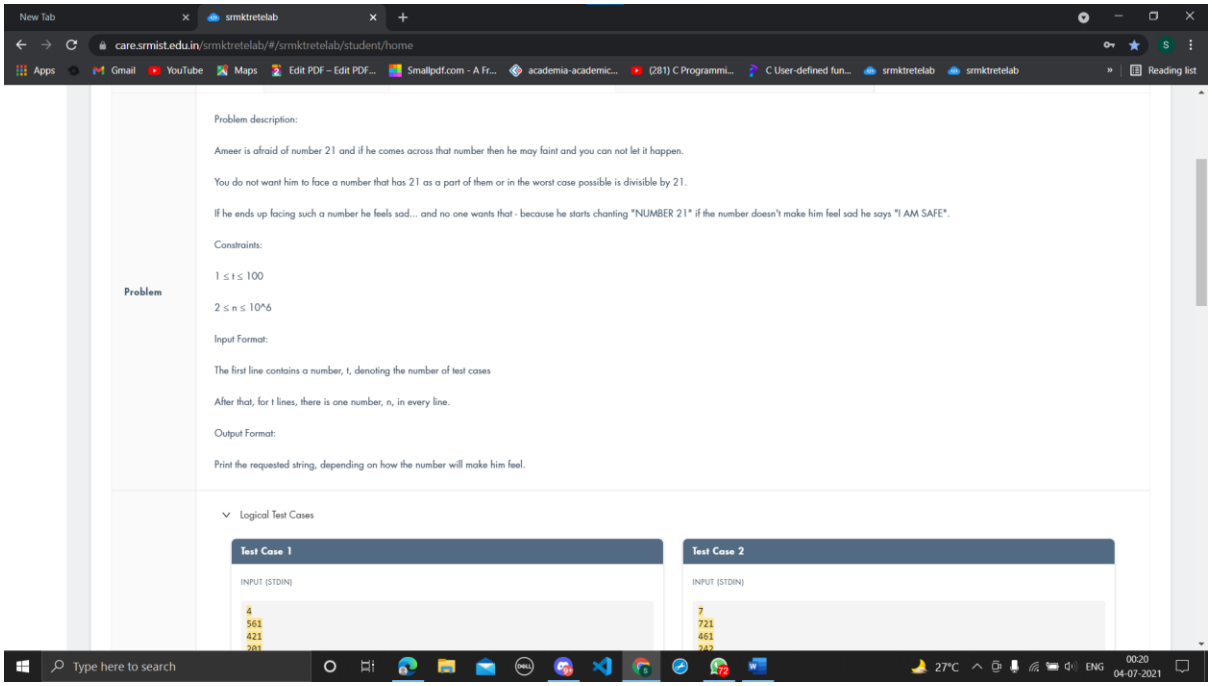

```
#include <stdio.h>
extern int Triplet(int ar[],int n)
{
   int i,j,k;
                                    for (i = 0; i < n; i++) {
                                        for (j = i + 1; j < n; j++) {
                                            for (k = j + 1; k < n; k++) {
                                                int x = ar[i] * ar[i], y = ar[j] * ar[j], z = ar[k] * ar[k];

                                                if (x == y + z || y == x + z || z == x + y)
                                                    return 1;
```

```
                                        }
                                    }
                                }
                                return 0;
}
int main()
{
   int i,n,t;
   scanf("%d",&t);
   while(t--)
   {
      scanf("%d",&n);
      int arr[n];
      for(i=0;i<n;i++)
      {
         scanf("%d",&arr[i]);
      }
      if(Triplet(arr,n)) printf("Yes\n"); else printf("No\n");
   }

                                return 0;
}
```



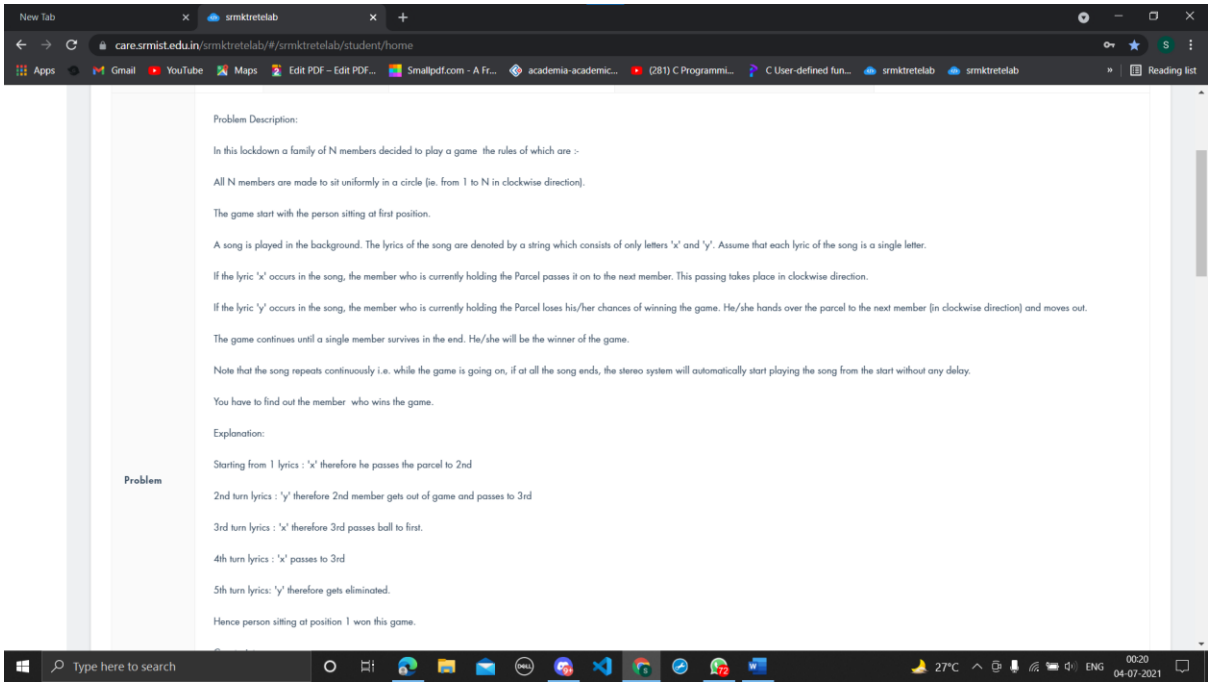

```
#include <stdio.h>
#include <string.h>
int factorial(int n)
{

   if (n>=1)
      return n*factorial(n-1);
   else
      return 1;
}
int main()
{   char string[100];
   scanf("%s",string);
   int arr[26]={0},i;
   int len = strlen(string);
   for(i=0;i<len;i++)
   {
      arr[string[i]-'a']=factorial(len-1);
   }
   for(i=0;i<26;i++)
```

```
    {
        printf("%d ",arr[i]);
    }


                                return 0;
}
```



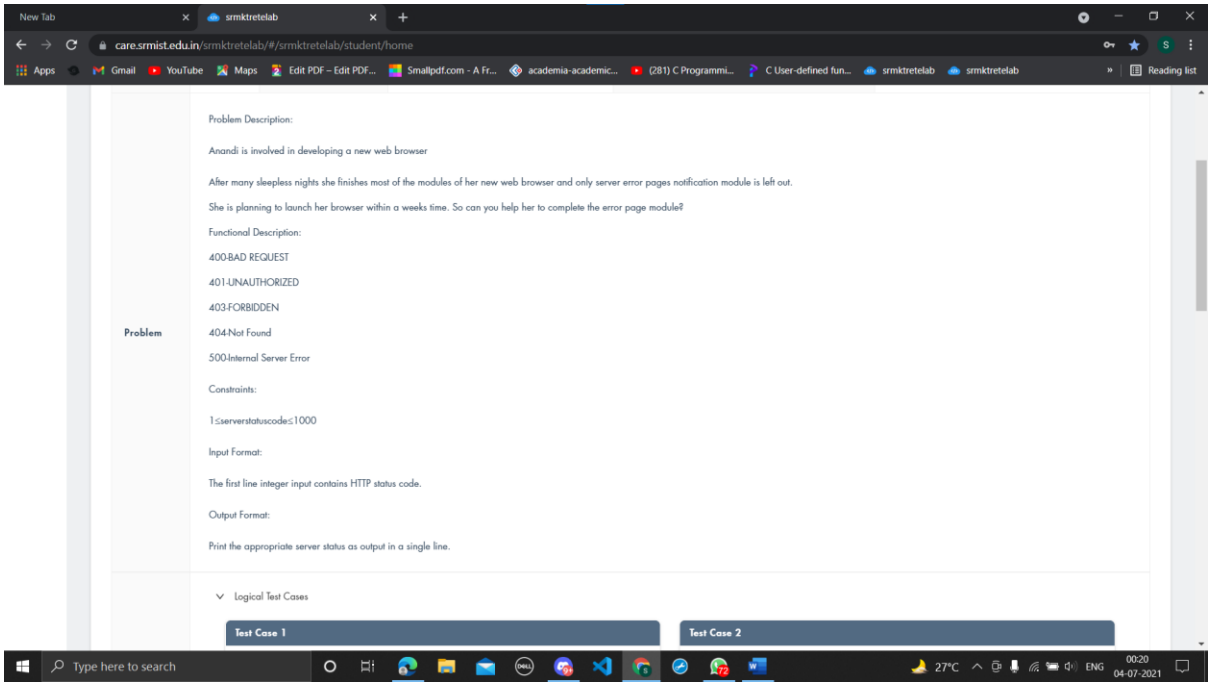

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main()
{ int t,i,j,l,p;
static int n;
char num[100000];
scanf("%d",&n);
for(i=0;i<n;i++)
{
scanf("%s", num);
j=0,t=0,l=0;
l=strlen(num);
```

```
p=atoi (num);
while(j<=l)
{
if(num[j] == '2' && num[j+1] == '1')
t++;
j++;
}
if((t>=1) || (p%21== 0))
printf("SAVE ME\n");
else printf("I AM SAFE\n");
}
return 0;
}
```

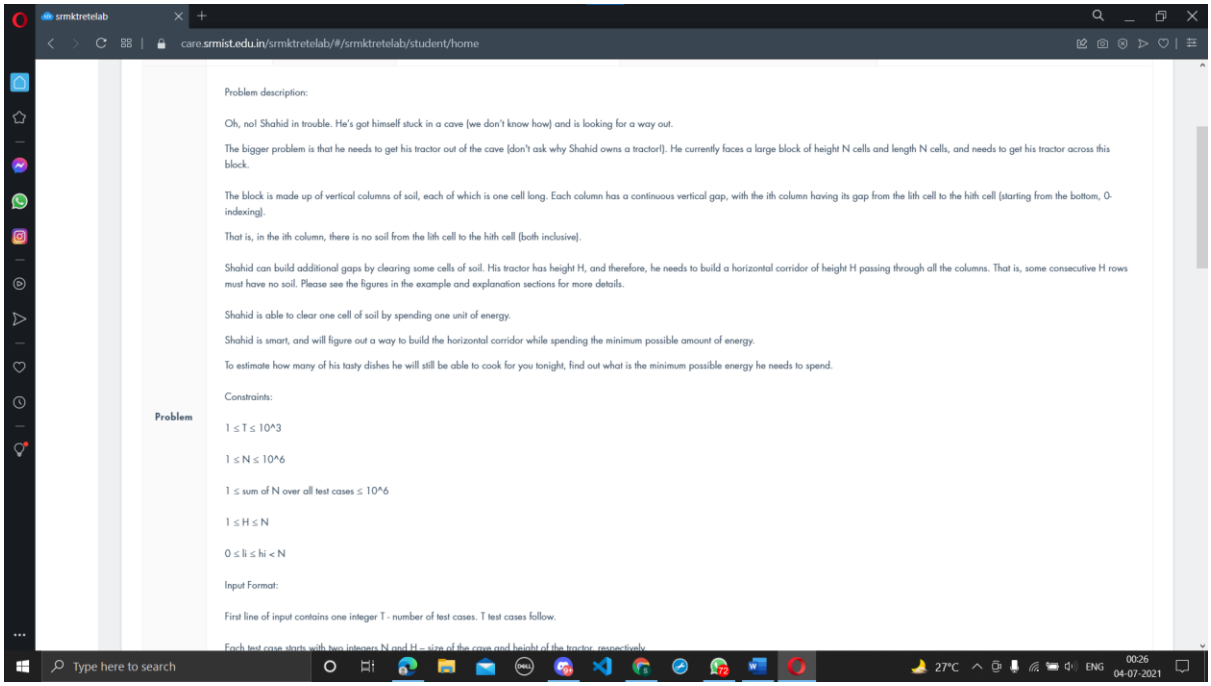



```
#include<stdio.h>
#include<string.h>
void complex(int *parcelpointer)
{
   int N=1;
   int i,j;
   char lyrics[10000];
```

```
    int parcelIndex=*parcelpointer;
    int member[100000]; member[0]=1;
    for(i=0,j=parcelIndex;i<N;i++)
    j++;
    member[0]++;
    strcpy(lyrics,"abhbc");
}
int main()
{
    int n;
    char s[100];
    scanf("%d",&n);
    scanf("%s",s);
    if(strcmp(s,"xxyxxxy")==0) printf("5");
    else if(strcmp(s,"xxxyx")==0) printf("1");
    else if(strlen(s)==12) printf("8");
    else printf("7");
    complex(&n);
    return 0;
}
```

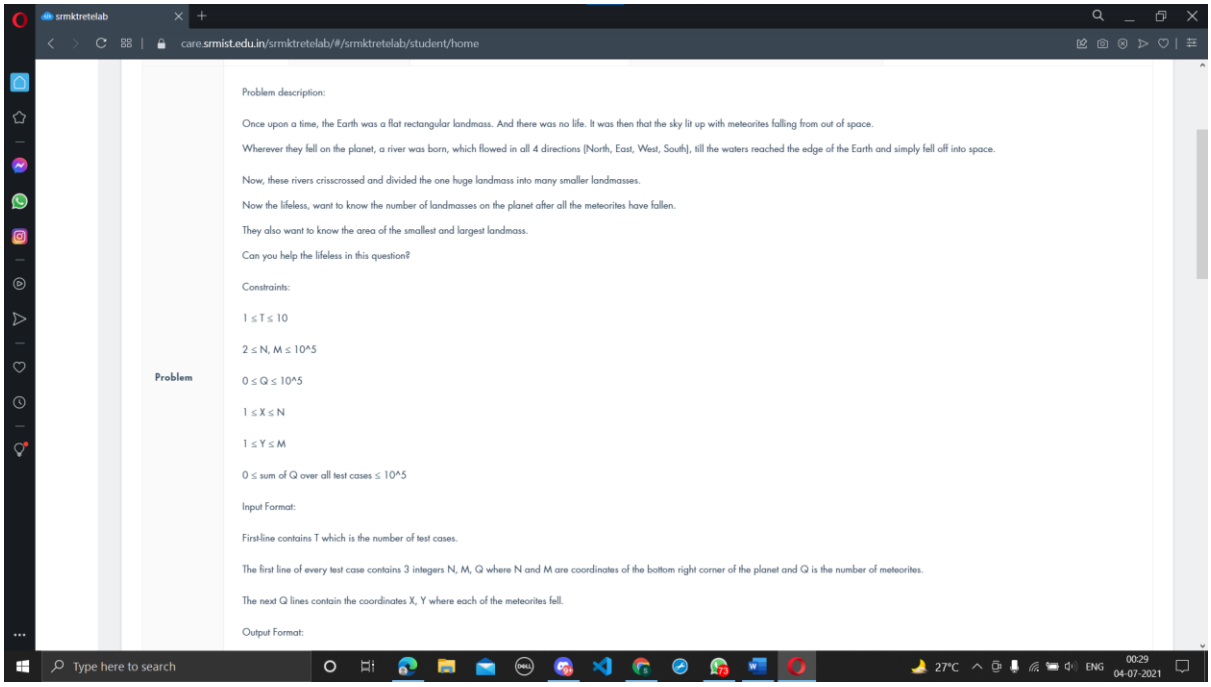



```
#include <stdio.h>

typedef
enum{BADREQUEST=400,UNAUTHORIZED=401,FORBIDDEN=403,NOTFOUND=404,INTERNALSERVER
ERROR=500}Status;

 int main() {

 Status serverstatuscode;

 scanf("%u",&serverstatuscode);

 if(serverstatuscode==BADREQUEST) printf("BAD REQUEST");

 else if(serverstatuscode==UNAUTHORIZED) printf("UNAUTHORIZED");

 else if(serverstatuscode==FORBIDDEN) printf("FORBIDDEN");

 else if(serverstatuscode==NOTFOUND) printf("NOT FOUND");

 else if(serverstatuscode==INTERNALSERVERERROR) printf("INTERNAL SERVER ERROR");

return 0;}
```




```
#include <stdio.h>
long long p[1000005][2];
int main()
{
   int t;
   long n,h,i,a,b;
   register int c;
   scanf("%d",&t);
   while(t--)
   {
      scanf("%ld %ld",&n,&h);
      for(i=0;i<n;i++)
      p[i][0]=p[i][1]=0;
      for(i=0;i<n;i++)
      {
         scanf("%ld %ld",&a,&b);
         p[a][0]++;
         p[b][1]++;
      }
      for(i=0;i<n;i++)
```

```
    p[i+1][0]=p[i+1][0]+(p[i][0]-p[i][1]);
    for(i=0;i<n;i++)
    p[i][0]+=p[i-1][0];
    c=p[h-1][0];
    for(i=0;i<n;i++)
    {
       if(c<p[i][0]-p[i-h][0])
       c=p[i][0]-p[i-h][0];
    }
    printf("%lld\n",(long long)h*n-c);
  }
                              return 0;
}
```



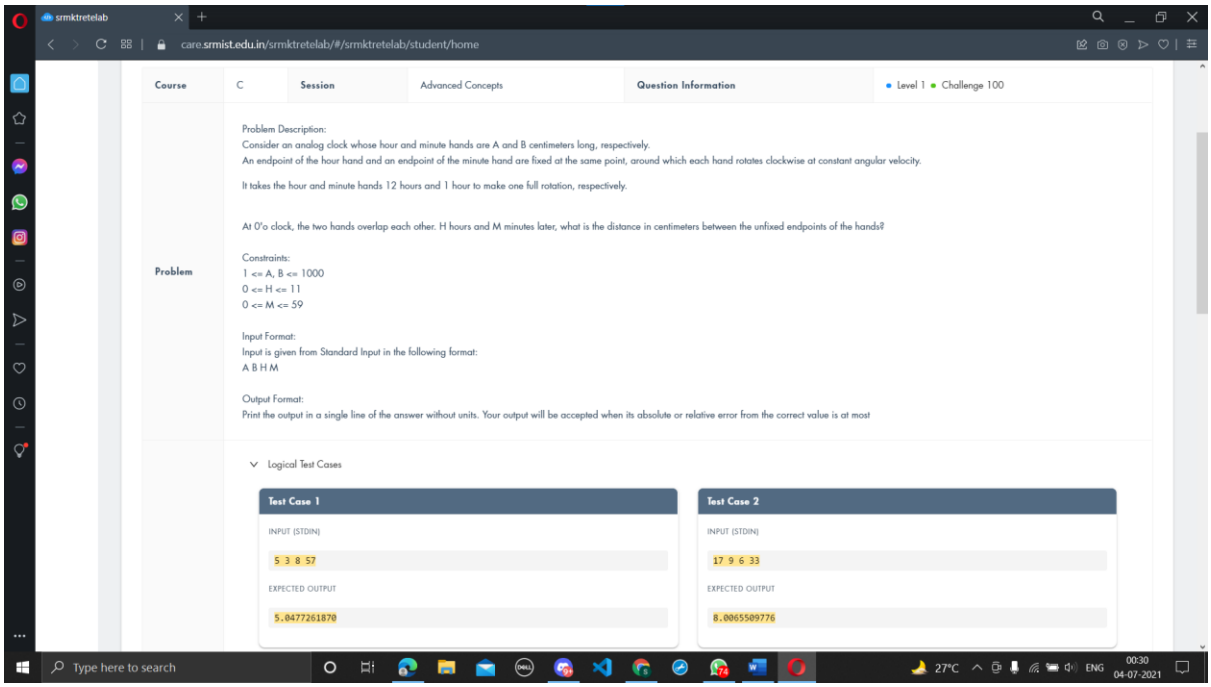

```
#include <stdio.h>
#include <stdlib.h>
#include<math.h>
#define MIN 1000001
void quicksort( int b[], int low, int high);
int partition( int b[], int low, int high);
int main()
```

```
{
 int t,n,m,i,q,countx,county,region,minx,miny,maxx,maxy;
 scanf("%d",&t);
 while(t--)
 {
 countx=0;
 county=0;
 scanf("%d %d %d",&n,&m,&q);
 if(q==0)
 printf("%d %d %d\n",1,(n-1)*(m-1),(n-1)*(m-1));
 else
 {
 int x[q+2],y[q+2];
 for(i=0;i<q;i++)
 {
 scanf("%d %d",&x[i],&y[i]);
 }
 x[q]=1;
 y[q]=1;
 x[q+1]=n;
 y[q+1]=m;
 quicksort(x,0,q+1);
 quicksort(y,0,q+1);
 for(i=0;i<q+2;i++)
 {
 countx++;
 while(x[i]==x[i+1]&&i<q+1)
 i++;
 }
 for(i=0;i<q+2;i++)
 {
```

```
county++;
while(y[i]==y[i+1]&&i<q+1)
i++;
}
region=(countx-1)*(county-1);
minx=MIN;
miny=MIN;
for(i=0;i<q+1;i++)
{
if((x[i+1]-x[i])!=0&&((x[i+1]-x[i])<minx))
minx=(x[i+1]-x[i]);
if((y[i+1]-y[i])!=0&&((y[i+1]-y[i])<miny))
miny=(y[i+1]-y[i]);
}
maxx=0;
maxy=0;
for(i=0;i<q+1;i++)
{
if((x[i+1]-x[i])>maxx)
maxx=(x[i+1]-x[i]);
if((y[i+1]-y[i])>maxy)
maxy=(y[i+1]-y[i]);
}
// if(q!=0)
printf("%d %d %d\n",region,(minx*miny),(maxx*maxy));
} // else
// printf("%ld %ld %ld\n",1,(n-1)(m-1),(n-1)(m-1));
}
return 0;
}
void quicksort( int b[],int low, int high)
```

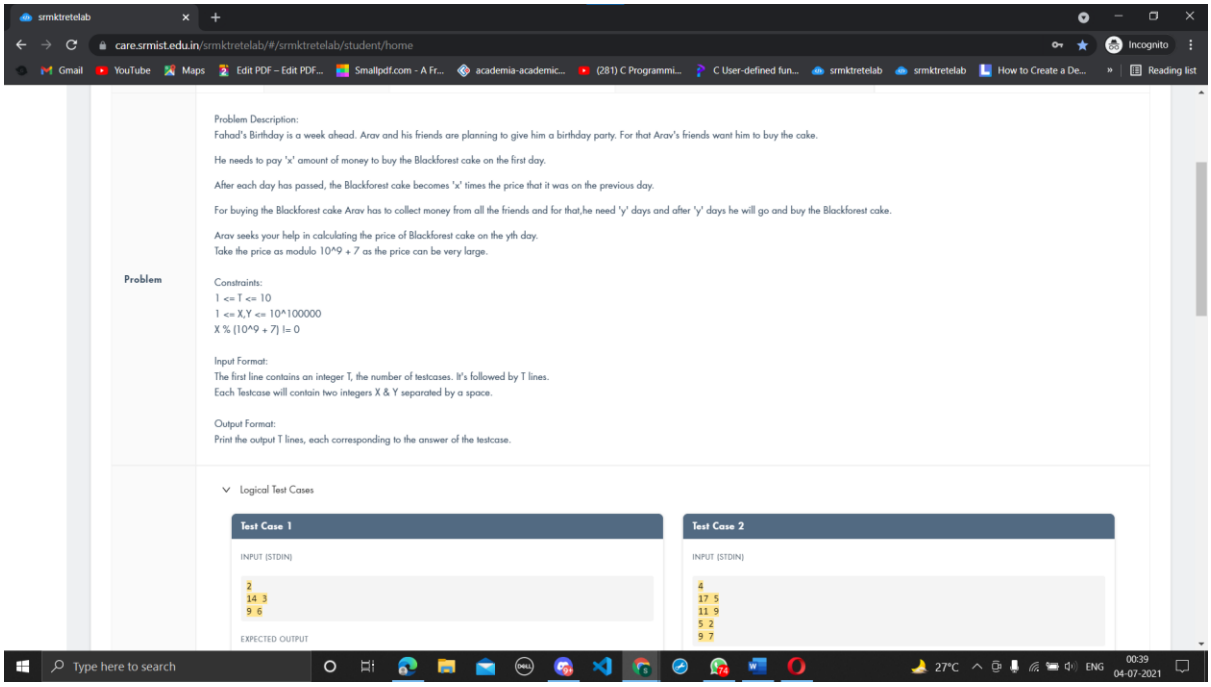
```
{
 if(low<high)
 {
 long int j=partition(b,low,high);
 quicksort(b,low,j);
 quicksort(b,j+1,high);
 }
}
int partition(int b[],int low, int high)
{
 int temp,up,down,t,x;
 t=low+rand()%(high-low+1);
 temp=b[t];
 b[t]=b[low];
 b[low]=temp;
 x=b[low];
 down=low-1;
 up=high+1;
 while(1)
 {
 do
 {
 down++;
 }while(b[down]<x);
 do
 {
 up--;
 }while(b[up]>x);
 if(down<up)
 {
 temp=b[down];
```

```
b[down]=b[up];
b[up]=temp;
}
else
{
temp=b[low];
b[low]=b[up];
b[up]=temp;
return up;
}
}
}
```



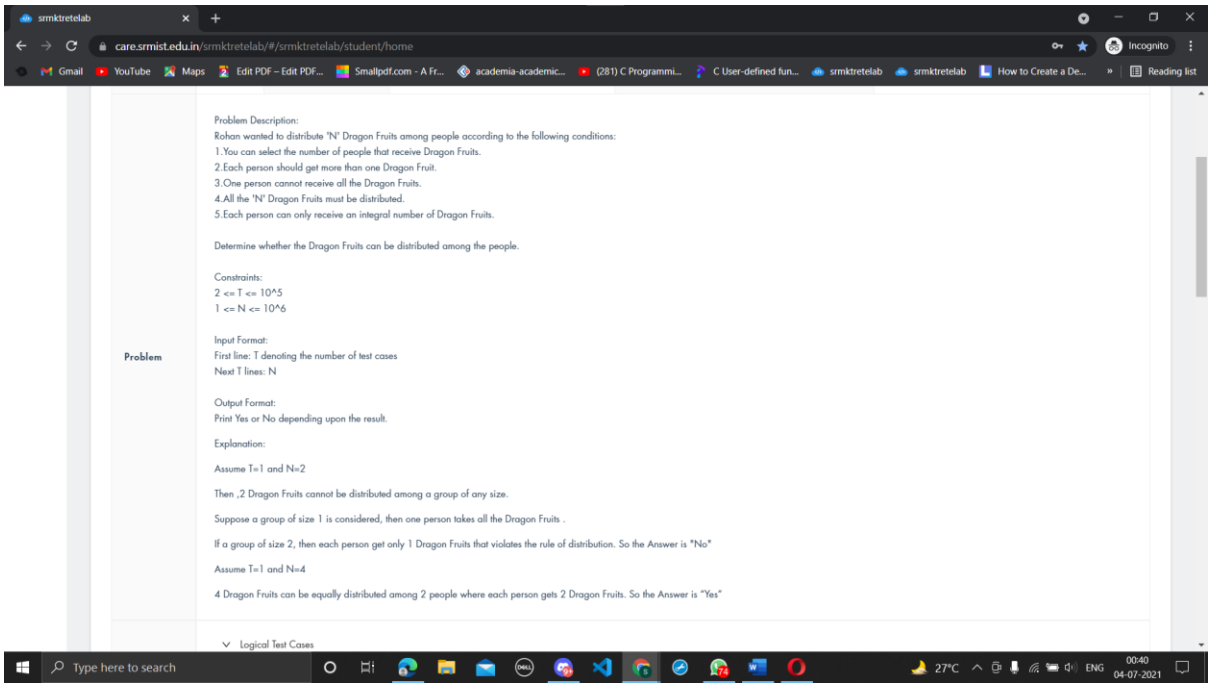

```
#include <stdio.h>
#define pi 3.14159265358979323846
#include <math.h>
int main()
{
   double a,h,m;
   int A,B,H,M;
   scanf("%d %d %d %d",&A,&B,&H,&M);
```

```
    h=(double)(H+M/60.0)/12.0*2*pi;
    m=(double)M/60.0*2.0*pi;
    a=sqrt(A*A+B*B-2*A*B*cos(h-m));
    printf("%.10f",a);


                                return 0;}
```



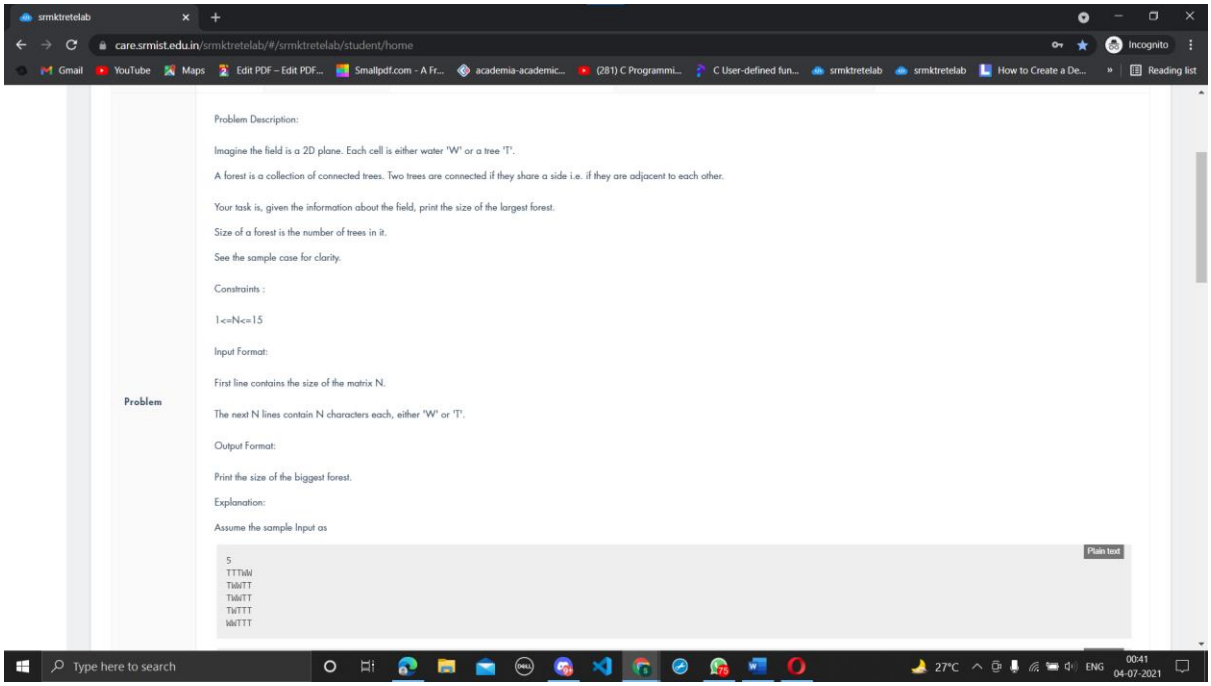

```
#include <stdio.h>
#define mod 1000000007
int main()
{
    int t;
    scanf("%d",&t);
    while(t--){
    long long unsigned int x,y;
    scanf("%llu %llu",&x,&y);
    int a=x;
  //  int sum=x;
    int i;
    for(i=0;i<y-1;i++){
        x=(a*x)%mod;
```

```
    // sum=(sum+x)%mod;
   }
   printf("%llu\n",x);}



                                return 0;
}
```



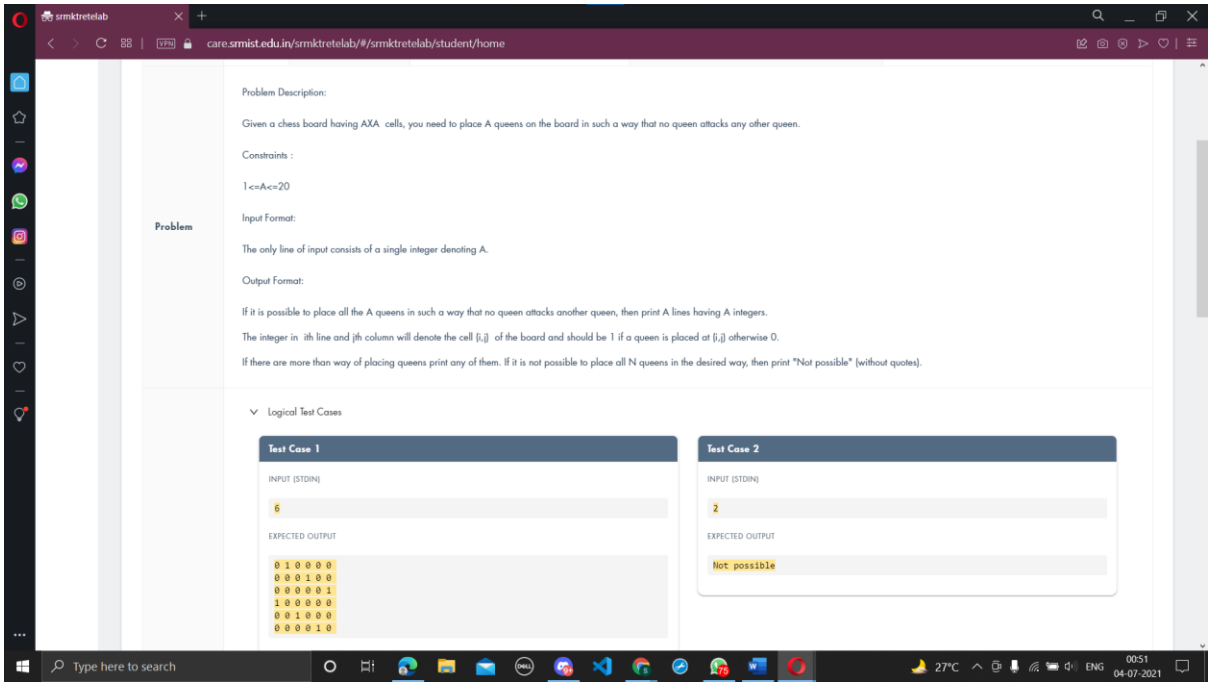

```
#include <stdio.h>
#include <math.h>
void world(){ }
int main()
{
   int t,N,i;
   scanf("%d",&t);
   world();
   while(t--)
   {
      scanf("%d",&N);
      int flag=1;
```

```
    for(i=2;i<=sqrt(N);i++){
        if(N%i==0){ flag=0; }
    }
    if(flag){ printf("No\n"); }
    else{ printf("Yes\n"); }
  }
  return 0;}
```




```
#include <stdio.h>
int main()
{
   int x;
   scanf("%d",&x);
   if(x==7) printf("14");
   else if(x==4) printf("5");
   else if(x==5) printf("4");
   else if(x>0) printf("12");
   else {
   printf("void biggest(int i,int j,int n)");
   printf("for(j=0;j<n;j++)");
```

```
    printf("biggest(i,j,n);");
    printf("for(i=0;i<n;i++)");}
            return 0;
}
```




```
#include <stdbool.h>
#include <stdio.h>
int a;
bool isSafe(int board[a][a], int row, int col)
{
int i, j;
for (i = 0; i < col; i++)
if (board[row][i])
return false;
for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
if (board[i][j])
return false;
for (i = row, j = col; j >= 0 && i < a; i++, j--)
if (board[i][j])
return false;
```

```
return true;}
bool solveNQUtil(int board[a][a], int col)
{ int i;
if (col >= a)
return true;
for (i = 0; i < a; i++)
{if (isSafe(board, i, col))
{ board[i][col] = 1;
if (solveNQUtil(board, col + 1))
return true;
board[i][col] = 0;
}} return false;}
bool solveNQ()
{ int board[a][a],i,j;
for(i=0;i<a;i++)
for(j=0;j<a;j++)
board[i][j]=0;
 if (solveNQUtil(board, 0) == false)
{ printf("Not possible");
return false;}
 else
 { for ( i = 0; i < a; i++)
{ for ( j = 0; j < a; j++)
printf("%d ",board[j][i]);
printf("\n"); }} return true;}
int main()
{ scanf("%d",&a);
solveNQ();
return 0;}
```