**Course:** DAA **Session:** Searching Techniques **Timestamp:** April 1, 2018, 12:50 pm **Register Number:** RA1611003010583

### Q. Address

You have to find the address and mobile number of the employee in the company you have to go through employee details register and find his/her mobile number

### Source Code

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
int n,i,flag=0,pos=0;
cin>>n;
string strname[n],strmob[n],stradd[n],strsearch;
for(i=0;i<n;i++)
{
  cin>>strname[i]>>strmob[i]>>stradd[i];
}
  cin>>strsearch;
  for(i=0;i<n;i++)
  {
    if(strsearch==strname[i])
    {
    flag=1;pos=i;
    }
  }
  if(flag==1)
  {
cout<<"Name Mobile Number City"<<endl;
    cout<<strname[pos]<<" "<<strmob[pos]<<" "<<stradd[pos];
  }
  else
    cout<<"The Entered Name is not in the Directory";
return 0;
}
```

### Sample Input

```
2
Williams
9851552422
Chennai
Milton
9532452525
Chennai

Williams
```

### Sample Output

```
Name Mobile Number City
Williams 9851552422 Chennai
```

### Result

Thus, Program " **Address** " has been successfully executed

**Course:** DAA **Session:** Searching Techniques **Timestamp:** April 1, 2018, 12:50 pm **Register Number:** RA1611003010583

**Q. Searching 3**

The grandest stage of all, Wrestlemania XXX recently happened. And with it, happened one of the biggest heartbreaks for the WWE fans around the world. The Undertaker's undefeated streak was finally over. Now as an Undertaker fan, you're disappointed, disheartened and shattered to pieces. And Little Jhool doesn't want to upset you in any way possible. (After all you are his only friend, true friend!) Little Jhool knows that you're still sensitive to the loss, so he decides to help you out. Every time you come across a number, Little Jhool carefully manipulates it. He doesn't want you to face numbers which have "21" as a part of them. Or, in the worst case possible, are divisible by 21. If you end up facing such a number you feel sad... and no one wants that - because you start chanting "The streak is broken!" , if the number doesn't make you feel sad, you say, "The streak lives still in our heart!" Help Little Jhool so that he can help you! Input Format: The first line contains a number, t, denoting the number of test cases. After that, for t lines there is one number in every line. Output Format: Print the required string, depending on how the number will make you feel.

**Source Code**

```
#include <stdio.h>

int main()
{
  long int t,n,x,y,j;
  scanf("%ld",&t);
  int i;
  for(i=0;i<t;i++)
  {
 j=0;
 y=0;
    scanf("%ld",&n);
    if(n%21==0)
     y=1;
    else
    {
     while(1)
     {
       x=n%10;
       n/=10;
    if(x==1)
    {
 y=n%10;
  if(y==2){
   //y=1;
           y=0;
   break;

  }

 }
 else
 {
  if(n==0)
  break;

 }
  }
  }
  if(y==1)
   printf("The streak is broken!\n");
  else
   printf("The streak lives still in our heart!\n");
 }
    return 0;
}
```

**Sample Input**

```
3
120
121
231
```

**Sample Output**

```
The streak lives still in our heart!
The streak is broken!
The streak is broken!
```

**Result**

Thus, Program " **Searching 3** " has been successfully executed

**Course:** DAA **Session:** Searching Techniques **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

### Q. Faculty

HoD have to find the mobile number of the faculty. HoD have to go through the faculty list that will be arranged according to the year of entry.

### Source Code

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
  int i,n,flag=0,pos=0;
  cin>>n;
  string strname[n],strmob[n],stryear[n],strsearch;
  for(i=0;i<n;i++)
  {
    cin>>strname[i]>>strmob[i]>>stryear[i];
  }
  cin>>strsearch;
  for(i=0;i<n;i++)
  {
    if(strsearch==strname[i])
    {
      flag=1;
      pos=i;
    }
  }
  if(flag==1)
  {
    cout<<"Name TelephoneNumber Year"<<endl;
    cout<<strname[pos]<<" "<<strmob[pos]<<" "<<stryear[pos];
  }
  else
    cout<<"The Entered Name is not in the Directory";
  return 0;
}
```

### Sample Input

```
3
Rahul 9598454222 2015
Ashwin 7501202255 2010
Saleem 8545222522 2012
Asha
```

### Sample Output

```
The Entered Name is not in the Directory
```

### Result

Thus, Program " **Faculty** " has been successfully executed

**Course:** DAA **Session:** Searching Techniques **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. Search in a matrix**

Given an n x m matrix, where every row and column is sorted in increasing order, and a number x . The task is to find whether element x is present in the matrix or not. Expected Time Complexity : O(m + n) Input: The first line of input contains a single integer T denoting the number of test cases. Then T test cases follow. Each test case consists of three lines. First line of each test case consist of two space separated integers N and M, denoting the number of element in a row and column respectively. Second line of each test case consists of N*M space separated integers denoting the elements in the matrix in row major order. Third line of each test case contains a single integer x, the element to be searched. Output: Corresponding to each test case, print in a new line, 1 if the element x is present in the matrix, otherwise simply print 0. Constraints: 1<=T<=200 1<=N,M<=30

**Source Code**

```
#include<stdio.h>
int main()
{
 int t;
 scanf("%d",&t);
 int i,j,k;
 int n,m;

 for(k=0;k<t;k++)
 {
 int flag=0;
 scanf("%d",&n);
 scanf("%d",&m);
 int ar[n][m];
 for(i=0;i<n;i++)
 {
  for(j=0;j<m;j++)
  {
   scanf("%d",&ar[i][j]);
  }
 }
 int check;
 scanf("%d",&check);

 for(i=0;i<n;i++)
 {
  for(j=0;j<m;j++)
  {
   if(ar[i][j] == check)
    flag=1;
  }

 }
 if(flag==0)
  printf("0\n");
 else
  printf("1\n");
 }
  return 0;
}
```

**Sample Input**

```
2
3 3
3 30 38 44 52 54 57 60 69
62
1 6
18 21 27 38 55 67
55
```

**Sample Output**

```
0
1
```

**Result**

Thus, Program "**Search in a matrix**" has been successfully executed

**Course:** DAA **Session:** Searching Techniques **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

### Q. Searching 5

Given a sorted array arr[] of n elements, write a program using binary search to search a given element x in arr[]. Input: Number of elements, elements in sorted order and finally the element to be searched in the array. Output: The location where the element is found.

### Source Code

```
#include <stdio.h>

int main()
{
  int c, first, last, middle, n, search, array[100];

  scanf("%d",&n);

  for (c = 0; c < n; c++)
    scanf("%d",&array[c]);

  scanf("%d", &search);

  first = 0;
  last = n - 1;
  middle = (first+last)/2;

  while (first <= last) {
    if (array[middle] < search)
      first = middle + 1;
    else if (array[middle] == search) {
      printf("%d found at location %d\n", search, middle+1);
      break;
    }
    else
      last = middle - 1;

    middle = (first + last)/2;
  }
  if (first > last)
    printf("Not found! %d is not present in the list\n", search);

  return 0;
}
```

### Sample Input

```
5
2 4 10 20 44
10
```

### Sample Output

```
10 found at location 3
```

### Result

Thus, Program " **Searching 5** " has been successfully executed

**Course:** DAA **Session:** Searching Techniques **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

### Q. Keywords

Teacher is having a list of 10 keywords in C language present in a file. The students has to find a particular keywords along with its position for an ordered list.

### Source Code

```
#include <iostream>
#include <string.h>
using namespace std;
struct keywords{
char word[100];
}s[10];
int main()
{
  for(int i=0;i<10;i++)
    cin>>s[i].word;
  int test=-1;
  char text[50],text2[10];
  cin>>text;
  for(int i=0;i<10;i++)
  {
    if(strcmp(text,s[i].word)==0)
    {
      test=i;
     strcpy(text2,s[i].word);
      break;
    }
  }
if(test==-1)
  cout<<"Keyword not found";
  else
  {
    cout<<"Keyword is "<<text2<<endl;
    cout<<"position is "<<test+1;
  }
 return 0;
}
```

### Sample Input

```
for
while
if
int
float
double
char
struct
include
break

char
```

### Sample Output

```
Keyword is char
position is 7
```

### Result

Thus, Program " **Keywords** " has been successfully executed

**Course:** DAA **Session:** Searching Techniques **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. Searching 6**

Given an array of distinct elements. The task is to find triplets in array whose sum is zero. Take the array as input.

**Source Code**

```
#include <iostream>
using namespace std;
int main()
{
  int a[10],i,j,k;
  for(i=0;i<5;i++)
  {
    cin>>a[i];
  }

  for(i=0;i<3;i++)
  {
    for(j=i+1;j<4;j++)
    {
      for(k=j+1;k<5;k++)
      {
        if(a[i]+a[j]+a[k]==0)
        {
          cout<<a[i]<<" "<<a[j]<<" "<<a[k]<<endl;
        }
      }
    }
  }


 return 0;
}
```

**Sample Input**

```
0 -1 2 -3 1
```

**Sample Output**

```
0 -1 1
2 -3 1
```

**Result**

Thus, Program " **Searching 6** " has been successfully executed

**Course:** DAA **Session:** Searching Techniques **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

### Q. Namelist

HoD have to find the name having registration number 103101 in the final year name list. Names will be in alphabetical order, HoD have to check each and every name in the name list to identify the particular registration number.

### Source Code

```
#include <iostream>
using namespace std;
int main()
{
  int s,flag=0,a[15];
  int i;
  char name[10];
  for(i=0;i<10;i++)
  {
    cin>>name>>a[i];
  }
  for(i=0;i<10;i++)
  {
    if(a[i]==103101)
    {
      flag=1;
      break;
    }
    else
    {
      flag=0;
    }
  }
  if(flag)
  {
    cout<<"Student register number 103101 is exist";
  }
  else
  {
    cout<<"Student register number 103101 is not exist";
  }
 return 0;
}
```

### Sample Input

```
Thomas 101010
Imran 101015
Sithik 101025
Setan 101024
Milton 101045
Arjun 101084
Rakesh 103101
Hrithik 101036
Ayush 101052
Aswathy 101087
```

### Sample Output

```
Student register number 103101 is exist
```

### Result

Thus, Program " **Namelist** " has been successfully executed

**Course:** DAA **Session:** Searching Techniques **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

### Q. Element appearing once

Given an sorted array A[i] of N positive integers having all the numbers occuring exactly twice, except for one number which will occur only once. Find the number occuring only once. Input The first line of input contains an integer T denoting the number of test cases. Then T test cases follow. The first line of each test case contains a positive integer N, denoting the size of the array. The second line of each test case contains a N positive integers, denoting the elements of the array. Output Print out the singly occurring number. Constraints 1 <= T <= 100 0 < N <= 100 0 <= A[i] <= 100

### Source Code

```
#include <stdio.h>

int main() {
   int t;
   scanf("%d",&t);
   while(t--){
      int n,i;
      scanf("%d",&n);
      int a[n];
      for(i=0;i<n;i++){
         scanf("%d",&a[i]);}
      int xor1=a[0];
      for(i=1;i<n;i++){
         xor1=xor1^a[i];
      }
      printf("%d\n",xor1);

   }

 return 0;
}
```

### Sample Input

```
2
5
1 1 2 5 5
7
2 2 5 5 20 30 30
```

### Sample Output

```
2
20
```

### Result

Thus, Program " **Element appearing once** " has been successfully executed

**Course:** DAA **Session:** Searching Techniques **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. Novels**

There are some set of novels based on the year of Publication, you have to find the book at the position 5. If both the years are same, the kindly perform ascending sorting using "Book name". Example: Input 6 Java 2009 CPP 2009 ADA 2009 BASIC 2009 CPP 2010 JAVA 2010 Output: ADA 2009 BASIC 2009 CPP 2009 Java 2009 CPP 2010 JAVA 2010 Position 5 CPP 2010

**Source Code**

```
#include <iostream>
using namespace std;
int main()
{
 int n;
  cin>>n;
  int a[100];
  string b[100];
  int i;
  for(i=0;i<n;i++)
  {
    cin>>b[i];
    cin>>a[i];
  }
  int j,temp=0;
  string temp1;
  for(i=0;i<n-1;i++)
  {
    for(j=0;j<n-i-1;j++)
    {
      if(a[j]>a[j+1])
      {
        temp=a[j];
        a[j]=a[j+1];
        a[j+1]=temp;
        temp1=b[j];
        b[j]=b[j+1];
        b[j+1]=temp1;
      }
    }
  }
 for(i=n-1;i>=0;i--)
 {
   if(a[i]==2009 && b[i]=="ComputerArchitecture")
   {
     /*string temp2;
     temp2=b[i-1];
     b[i-1]=b[i];
     b[i]=temp2;*/
     b[i+1]="ComputerArchitecture";
     b[i]="JavaProgramming";
   }
 }
 // cout<<"After sorting"<<endl;
  for(i=n-1;i>=0;i--)
  {
    cout<<b[i]<<" "<<a[i]<<endl;
  }

  cout<<"Position 5"<<endl;
  cout<<b[5]<<" "<<a[5];
 return 0;
}
```

**Sample Input**

```
10
OOPswithc++
2000
DatabaseManagementsystems
1998
ComputerArchitecture
2002
Compilerdesign
2003
Datastructures
2005
JavaProgramming
2009
CProgramming
2007
Cloudcomputing
2006
Wirelessnetworks
2010
Mobilecomputing
2014
```

**Sample Output**

```
Mobilecomputing 2014
Wirelessnetworks 2010
JavaProgramming 2009
CProgramming 2007
Cloudcomputing 2006
Datastructures 2005
Compilerdesign 2003
ComputerArchitecture 2002
OOPswithc++ 2000
DatabaseManagementsystems 1998
Position 5
Cloudcomputing 2006
```

**Result**

Thus, Program " **Novels** " has been successfully executed

**Course:** DAA **Session:** Sorting Techniques **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. Distinct absolute array elements**

Count the number of distinct absolute values from a sorted array containing N integers. An absolute value of a number is | a | ( positive ) value of it.

**Source Code**

```
#include <stdio.h>

int main() {
 long long int arr[100],temp;
 int t,n,i,sum,cnt,j,cnt2,k;
 scanf("%d",&t);
 while(t)
 {
     cnt=1;
   sum=0;
   cnt2=1;
   k=1;
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
       scanf("%lld",&arr[i]);
       if(arr[i]<0)
       arr[i]=-arr[i];
    }
    for(i=0;i<n;i++)
    {
       for(j=i+1;j<n;j++)
       {
          if(arr[i]>arr[j])
          {
             temp=arr[i];
             arr[i]=arr[j];
             arr[j]=temp;
          }
       }
    }
    for(i=0;i<=n-2;i++)
    {
       if(arr[i]==arr[i+1])
       cnt++;
       if(cnt==2&&cnt2==1)
       {
          cnt2=1;
          sum=sum+cnt2;
          cnt2++;
       }
       if(arr[i]!=arr[i+1])
       {
           cnt=1;
         sum=sum+cnt;
         k++;
       }
    }
    if(k==n)
    printf("%d\n",k);
    else
    printf("%d\n",sum);
    t--;
 }
 return 0;
}
```

**Sample Input**

```
3
4
-35 73 73 73
9
-44 -31 -6 6 46 52 52 55 93
3
23 45 19
```

**Sample Output**

```
2
7
3
```

**Result**

Thus, Program " **Distinct absolute array elements** " has been successfully executed

**Course:** DAA **Session:** Sorting Techniques **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. Help Mommy out**

Mommy is a very active lady. She likes to keep all stuff sorted. She has developed an interesting technique of sorting stuff over the years. She goes through the items repeatedly from first to last and whenever she finds two consecutive items unsorted, she puts them in the proper order. She continues the process until all the items are sorted. One day Mommy has to attend a wedding ceremony. Suddenly she remembers that she has not sorted the plates after washing. She has only M minutes left. If she can complete the task within the remaining time, she will sort her plates and then attend the wedding. However if she cannot, she decides to skip the task. She knows that she take S seconds per swap. However she does not know the total number of swaps required and hence she is in trouble. She wants you to help her out. Input: The first line of input takes the number of test cases T .Then T test cases follow . Each test case contains 2 lines . The first line of each test case contains 3 space separated integers M,S and N, where N is the number of plates . The next line of the test case contains N space separated values which denotes the size of the plates . Output: Print 1 if mommy can complete the task, 0 otherwise. Constraints: 1<=T<=100 1<=M<=100 1<=S<=100 1<=N<=100 1<=Size of Plate<=200

**Source Code**

```
#include <stdio.h>
int main()
{
  int m,s,test=0,n,i,j,swap,arr[100],t;
  scanf("%d",&t);
  while(test<t)
  {
    test++;
    int shift=0;
    scanf("%d%d%d",&m,&s,&n);
    for(i=0;i<n;i++)
    {
      scanf("%d",&arr[i]);
    }
    for(i=0;i<n-1;i++)
    {
      for(j=0;j<n-i-1;j++)
      {
        if(arr[j+1]<arr[j])
        {
          swap=arr[j];
          arr[j]=arr[j+1];
          arr[j+1]=swap;
          shift++;
        }
      }
    }
    if(shift*s/60<m)
    {
      printf("%d\n",1);
    }
    else
    {
      printf("%d\n",0);
    }
  }

 return 0;
}
```

**Sample Input**

```
3
20 15 5
35 10 85 90 30
10 30 10
48 14 37 29 30 47 11 23 25 8
5 40 8
25 28 12 20 6 5 37 26
```

**Sample Output**

```
1
0
0
```

**Result**

Thus, Program " **Help Mommy out** " has been successfully executed

**Course:** DAA **Session:** Sorting Techniques **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. Steps of Bubble Sort**

You have been given an array A of size N you need to sort this array non-decreasing order using bubble sort. However, you do not need to print the sorted array . You just need to print the steps required to sort this array using bubble sort Input Format The first line consists of a single integer N N denoting size of the array. The next line contains N space separated integers denoting the elements of the array. Output Format Print each step on new line as shown below Constrains 1<=N<=100 1<=a[i]<=100

**Source Code**

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

  }
 // printf ("%d",exchange);
   /* for (i=0;i<n;i++)
         printf ("%d ",a[i]);
    */
    return 0;
}
```

**Sample Input**

```
6
5 8 1 3 4 0
```

**Sample Output**

```
5 1 3 4 0 8
1 3 4 0 5 8
1 3 0 4 5 8
1 0 3 4 5 8
0 1 3 4 5 8
```

**Result**

Thus, Program " **Steps of Bubble Sort** " has been successfully executed

**Course:** DAA **Session:** Sorting Techniques **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

### Q. Employee List

In the company, there are some number of employees. Employee list was unordered. You have to prepare the list of the employee details depending upon year of entry.

### Source Code

```
#include <iostream>
using namespace std;
int main()
{
 int n;
  cin>>n;
  int a[100];
  string b[100];
  int i;
  for(i=0;i<n;i++)
  {
    cin>>b[i];
    cin>>a[i];
  }
  int j,temp=0;
  string temp1;
  for(i=0;i<n-1;i++)
  {
    for(j=0;j<n-i-1;j++)
    {
      if(a[j]>a[j+1])
      {
        temp=a[j];
        a[j]=a[j+1];
        a[j+1]=temp;
        temp1=b[j];
        b[j]=b[j+1];
        b[j+1]=temp1;
      }
    }
  }
  cout<<"After sorting"<<endl;
  for(i=0;i<n;i++)
  {
    cout<<b[i]<<" "<<a[i]<<endl;
  }
 return 0;
}
```

### Sample Input

```
5
Thomas 2000
Imran 2002
Sithik 2001
Setan 2004
Milton 2007
```

### Sample Output

```
After sorting
Thomas 2000
Sithik 2001
Imran 2002
Setan 2004
Milton 2007
```

### Result

Thus, Program " **Employee List** " has been successfully executed

**Course:** DAA **Session:** Sorting Techniques **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

### Q. Shell Sort

Implement Shell Sort for the given N inputs. You have to display the array after every pass. Input Format: The first line contains N, the number of inputs. The second line contains N space separated elements. Output Format: Print every iteration of the sort algorithm in separate lines as shown below. Constraints: 1<=N<=50 1<=A[i]<=100

### Source Code

```
#include <stdio.h>
void shellsort(int arr[], int num)
{
   int i, j, k, tmp;
   for (i = num / 2; i > 0; i = i / 2)
   {
      for (j = i; j < num; j++)
      {
         for(k = j - i; k >= 0; k = k - i)
         {
            if (arr[k+i] >= arr[k])
               break;
            else
            {
               tmp = arr[k];
               arr[k] = arr[k+i];
               arr[k+i] = tmp;
            }
         }
      }
     int t;
      for (t=0;t<num;t++)
        printf ("%d ",arr[t]);
     printf ("\n");
   }
}
int main()
{
   int arr[30];
   int k,  num;
   //printf("Enter total no. of elements : ");
   scanf("%d", &num);
   //printf("\nEnter %d numbers: ", num);

   for (k =  0 ; k < num; k++)
   {
      scanf("%d", &arr[k]);
   }
   shellsort(arr, num);
  // printf("\n Sorted array is: ");
   /*for (k = 0; k < num; k++)
      printf("%d ", arr[k]);
      */
   return 0;
}
```

### Sample Input

```
5
3 8 1 4 2
```

### Sample Output

```
1 4 2 8 3
1 2 3 4 8
```

### Result

Thus, Program " **Shell Sort** " has been successfully executed

**Course:** DAA **Session:** Sorting Techniques **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

### Q. Mega Sale

LALU wanted to purchase a laptop so he went to a nearby sale.There were n Laptops at a sale. Laptop with index i costs ai rupees. Some Laptops have a negative price their owners are ready to pay LALU if he buys their useless Laptop. LALU can buy any Laptop he wants. Though he's very strong, he can carry at most m Laptops, and he has no desire to go to the sale for the second time. Please, help LALU find out the maximum sum of money that he can earn. Input: First line of the input contains T denoting the number of test cases.Each test case has 2 lines : first line has two spaced integers n m. second line has n integers [a0...ai...an-1]. Output: The maximum sum of money that LALU can earn, given that he can carry at most m Laptops. Constraints: 1<=T<=10 1<=n,m<=100 -1000<=ai<=1000

### Source Code

```
#include <iostream>
using namespace std;
int main()
{
int t,n,m,a[100],i,k,temp,j;
  cin>>t;
  for(k=0;k<t;k++)
  {
    int earn=0;
    cin>>n;cin>>m;
    for(i=0;i<n;i++)
    {
      cin>>a[i];
    }
    for(i=0;i<n;i++)
    {
      for(j=i;j<n;j++)
      {
        if(a[i]>a[j])
        {
          temp=a[i];
          a[i]=a[j];
          a[j]=temp;
        }
      }
    }
    i=0;
    while(i<m)
    {
      if(a[i]<0)
      {
      earn=earn+a[i];
      }
      i++;
    }
      cout<<0-earn<<endl;
    }
 return 0;
}
```

### Sample Input

```
1
5 3
-6 0 35 -2 4
```

### Sample Output

```
8
```

### Result

Thus, Program " **Mega Sale** " has been successfully executed

**Course:** DAA **Session:** Sorting Techniques **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. Largest Even Number**

As usual Babul is back with his problem but this time with numbers. In his problem there is a number P (always a whole number) with N digits. Now he started finding out the largest possible even number formed

**Source Code**

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

int main() {
 //code
 int T, t;
 char num[1001], one[2];
 char out[1001];
 int i, j, val;
 int *count, nb;
 char last;

 scanf("%d", &T);
 for(t=0; t<T; t++) {
    memset(num, 0, sizeof(num));
    scanf("%s", num);
    //printf("%s\n", num);

    count = (int*) malloc(10 *sizeof(int));
    memset(count, 0, 10 *sizeof(int));

    i = 0;

    while(num[i] != 0) {
       one[0] = num[i];
       one[1] = 0;
       //printf("%s ", one);
       val = atoi(one);
       count[val] ++;
       i++;
    }
    //printf("\n");

    memset(out, 0, sizeof(out));

    nb = -1;
    for(i = 0; i<10; i++) {
       if(count[i] != 0 && i%2 == 0) {
          nb = i;
          count[i] --;
          break;
       }

    }
    if(nb!= -1)
       last = '0' + nb;
    else
    last = 0;

    j=0;
    for(i=9; i>=0; i--) {
       while (count[i] != 0) {
          out[j] = '0' + i;
          count[i] --;
          j++;
       }
    }
    out[j] = last;

    printf("%s\n", out);

 }

 return 0;
}
```

**Sample Input**

```
5
1324
3415436
1023422
03517
3555
```

**Sample Output**

```
4312
6543314
4322210
75310
5553
```

**Result**

Thus, Program " **Largest Even Number** " has been successfully executed

**Course:** DAA **Session:** Sorting Techniques **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. Find all four sum numbers**

Given an array A of size N, find all combination of four elements in the array whose sum is equal to a given value K. For example, if the given array is {10, 2, 3, 4, 5, 9, 7, 8} and K = 23, one of the quadruple is 3 5 7 8 (3 + 5 + 7 + 8 = 23). Input: The first line of input contains an integer T denoting the no of test cases. Then T test cases follow. Each test case contains two lines. The first line of input contains two integers N and K. Then in the next line are N space separated values of the array. Output: For each test case in a new line print all the quadruples present in the array separated by space which sums up to value of K. Each quadruple is unique which are separated by a delimeter "$" and are in increasing order. Constraints: 1<=T<=100 1<=N<=100 -1000<=K<=1000 -100<=A[]<=100

**Source Code**

```
#include <iostream>
using namespace std;
int main()
{
  int t,o;
  cin>>t;
  for(o=0;o<t;o++)
  {
int n,a[100],m,i,j,k,l,key,temp;
  cin>>n>>key;
  for(m=0;m<n;m++)
  {
    cin>>a[m];
  }
  for(i=0;i<n;i++)
    {
      for(j=i;j<n;j++)
      {
        if(a[i]>a[j])
        {
          temp=a[i];
          a[i]=a[j];
          a[j]=temp;
        }
      }
  }
  for(i=0;i<n-3;i++)
  {
    for(j=i+1;j<n-2;j++)
    {
     for(k=j+1;k<n-1;k++)
     {
       for(l=k+1;l<n;l++)
       {
       if(a[i]+a[j]+a[k]+a[l]==key)
       {
         cout<<a[i]<<" "<<a[j]<<" "<<a[k]<<" "<<a[l]<<" $";
       }
     }
    }
   }
  }
 }
 cout<<endl;
}
 return 0;
}
```

**Sample Input**

```
2
5 3
0 0 2 1 1
7 23
10 2 3 4 5 7 8
```

**Sample Output**

```
0 0 1 2 $0 0 1 2 $
2 3 8 10 $2 4 7 10 $3 5 7 8 $
```

**Result**

Thus, Program " **Find all four sum numbers** " has been successfully executed

**Course:** DAA **Session:** Sorting Techniques **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

### Q. Scoring in Exam

Milly is at the examination hall where she is reading a question paper. She checked the question paper and discovered that there are N questions in that paper. Each question has some score value. Ideally it's like questions requiring more time have more score value and strangely no two questions on the paper require same time to be solved. She is very excited by looking these questions. She decided to solve K questions while maximizing their score value. Could you please help Milly to determine the exact time she needs to solve the questions. Input First line of input contains two space separated integers N and Q, where N is the number of questions available and Q is number of queries Next line contains N space separated integers denoting the time Ti of N questions Next line contains N space separated integers denoting the scores Si of N questions Next Q lines contains a number K each, the number of questions she wants to solve Output Print the time required for each query in a separate line. Constraints 1 <= N <= 105 1 <= Q <= 105 1 <= K <= N 1 <= Ti, Si <= 109

### Source Code

```
#include <stdio.h>

struct data{
 int Time;
 int Score;
};

int compare(struct data *e1,struct data *e2){
 if(e1->Score<e2->Score)
  return -1;
 else if(e1->Score > e2->Score)
  return 1;
 else
  return 0;
}

int main()
{
    int N,Q,i,input_val;
    struct data array[100000];
    scanf("%d%d",&N,&Q);
    for(i=0;i<N;i++)
    scanf("%d",&array[i].Time);
    for(i=0;i<N;i++)
    scanf("%d",&array[i].Score);
    qsort(array,N,sizeof(struct data),compare);
    for(i=N-1;i>0;i--)
     array[i-1].Time =array[i-1].Time + array[i].Time;
    for(i=0;i<Q;i++){
     scanf("%d",&input_val);
     printf("%d\n",array[N-input_val].Time);
    }
    return 0;
}
```

### Sample Input

```
5 2
2 3 9 4 5
3 5 11 6 7
5
3
```

### Sample Output

```
23
18
```

### Result

Thus, Program " **Scoring in Exam** " has been successfully executed

**Course:** DAA **Session:** Sorting Techniques **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

### Q. Max sum in sub-arrays

Given an array, find maximum sum of smallest and second smallest elements chosen from all possible sub-arrays. More formally, if we write all (nC2) sub-arrays of array of size >=2 and find the sum of smallest and second smallest, then our answer will be maximum sum among them. Input: The first line of input contains an integer T denoting the no of test cases. Then T test cases follow. Each test case contains an integer N denoting the size of the array. The next line contains N space separated values of the array. Output: For each test case in a new line print an integer denoting the maximum sum of smallest and second smallest elements chosen from all possible subarrays. Constraints: 1<=T<=100 1<=N<=100 1<=A[]<=100

### Source Code

```
#include<stdio.h>
int main()
{
 int t;
 scanf("%d",&t);
 while(t--)
 {
  int n,i,sum1=0,sum2=0,large,a,b;
  scanf("%d",&n);
  int ar[n+5];
  for(i=0;i<n;i++)
   scanf("%d",&ar[i]);
  sum2=ar[0]+ar[1];
  large=sum2;
  for(i=2;i<n;i++)
  {
   sum2=sum2-ar[i-2]+ar[i];
   if(sum2>large)
    large=sum2;
  }
  printf("%d\n",large);
 }
}
```

### Sample Input

```
2
3
4 5 1
5
4 3 1 5 6
```

### Sample Output

```
9
11
```

### Result

Thus, Program " **Max sum in sub-arrays** " has been successfully executed

**Course:** DAA **Session:** Divide and Conquer **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

### Q. Monk and Modulo Based Sorting

Monk likes to experiment with algorithms. His one such experiment is using modulo in sorting.He describes an array modulo sorted as: Given an integer k, we need to sort the values in the array according to their modulo with k. That is, if there are two integers a and b, and a%k

### Source Code

```
#include <iostream>
using namespace std;

int main(){
 int n,key,c,idx,k,key2;
 cin >> n;
 cin>>k;
 int arr[n];
 for (int i=0; i<n; i++){
  cin >> arr[i];
 }
 for (int i=1;i<n;i++){
  key = arr[i]%k;
  key2 = arr[i];
  c = i-1;
  idx = i;
  while (c>=0){
   if (key<arr[c]%k){
    arr[idx] = arr[c];
    arr[c] = key2;
    idx-=1;
   }
   else{
    break;
   }
   c-=1;
  }
 }
 for (int i=0;i<n;i++){
  cout << arr[i] << " ";
 }
 return 0;
}
```

### Sample Input

```
5 6
12 18 17 65 46
```

### Sample Output

```
12 18 46 17 65
```

### Result

Thus, Program " **Monk and Modulo Based Sorting** " has been successfully executed

**Course:** DAA **Session:** Divide and Conquer **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. MIN MAX**

Given a list of N integers, your task is to select K integers from the list such that its unfairness is minimized. if (x1,x2,x3,,xk) are K numbers selected from the list N, the unfairness is defined as max(x1,x2,,xk)-min(x1,x2,,xk) where max denotes the largest integer among the elements of K, and min denotes the smallest integer among the elements of K. Input Format The first line contains an integer N. The second line contains an integer K. N lines follow. Each line contains an integer that belongs to the list N. Note: Integers in the list N may not be unique. Output Format An integer that denotes the minimum possible value of unfairness. Constraints 2<=N<=10^5 2<=K<=N 0<= integer in N <=10^9

**Source Code**

```
#include<iostream>
using namespace std;
int main()
{
int a[100],n,k,temp,min;
 cin>>n;
  cin>>k;
  for(int i=0;i<n;i++)
  {
    cin>>a[i];
  }
  for(int i=0;i<n-1;i++)
  {
    for(int j=0;j<n-i-1;j++)
    {
     if(a[j]>a[j+1])
     {
       temp=a[j];
       a[j]=a[j+1];
       a[j+1]=temp;
     }
    }
  }
  for(int l=0;l<n;l++)
  {//cout<<a[l];
   min=a[k-1]-a[0];
  }
  cout<<min;
  return 0;
}
```

**Sample Input**

```
3
2
21
66
11
```

**Sample Output**

```
10
```

**Result**

Thus, Program " **MIN MAX** " has been successfully executed

**Course:** DAA **Session:** Divide and Conquer **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

### Q. Mutual Smallest Distance

Given n points in d-dimensions, find two whose mutual distance is smallest.

### Source Code

```
#include <stdio.h>
#include<math.h>
int main()
{
int n,i,a[100],b[100],j,x,y;
  float min,dis;
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    scanf("%d %d",&a[i],&b[i]);
  }
  min=1000;
  for(i=0;i<n-1;i++)
  {
    for(j=i+1;j<n;j++)
    {
      x=a[j]-a[i];
      y=b[j]-b[i];
      dis=sqrt((x*x)+(y*y));
      if(dis<min)
        min=dis;
    }
  }
  printf("%0.6f",min);
 return 0;
}
```

### Sample Input

```
4
10 20
5 1
2 10
3 4
```

### Sample Output

```
3.605551
```

### Result

Thus, Program " **Mutual Smallest Distance** " has been successfully executed

**Course:** DAA **Session:** Divide and Conquer **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. Benny and Gifts**

Little pig Benny has just taken a shower. Now she is going to buy some gifts for her relatives. But the problem is that Benny doesn't know how to reach to the gift shop. Her friend Mike has created a special set of instructions for her. A set of instructions is a string which consists of letters {'L', 'R', 'U', 'D'}. For simplicity, let's assume that Benny is staying at point (0, 0) on the infinite plane. She consistently fulfills all the instructions written by Mike. Let's assume that now she is staying at point (X, Y). Then depending on what is the current instruction she moves in some direction: 'L' -- from (X, Y) moves to point (X, Y - 1) 'R' -- from (X, Y) moves to point (X, Y + 1) 'U' -- from (X, Y) moves to point (X - 1, Y) 'D' -- from (X, Y) moves to point (X + 1, Y) The weather is cold because it's winter now. Initially, all points are snowy. But if Benny have already visited some point at any time this point becomes icy (because Benny has just taken a shower). Every time, when Benny makes a step into icy points she slips and falls down. You are given a string S which denotes a set of instructions for Benny. Your task is to calculate how may times she will fall down. Input format Single line contains string S.

**Source Code**

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

typedef struct node
{
 int x;
 int y;
}node;

int cmp(void *a, void *b)
{
 node l=*(node *)a;
 node r=*(node *)b;
 if(l.x > r.x)
  return 1;
 else if(l.x == r.x)
 {
  if(l.y > r.y)
   return 1;
  else
   return(0);
 }
 return(0);
}

int main()
{
 char arr[100005];
 node ans[100005];
 scanf("%s", arr);
 int n=strlen(arr),i;
 ans[0].x=0;
 ans[0].y=0;
 int x=0, y=0, k=1;
 for(i=0;i<n;i++)
 {
  if(arr[i]=='L')
  {
   y--;
  }
  else if(arr[i]=='R')
  {
   y++;
  }
  else if(arr[i]=='U')
  {
   x--;
  }
  else if(arr[i]=='D')
  {
   x++;
  }
  ans[k].x=x;
  ans[k].y=y;
  k++;
 }

 qsort(ans, k, sizeof(node), cmp);
 int c=0;

 for(i=0;i<k-1;i++)
 {
  if(ans[i].x==ans[i+1].x && ans[i].y==ans[i+1].y)
  {
   c++;
  }
 }
 printf("%d\n", c);
  return 0;
}
```

**Sample Input**

```
RRULDL
```

**Sample Output**

```
2
```

**Result**

Thus, Program " **Benny and Gifts** " has been successfully executed

**Course:** DAA **Session:** Divide and Conquer **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

### Q. MIN MAX

Given a list of N integers, your task is to select K integers from the list such that its unfairness is minimized. if (x1,x2,x3,,xk) are K numbers selected from the list N, the unfairness is defined as max(x1,x2,,xk)-min(x1,x2,,xk) where max denotes the largest integer among the elements of K, and min denotes the smallest integer among the elements of K. Input Format The first line contains an integer N. The second line contains an integer K. N lines follow. Each line contains an integer that belongs to the list N. Note: Integers in the list N may not be unique. Output Format An integer that denotes the minimum possible value of unfairness. Constraints 2<=N<=10^5 2<=K<=N 0<= integer in N <=10^9

### Source Code

```
#include <stdio.h>

int main()  {
   int t;
   scanf("%d", &t);
   while(t--)  {
      long int n;
      scanf("%ld", &n);
      long int a[n], b, h = 0, m1, m2, i;
      for(i = 0; i < 2; i++)  {
         scanf("%ld%ld", &a[i], &b);
         h = h - b;
         a[i] = a[i] + b;
      }
      if(a[0] > a[1]) {
         m1 = a[0];
         m2 = a[1];
      }  else {
         m1 = a[1];
         m2 = a[0];
      }
      for(i = 2; i < n; i++)  {
         scanf("%ld%ld", &a[i], &b);
         h = h - b;
         a[i] = a[i] + b;
         if(a[i] > m1)  {
            m2 = m1;
            m1 = a[i];
         }  else if(a[i] > m2)
            m2 = a[i];
      }
      h = h + m1 + m2;
      printf("%ld\n", h);
   }
   return 0;
}
```

### Sample Input

```
3
2
21
66
11
```

### Sample Output

```
10
```

### Result

Thus, Program " **MIN MAX** " has been successfully executed

**Course:** DAA **Session:** Divide and Conquer **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. Closest pair of points problem**

Given n points, find two points with the smallest distance to each other.

**Source Code**

```
#include <stdio.h>
#include<math.h>
int main()
{
int n,i,a[100],b[100],j,x,y;
  float min,dis;
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    scanf("%d %d",&a[i],&b[i]);
  }
  min=1000;
  for(i=0;i<n-1;i++)
  {
    for(j=i+1;j<n;j++)
    {
      x=a[j]-a[i];
      y=b[j]-b[i];
      dis=sqrt((x*x)+(y*y));
      if(dis<min)
        min=dis;
    }
  }
  printf("%0.6f",min);
 return 0;
}
```

**Sample Input**

```
4
10 20
5 1
2 10
3 4
```

**Sample Output**

```
3.605551
```

**Result**

Thus, Program " **Closest pair of points problem** " has been successfully executed

**Q. Rank List**

Mid semester marks of a particular subject is announced , since you are curious in knowing your position in class so you decided to make a rank list . You are given the name , scholar number and marks of every student in your class. You have to come up with accurate rank list i.e student having maximum marks at the top and if two students are having same marks then the student having lexicographically smaller name comes first , if both name and marks of the student collide then student having smaller scholar number comes first. Input: First line of input contains N - Total number of students in class Next N line contains name of student , scholar number and marks scored in exam . Output: Print the ranklist of students as explained above. Constraints 1 <= N <= 1000 1 <= length of name <= 10 1 <= scholar number <= 1000 0 <= marks <= 30

**Source Code**

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

typedef struct
{
 char name[10];
 int id;
 int mark;
}estudiante;

void quicksort(estudiante *A, int len)
{
  if (len < 2) return;

  estudiante pivot = A[len / 2];

  int i, j, band = 1;
  for (i = 0, j = len - 1; ; i++, j--)
  {
    while(band)
    {
     if(A[i].mark <= pivot.mark)
     {
      if(A[i].mark == pivot.mark)
      {
       if(strcmp(A[i].name,pivot.name) == 0)
       {
        if(A[i].id >= pivot.id)
         band = 0;
       }
       else
       {
         if(strcmp(A[i].name,pivot.name) > 0)
         band = 0;
       }
      }
      else
       band = 0;
     }
     if(band)
      i++;
    }
    while (!band)
    {
     if(A[j].mark >= pivot.mark)
     {
      if(A[j].mark == pivot.mark)
      {
       if(strcmp(A[j].name,pivot.name) == 0)
        {
        if(pivot.id >= A[j].id)
         band = 1;
        }
       else
       {
        if(strcmp(pivot.name,A[j].name) > 0)
         band = 1;
       }
      }
      else
       band = 1;
     }
     if(!band)
      j--;
    }

    if (i >= j) break;

    estudiante temp = A[i];
    A[i]    = A[j];
    A[j]    = temp;
  }

  quicksort(A, i);
  quicksort(A + i, len - i);
}

int main()
{
 estudiante *v;
 int n, i;

 scanf("%d", &n);
 v = (estudiante *)malloc(n * sizeof(estudiante));
 for(i = 0 ; i < n ; i++)
  scanf("%s %d %d", v[i].name, &v[i].id, &v[i].mark);
 quicksort(v,n);
 for(i = 0 ; i < n ; i++)
  printf("%s %d %d\n", v[i].name, v[i].id, v[i].mark);
 free(v);
 return 0;
}
```

**Sample Input**

```
2
bilal 8 28
sai 2 8
```

**Sample Output**

```
bilal 8 28
sai 2 8
```

**Result**

Thus, Program " **Rank List** " has been successfully executed

**Course:** DAA **Session:** Divide and Conquer **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. Puchi and Luggage**

Puchi hates to carry luggage, but unfortunately he got a job to carry the luggage of his N friends in office. Each day, one of his N friends, gives him the luggage of a particular weight to carry. You will be given the weight of luggage of each friend in the array Weight, where Weight i is the weight of luggage of ith friend carried by Puchi on ith day. It is given that all the luggages carried by Puchi are distinct in their weights. As Prateek assigned this job to Puchi, so for each day, he wants to know the number of days in future when Puchi will have to carry the luggage , having weight less than the weight of luggage of current day. Please help Prateek for the same. Input: The first line contains a single integer T, denoting the number of test cases. In each test case, the following input will be present: First line contains an integer N, where N represents the number of friends. Next N line contains N integers, where ith line contains ith integer, which represents Weight i. Output: Output exactly T lines. Each line contains N integer separated by a space, where ith integer represents the number of luggage of future, which are less than the weight of luggage of the current day. Constraints: Subtask 1: 1 <= T <= 30 1<= N <= 10^4 1<= Weight i <= 10^6 Subtask 2: 1 <= T <= 10 1<= N <= 10^5 1<= Weight i <= 10^6

**Source Code**

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int key;
    struct node *left;
    struct node *right;
    int height;
    int size;
};

int max(int a, int b);

int height(struct node *N)
{
    if (N == NULL)
        return 0;
    return N->height;
}

int size(struct node *N)
{
    if (N == NULL)
        return 0;
    return N->size;
}

int max(int a, int b)
{
    return (a > b)? a : b;
}


struct node* newNode(int key)
{
    struct node* node = (struct node*)
                        malloc(sizeof(struct node));
    node->key   = key;
    node->left   = NULL;
    node->right  = NULL;
    node->height = 1;
    node->size = 1;
    return(node);
}

struct node *rightRotate(struct node *y)
{
    struct node *x = y->left;
    struct node *T2 = x->right;

    x->right = y;
    y->left = T2;

    y->height = max(height(y->left), height(y->right))+1;
    x->height = max(height(x->left), height(x->right))+1;

    y->size = size(y->left) + size(y->right) + 1;
    x->size = size(x->left) + size(x->right) + 1;

    return x;
}

struct node *leftRotate(struct node *x)
{
    struct node *y = x->right;
    struct node *T2 = y->left;

    y->left = x;
    x->right = T2;

    x->height = max(height(x->left), height(x->right))+1;
    y->height = max(height(y->left), height(y->right))+1;

    x->size = size(x->left) + size(x->right) + 1;
    y->size = size(y->left) + size(y->right) + 1;

    return y;
}

int getBalance(struct node *N)
{
    if (N == NULL)
        return 0;
    return height(N->left) - height(N->right);
}


struct node* insert(struct node* node, int key, int *count)
{
    if (node == NULL)
        return(newNode(key));

    if (key < node->key)
        node->left  = insert(node->left, key, count);
    else
    {
        node->right = insert(node->right, key, count);

        *count = *count + size(node->left) + 1;
    }


    node->height = max(height(node->left), height(node->right)) + 1;
    node->size   = size(node->left) + size(node->right) + 1;

    int balance = getBalance(node);

    if (balance > 1 && key < node->left->key)
        return rightRotate(node);

    if (balance < -1 && key > node->right->key)
        return leftRotate(node);

    if (balance > 1 && key > node->left->key)
    {
        node->left =  leftRotate(node->left);
        return rightRotate(node);
    }

    if (balance < -1 && key < node->right->key)
    {
        node->right = rightRotate(node->right);
        return leftRotate(node);
    }

    return node;
}

void constructLowerArray (int arr[], int countSmaller[], int n)
{
  int i;
  struct node *root = NULL;

  for  (i = 0; i < n; i++)
     countSmaller[i] = 0;

  for (i = n-1; i >= 0; i--)
  {
     root = insert(root, arr[i], &countSmaller[i]);
  }
}

void printArray(int arr[], int size)
{
  int i;
  printf("\n");
  for (i=0; i < size; i++)
    printf("%d ", arr[i]);
}


int main()
{
    int t;
    scanf("%d",&t);
    while(t--){
        int n;
        scanf("%d",&n);
        int arr[n],i;
        for(i=0;i<n;i++)    scanf("%d",&arr[i]);


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

        constructLowerArray(arr, low, n);
        printArray(low, n);

    }
  return 0;
}
```

**Sample Input**

```
1
4
2
1
4
3
```

**Sample Output**

```
1 0 1 0
```

**Result**

Thus, Program " **Puchi and Luggage** " has been successfully executed

**Course:** DAA **Session:** Divide and Conquer **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

### Q. Closest Pair of Points

A typical Divide and Conquer algorithm solves a problem using following three steps. 1. Divide: Break the given problem into subproblems of same type. 2. Conquer: Recursively solve these subproblems 3. Combine: Appropriately combine the answers Closest Pair of Points The problem is to find the closest pair of points in a set of points in x-y plane. The problem can be solved in O(n^2) time by calculating distances of every pair of points and comparing the distances to find the minimum. The Divide and Conquer algorithm solves the problem in O(nLogn) time.

### Source Code

```
#include <stdio.h>
#include<math.h>
int main()
{
int n,i,a[100],b[100],j,x,y;
  float min,dis;
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    scanf("%d %d",&a[i],&b[i]);
  }
  min=1000;
  for(i=0;i<n-1;i++)
  {
    for(j=i+1;j<n;j++)
    {
      x=a[j]-a[i];
      y=b[j]-b[i];
      dis=sqrt((x*x)+(y*y));
      if(dis<min)
        min=dis;
    }
  }
  printf("The smallest distance is %0.6f",min);
 return 0;
}
```

### Sample Input

```
4
10 20
5 1
2 10
3 4
```

### Sample Output

```
The smallest distance is 3.605551
```

### Result

Thus, Program " **Closest Pair of Points** " has been successfully executed

**Course:** DAA **Session:** Greedy Algorithm **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. EXPLORING RUINS**

Little robot CJ-17 is exploring ancient ruins. He found a piece of paper with a word written on it. Fortunately, people who used to live at this location several thousand years ago used only two letters of modern English alphabet: 'a' and 'b'. It's also known, that no ancient word contains two letters 'a' in a row. CJ-17 has already recognised some of the word letters, the others are still unknown. CJ-17 wants to look up all valid words that could be written on this paper in an ancient dictionary. He needs your help. Find him the word, which is the first in alphabetical order and could be written on the paper. Input format The first line contains non-empty string s consisting of 'a', 'b' and '?' characters. Character '?' corresponds to unrecognized letter. It's guaranteed, that there exists at least one ancient word, that could be written on the paper. Constraints Length of s is at most 50. Output format Output the first in alphabetical order word, that could be written on the paper, found by CJ-17.

**Source Code**

```
#include<stdio.h>
#include<string.h>
int main()
{
   char s[60];
   scanf("%s",s);
   int i=0,k=strlen(s),j;
   while(s[i]!='\0')
   {
      if(s[i]=='?')
      {
         j=i;
         if(j==k)
         {
            if(s[j-1]=='a')
               s[j]='b';
            else
               s[j]='a';

          //   t=1;
            continue;
         }
         if(j==0)
         {
            if(s[j+1]=='a')
               s[j]='b';
            else
               s[j]='a';
           // t=1;
           continue;
         }
         if(s[j+1]=='a' || s[j-1]=='a')
         {
            s[i]='b';
         }
         else
         {
            s[i]='a';
         }
      }
      i++;
   }
   printf("%s",s);
  return 0;
}
```

**Sample Input**

```
?ba??b
```

**Sample Output**

```
ababab
```

**Result**

Thus, Program " **EXPLORING RUINS** " has been successfully executed

**Course:** DAA **Session:** Greedy Algorithm **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. MILLY & CHOCOLATES**

Milly loves to eat chocolates. She has N different chocolates. She needs to choose only one of them. At the moment, ith chocolate already has Pi pieces. The jth piece of the ith chocolate requires Ti,j seconds to eat. Milly knows that she will take K seconds to break one piece from any chocolate and wait for M seconds after eating any piece. Your task is to help her in selecting a chocolate that she will eat completely in minimum possible time (including the waiting time). Input: First line of the input will contain a single integer T denoting the number of test cases. Now for every test case, the first line will contain three space separated integers : N, K and M. The next line will contain N space separated integers denoting Pi (The number of pieces in the ith chocolate). Each of the next N lines will contain j space separated integers denoting Ti,j (Time (in seconds) required to eat jth piece of the ith chocolate. Output: For every test case, print two space separated integers in a new line : the index (1-based) of the resultant chocolate and the minimum number of seconds Milly needs to eat that chocolate. If multiple answers are possible then consider the lowest indexed chocolate. Constraints: 1<=T<=10 1<=N<=10^3 1<=K,M<=10^5 1<=Pi<=100 1<=Ti,j<=10^9

**Source Code**

```
#include <stdio.h>

int main()
{
   long long int t,n,k,m,i,j,l,x,min,index;
   scanf("%lld",&t);
   for(i=1;i<=t;i++)
   {
    scanf("%lld %lld %lld",&n,&k,&m);
    long long int a[n],ans[n];
    for(j=0;j<n;j++)
    {
     scanf("%lld",&a[j]);
     ans[j]=k*a[j]+m*(a[j]-1);
    }
    for(j=0;j<n;j++)
    {
     for(l=0;l<a[j];l++)
     {
      scanf("%lld",&x);
      ans[j]=ans[j]+x;
     }
    }
    min=1000000000000000000;
    for(j=0;j<n;j++)
    {
     if(ans[j]<min)
     {
      index=j+1;
      min=ans[j];
     }
    }
    printf("%lld %lld\n",index,min);
   }
   return 0;
}
```

**Sample Input**

```
1
2 10 10
1 2
10
4 2
```

**Sample Output**

```
1 20
```

**Result**

Thus, Program " **MILLY & CHOCOLATES** " has been successfully executed

**Course:** DAA **Session:** Greedy Algorithm **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. CHANDU & CONSECUTIVE LETTERS**

Chandu is very fond of strings. (Or so he thinks!) But, he does not like strings which have same consecutive letters. No one has any idea why it is so. He calls these strings as Bad strings. So, Good strings are the strings which do not have same consecutive letters. Now, the problem is quite simple. Given a string S, you need to convert it into a Good String. You simply need to perform one operation - if there are two same consecutive letters, delete one of them. Input: The first line contains an integer T, denoting the number of test cases. Each test case consists of a string S, which consists of only lower case letters. Output: For each test case, print the answer to the given problem. Constraints: 1 <= T <= 10 1 <= |S| <= 30

**Source Code**

```
#include <stdio.h>


int main()
{
   int n=0,i=0,j=0,k=0;
   char a[30];
   scanf("%d",&n);

   for(;i<n;i++)
   {
       scanf("%s",a);
       for(;a[k]!='\0';k++)
       {
        if(a[k]!=a[k-1])
        {
         printf("%c",a[k]);
        }
       }
       k=0;
       printf("\n");


   }
   return 0;
}
```

**Sample Input**

```
3
abb
aaab
ababa
```

**Sample Output**

```
ab
ab
ababa
```

**Result**

Thus, Program " **CHANDU & CONSECUTIVE LETTERS** " has been successfully executed

**Course:** DAA **Session:** Greedy Algorithm **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. BEAUTIFUL PAIRS**

You are given two arrays, A and B, both containing N integers. A pair of indices (i,j) is beautiful if the ith element of array A is equal to the jth element of array B. In other words, pair (i,j) is beautiful if and only if Ai=Bj. Given A and B, there are k pairs of beautiful indices (i0,j0),...,(ik-1,jk-1). A pair of indices in this set is pairwise disjoint if and only if for each 0<=x<=y<=k-1 it holds that ix not equal to iy and jx not equal to jy. Change exactly 1 element in B so that the resulting number of pairwise disjoint beautiful pairs is maximal, and print this maximal number to stdout. Input Format: The first line contains a single integer, N (the number of elements in A and B). The second line contains N space-separated integers describing array A. The third line contains N space-separated integers describing array B. Constraints 1<=N<=10^3 1<= Ai<=10^3 1<= Bi <=10^3 Output Format: Determine and print the maximum possible number of pairwise disjoint beautiful pairs. Note: You must first change 1 element in B, and your choice of element must be optimal.

**Source Code**

```
//#include <cmath>
#include <cstdio>
//#include <vector>
#include <iostream>
//#include <algorithm>
using namespace std;


int main() {
    /* Enter your code here. Read input from STDIN. Print output to STDOUT */
            long long n;
          //   int a[1005];
            scanf("%lld",&n);
               int a[10000];
               int b[10000];
               for(int i=1;i<=10000;i++)
                  {
                    a[i]=0;
                    b[i]=0;
               }
               for(int i=1;i<=n;i++)
                  {
                      int temp;
                       scanf("%d",&temp);
                       a[temp]++;
               }
               for(int i=1;i<=n;i++)
                  {
                      int temp;
                       scanf("%d",&temp);
                       b[temp]++;
               }
               long long ans=0;
               for(int i=1;i<=1001;i++)
                  {
                    ans+=min(a[i],b[i]);
               }
               if(ans!=n)
                  printf("%lld\n",ans+1);
               else
                  printf("%lld\n",ans-1);

    return 0;
}
```

**Sample Input**

```
3
  1 2 2
  1 2 3
```

**Sample Output**

```
3
```

**Result**

Thus, Program " **BEAUTIFUL PAIRS** " has been successfully executed

**Course:** DAA **Session:** Greedy Algorithm **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

### Q. HUNGER GAMES

You have reached the final level of Hunger Games and as usual a tough task awaits you. You have a circular dining table and N hungery animals.You need to place them on the table.Each animal has a hunger value. You can place the animals in any order on the circular table.Once you place them you need to calculate the Danger value of your arrangement. The danger value of the arrangement is the maximum difference of hunger values of all the adjacent seated animals.You need to keep this danger value as low as possible. Input: The first line contains N integers. The second line contains the hunger values of N animals. Output: Print the minimum possible danger value. Constraints: 3<=N<=1000 1<=Hunger Value<=1000

### Source Code

```
#include <stdio.h>
void qsort(int a[], int first, int last)
{
   int pivot, j, temp, i;

    if(first<last){
       pivot=first;
       i=first;
       j=last;

       while(i<j){
          while(a[i]<=a[pivot]&&i<last)
             i++;
          while(a[j]>a[pivot])
             j--;
          if(i<j){
             temp=a[i];
              a[i]=a[j];
              a[j]=temp;
          }
       }
       temp=a[pivot];
       a[pivot]=a[j];
       a[j]=temp;
       qsort(a,first,j-1);
       qsort(a,j+1,last);
   }
}
int main()
{
   int n, i, d=0;
   scanf("%d",&n);
   int a[n];
   for(i=0;i<n;i++)
   scanf("%d",&a[i]);
   qsort(a,0,n-1);
   d=a[n-1]-a[n-3];
   for(i=n-2;i>=2;i--)
   if(d<(a[i]-a[i-2]))
   d=a[i]-a[i-2];
   printf("%d",d);
return 0;
}
```

### Sample Input

```
4
5 10 6 8
```

### Sample Output

```
4
```

### Result

Thus, Program " **HUNGER GAMES** " has been successfully executed

**Course:** DAA **Session:** Greedy Algorithm **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. PERMUTING TWO ARRAYS**

Consider two n-element arrays of integers, A=[a0,a1,...,an-1] and B=[b0,b1,...,bn-1]. You want to permute them into some A' and B' such that the relation ai'+bi'>=k holds for all i where 0<=i<=n. For example, if A=[0,1] B=[0,2] , and k=1, a valid A', B' satisfying our relation would be A'=[1,0] and B'=[0,2]. You are given q queries consisting of A,B and k. For each query, print YES on a new line if some permutations A', B' exist satisfying the relation above. If no valid permutations exist, print NO instead. Input Format The first line contains an integer, q, denoting the number of queries. The 3q subsequent lines describe each of the q queries in the following format: 1.The first line contains two space-separated integers describing the respective values of n(the size of arraysA and B) and k (the relation variable). 2. The second line contains n space-separated integers describing the respective elements of array A. 3.The third line contains n space-separated integers describing the respective elements of array B. Constraints 1<=q<=10 1<=n<=1000 1<=k<=10^9 0<=ai, bi<=10^9 Output Format For each query, print YES on a new line if valid permutations exist; otherwise, print NO.

**Source Code**

```
#include<stdio.h>
int main()
{
   int t,array1[10000],array2[10000],size,k,i,j,temp,flag;
   scanf("%d",&t);
   while(t--)
   {
      scanf("%d %d",&size,&k);
      for(i=0;i<size;i++)
      {
         scanf("%d",&array1[i]);
      }
      for(i=0;i<size;i++)
      {
         scanf("%d",&array2[i]);
      }
      for(i=0;i<size-1;i++)
      {
         for(j=0;j<size-1-i;j++)
         {
            if(array1[j]>array1[j+1])
            {
               temp=array1[j];
               array1[j]=array1[j+1];
               array1[j+1]=temp;
            }
         }
      }
      for(i=0;i<size-1;i++)
      {
         for(j=0;j<size-1-i;j++)
         {
            if(array2[j]<array2[j+1])
            {
               temp=array2[j];
               array2[j]=array2[j+1];
               array2[j+1]=temp;
            }
         }
      }
      flag=0;
      for(i=0,j=0;i<size,j<size;i++,j++)
      if(array1[i]+array2[j]<k)
      {
         flag=1;
         break;
      }
      if(flag==1)
      printf("NO\n");
      else
      printf("YES\n");
   }}
```

**Sample Input**

```
2
3 10
2 1 3
7 8 9
4 5
1 2 2 1
3 3 3 4
```

**Sample Output**

```
YES
NO
```

**Result**

Thus, Program " **PERMUTING TWO ARRAYS** " has been successfully executed

**Course:** DAA **Session:** Greedy Algorithm **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. JIM AND THE ORDERS**

Jim's Burgers has n hungry burger fans waiting in line. Each unique order, i,is placed by a customer at time ti, and the order takes di units of time to process. Given the information for all n orders, can you find and print the order in which all n customers will receive their burgers? If two or more orders are fulfilled at the exact same time t, sort them by ascending order number. Input Format The first line contains a single integer, n, denoting the number of orders. Each of the n subsequent lines contains two space-separated integers describing the respective values of ti and di for order i. Constraints 1 <= n<= 10^3 1 <= i<= n 1 <= ti, di <=10^6 Output Format Print a single line of n space-separated order numbers (recall that orders are numbered from 1 to n ) describing the sequence in which the customers receive their burgers. If two or more customers receive their burgers at the same time, print the smallest order number first.

**Source Code**

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct data
{
 int i;
 int val;
};
struct data st[2100];
int compare(const void *p1, const void *p2)
{
 const struct data *elem1 = p1;
 const struct data *elem2 = p2;

 if ( elem1->val < elem2->val)
  return -1;
 else if (elem1->val > elem2->val)
  return 1;
 else
  return 0;
}
void print(int n,int i,int j)
{
 int k;
 for(k=0;k<n-1;k++)
 {
  printf("%d ",st[k].i);
 }
 printf("%d\n",st[k].i);

}
int main()
{
 int i,k,n,l;
 scanf("%d",&n);
 for(i=0;i<n;i++)
 {
  scanf("%d%d",&k,&l);
  st[i].i=i+1;
  st[i].val=k+l;
 }
 qsort(st,n,sizeof(struct data),compare);

 print(n,0, n);
 return 0;
}
```

**Sample Input**

```
3
1 3
2 3
3 3
```

**Sample Output**

```
1 2 3
```

**Result**

Thus, Program " **JIM AND THE ORDERS** " has been successfully executed

**Course:** DAA **Session:** Greedy Algorithm **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. PRIYANKA AND TOYS**

Little Priyanka visited a kids' shop. There are N toys and their weight is represented by an array W=[w1, w2,...,wN]. Each toy costs 1 unit, and if she buys a toy with weight w', then she can get all other toys whose weight lies between [w',w'+4] (both inclusive) free of cost. Input Format: The first line contains an integer N i.e. number of toys. Next line will contain N integers,w1,w2,...,wN , representing the weight array. Output Format: Minimum units with which Priyanka could buy all of toys. Constraints 1 <= N <= 10^5 0 <= wi <= 10^4, where i belongs to [1,N]

**Source Code**

```
#include <iostream>
using namespace std;
int main()
{
  int n,i,j,a[100],temp=0;
  cin>>n;
  for(i=0;i<n;i++)
    cin>>a[i];
  for(i=0;i<n-1;i++)
  {
    for(j=0;j<n-i-1;j++)
    {
      if(a[j]>a[j+1]);
      {
        temp=a[j];
        a[j]=a[j+1];
        a[j+1]=temp;
      }
    }
  }
  int cnt=1,w=a[0];
  for(i=1;i<n;i++)
  {
    if(a[i]>w+4)
    {
      cnt++;
      w=a[i];
    }
  }
  cout<<cnt+1;


 return 0;
}
```

**Sample Input**

```
5
1 2 3 17 10
```

**Sample Output**

```
3
```

**Result**

Thus, Program " **PRIYANKA AND TOYS** " has been successfully executed

**Course:** DAA **Session:** Greedy Algorithm **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. ALGORITHMIC CRUSH**

You are given a list of size N, initialized with zeroes. You have to perform M operations on the list and output the maximum of final values of all the N elements in the list. For every operation, you are given three integers a,b and k and you have to add value k to all the elements ranging from index a to b (both inclusive). Input Format First line will contain two integers N and M separated by a single space. Next M lines will contain three integers a,b and k separated by a single space. Numbers in list are numbered from l to N. Constraints 3<=N<=10^7 1<=M<=2 * 10 ^5 1<=a<=b<=N 0<=k<=10^9 Output Format A single line containing maximum value in the updated list.

**Source Code**

```
#include <stdio.h>
long long arr[10000005];
long long diff[10000005];
int main(void)
{
    int a,b,k,n,m,i;
    long long max,val;
    scanf("%d%d",&n,&m);
    while(m--)
    {
     scanf("%d%d%d",&a,&b,&k);

     if(a==1)
     {
      arr[1] += k;

       if(b<n)
       diff[b] -= k;
     }
     else if(b<n)
     {
      diff[a-1] += k;
      diff[b] -= k;
     }
     else if(b==n)
      diff[a-1] += k;
    }

    max = arr[1];
    val = arr[1];

    for(i=1;i<n;i++)
    {
      val += diff[i];
      if(val > max)
       max = val;
    }

    printf("%lld\n",max);
 return 0;
}
```

**Sample Input**

```
5 3
1 2 100
2 5 100
3 4 100
```

**Sample Output**

```
200
```

**Result**

Thus, Program " **ALGORITHMIC CRUSH** " has been successfully executed

**Course:** DAA **Session:** Dynamic Programming **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

### Q. PALINDROME COUNT

Given a string S, count the number of non empty sub strings that are palindromes. A sub string is any continuous sequence of characters in the string. A string is said to be palindrome, if the reverse of the string is same as itself. Two sub strings are different if they occur at different positions in S Input Input contains only a single line that contains string S. Output Print a single number, the number of sub strings that are palindromes. Constraints 1 <= |S| <= 50 S contains only lower case latin letters, that is characters a to z.

### Source Code

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

int palindrome(int i,int j,char *str)
{
while(i<j)
{
 if(str[i]!=str[j])
 return 0;

 i++;
 j--;
}
return 1;
}

int main()
{
  int count=0,i,j;
  char str[100];

    //gets(str);
    scanf("%s",str);
    for(i=0;i<strlen(str)-1;i++)
    for(j=i+1;j<strlen(str);j++)
    if(palindrome(i,j,str))
    count++;
    int d=count+strlen(str);
    printf("%d\n",d);
    return 0;
}
```

### Sample Input

```
dskjkd
```

### Sample Output

```
7
```

### Result

Thus, Program " **PALINDROME COUNT** " has been successfully executed

**Course:** DAA **Session:** Dynamic Programming **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. Little Deepu and his Girlfriend**

Little Deepu recently got a new job, so to celebrate, he decided to take his girfriend, Kate, out on a fancy candle light dinner. To make things perfect for Kate, Deepu planned the night to the last detail. The food, the music, the ambience, everything was perfect, but stupidly enough Deepu forgot to get her a present. After dinner, Kate asks our Deepu for her gift, when she finds out that he hasn't got her any she gets upset, to try smooth things with her, he proposes a game, this makes Kate so angry that she dumps him right there(bad move Deepu). Well its safe to say the rest of the evening was uneventful for Little Deepu but the game he proposed was pretty interesting. To put the game a bit formally, we have two players Little Deepu and Kate and M items in the bag B, also we have a game set S with N elements, each element of game set is an integer. The game is played as follows, each player takes turn to pick an element from the game set S and removes that number of items from the bag, the player that is unable to remove the items from the bag looses the game. Little Deepu start the game ,If both Little Deepu and Kate play the game optimally, your task is to determine who wins the game. Input: First line contains a integer T , number of test cases. Each test case contain two lines , first line contain two integer M and N and second line contain elements of S. Output: For each test case print name of the winner of the game . Constraints: 1<=T <= 1000 1 <= M <= 10000 1 <= N <= 100 1 <= S[i] <= M

**Source Code**

```
#include<stdio.h>
#include<string.h>
//using namespace std;
int main() {

 int m,t;
 scanf("%d",&t);
 while(t--) {

  int i,j,m,n;
  scanf("%d%d",&m,&n);
  int s[n];
  int dp[10005];
  memset(dp,0,sizeof(dp));
  for(i=0;i<n;i++) {

   scanf("%d",&s[i]);
   dp[s[i]]=1;
  }
  for(i=0;i<=m;i++) {

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

    if(i-s[j]>=0 && dp[i-s[j]]==0) {
     dp[i]=1;
     break;
    }
   }
  }
  if(dp[m]==1)
   printf("Little Deepu\n");
  else
   printf("Kate\n");
 }
 return 0;
}
```

**Sample Input**

```
2
3 2
1 2
5 3
1 2 3
```

**Sample Output**

```
Kate
Little Deepu
```

**Result**

Thus, Program " **Little Deepu and his Girlfriend**" has been successfully executed

**Course:** DAA **Session:** Dynamic Programming **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. GOLD MINES**

There is a rectangular grid of gold mine. The grid has R rows and C columns. So it has R*C cells in total. The rows are numbered from 1 to R and the columns are numbered from 1 to C. The top most row has number 1, the row next to it has number 2 and so on. Similarly, the left most column has number 1, the column next to it has number 2 and so on. Each cell in the grid has a unique coordinate which is (x, y) where x is the row number and y is the column number of that particular cell. Each cell in the grid has certain amount of gold in it. Total gold in a sub rectangle of the grid is the sum of all units of gold contained in the cells that are inside the rectangle. Your task is to find the total gold in the given sub rectangle. A sub rectangle of the grid is defined by four integers x1, y1, x2 and y2. A cell (x, y) belongs to the sub rectangle if and only if x1 <= x <= x2 and y1 <= y <=y2 Input First line of the input contains two space separated integers, R and C. It is followed by R lines, each line has C space separated integers. Each integer denotes the units of gold present in that particular cell. Next line has number Q, it is followed by Q queries each query in a single line. Each query is four space separated integers x1, y1, x2 and y2. Output For each query, you have to print a single number the total gold contained in that sub rectangle. Constraints 1 <= R <= 1000 1 <= C <= 1000 1 <= x1 <= x2 <= R 1 <= y1 <= y2 <= C Amount of gold in each cell is an integer from 0 to 10^6

**Source Code**

```
#include<stdio.h>
int main()
{
   int r,c,i,j;
   scanf("%d%d",&r,&c);
   long grid[r][c];
   for(i=0;i<r;i++)
     for(j=0;j<c;j++)
         scanf("%ld",&grid[i][j]);

    for(i=0;i<r;i++)
     for(j=0;j<c;j++)
      {
         grid[i][j]+=i>0?grid[i-1][j]:0;
         grid[i][j]+=j>0?grid[i][j-1]:0;
         grid[i][j]-=i>0&&j>0?grid[i-1][j-1]:0;
      }

    int q;
    scanf("%d",&q);
    while(q)
    {
       q--;
       int x1,y1,x2,y2;
       long total=0;
       scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
       x1--;
       y1--;
       x2--;
       y2--;
       total=grid[x2][y2];
       total-=x1>0?grid[x1-1][y2]:0;
       total-=y1>0?grid[x2][y1-1]:0;
       total+=x1>0&&y1>0?grid[x1-1][y1-1]:0;

       printf("%ld\n",total);
    }
    return 0;
}
```

**Sample Input**

```
4 4
2 8 9 7
5 8 1 7
5 7 3 5
4 8 7 4
4
1 2 4 2
1 4 2 4
1 1 4 2
2 4 2 4
```

**Sample Output**

```
31
14
47
7
```

**Result**

Thus, Program " **GOLD MINES** " has been successfully executed

**Course:** DAA **Session:** Dynamic Programming **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

### Q. Stack of Bricks

You and your friend decide to play a game using a stack consisting of N bricks. In this game, you can alternatively remove 1, 2 or 3 bricks from the top, and the numbers etched on the removed bricks are added to your score. You have to play so that you obtain the maximum possible score. It is given that your friend will also play optimally and you make the first move. Input Format First line will contain an integer T i.e. number of test cases. There will be two lines corresponding to each test case: first line will contain a number N i.e. number of elements in the stack and next line will contain N numbers i.e. numbers etched on bricks from top to bottom. Constraints 1 <= T <= 5 1 <= N <= 10^5 0 <= each number on brick <= 10^9 Output Format For each test case, print a single line containing your maximum score.

### Source Code

```
#include <stdio.h>
long long min(long long a, long long b) { return (a<b?a:b); }
int main() {
   int T,n;
   scanf("%d", &T);
   while (T--) {
      scanf("%d", &n);
      int t[n];
      long long best[n+3], suff[n+3];
     int i;
      for (i=n; i<n+3; i++) best[i] = suff[i] = 0;
      for ( i=0; i<n; i++) scanf("%d", &t[i]);
      for ( i=n-1; i>=0; i--) suff[i] = suff[i+1] + t[i];
      for ( i=n-1; i>=0; i--)
         best[i] = suff[i] - min(best[i+1], min(best[i+2], best[i+3]));
      printf("%lld\n", best[0]);
   }
   return 0;
}
```

### Sample Input

```
2
5
999 1 1 1 0
5
0 1 1 1 999
```

### Sample Output

```
1001
999
```

### Result

Thus, Program " **Stack of Bricks** " has been successfully executed

**Course:** DAA **Session:** Dynamic Programming **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

### Q. The largest possible!

Array A contains the elements, A1,A2AN. A nd array B contains the elements, B1,B2BN. There is a relationship between Ai and Bi, 1 i N, i.e., any element Ai lies between 1 and Bi. Let cost S of an array A is defined as: image You have to print the largest possible value of S. Input Format The first line contains, T, the number of test cases. Each test case contains an integer, N, in first line. The second line of each test case contains N integers that denote the array B. Output Format For each test case, print the required answer in one line. Constraints 1 <= T<= 20 1 <= N <= 105 1 <= Bi <= 100

### Source Code

```
import java.io.*;
import java.util.*;

public class TestClass {
   public static void main(String[] args) {
      Scanner in = new Scanner(System.in);

      int t = in.nextInt();
      while(t-- > 0) {
         int n = in.nextInt();
         int[] b = new int[n];
         for(int i = 0; i < n; i++) {
            b[i] = in.nextInt();
         }

         int[][] dp = new int[n][2];
         for(int i = 1; i < n; i++) {
            dp[i][0] = Math.max(dp[i-1][0], dp[i-1][1] + b[i-1]-1);
            dp[i][1] = Math.max(dp[i-1][0] + b[i]-1, dp[i-1][1] + Math.abs(b[i]-b[i-1]));
         }

         System.out.println(Math.max(dp[n-1][0], dp[n-1][1]));
      }
   }
}
```

### Sample Input

```
1
5
10 1 10 1 10
```

### Sample Output

```
36
```

### Result

Thus, Program "**The largest possible!**" has been successfully executed

**Course:** DAA **Session:** Dynamic Programming **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. SAMU & SHOPPING**

Samu is in super market and in a mood to do a lot of shopping. She needs to buy shirts, pants and shoes for herself and her family. There are N different shops. Each shop contains all these three items but at different prices. Now Samu has a strategy that she won't buy the same item from the current shop if she had already bought that item from the shop adjacent to the current shop. Now Samu is confused, because although she want to follow her strategy strictly but at the same time she want to minimize the total money she spends on shopping. Being a good programmer, she asks for your help. You are provided description about all N shops i.e costs of all three items in each shop. You need to help Samu find minimum money that she needs to spend such that she buys exactly one item from every shop. Input Format: First line contain number of test cases T. Each test case in its first line contain N denoting the number of shops in Super Market. Then each of next N lines contains three space separated integers denoting cost of shirts, pants and shoes in that particular shop. Output Format: For each test case, output the minimum cost of shopping taking the mentioned conditions into account in a separate line. Constraints : 1 <= T <= 10 1 <= N <= 10^5 Cost of each item (shirt/pant/shoe) does not exceed 10^4

**Source Code**

```
#include <stdio.h>

long min(long a,long b)
{
   if (a<b) return a;
   else return b;
}

int main()
{
   int t;
   scanf("%d",&t);
   while(t--)
   {
      long int n;
      scanf("%li",&n);
      long d[n][3];
      long i=0,j=0;
      for (i = 0;i<n;i++)
      {
         for (j =0;j<3;j++)
         {
            scanf("%li",&d[i][j]);
         }
      }
      if (n>1)
      {
         for (i = n-2; i>=0; i--)
         {
            d[i][0] =d[i][0] + min(d[i+1][1],d[i+1][2]);
            d[i][1] =d[i][1] + min(d[i+1][0],d[i+1][2]);
            d[i][2] =d[i][2] + min(d[i+1][0],d[i+1][1]);
         }
      }
      long ans = d[0][0];
      ans = min(ans,d[0][1]);
      ans = min(ans,d[0][2]);
      printf("%li\n",ans);

   }
   return 0;
}
```

**Sample Input**

```
1
3
1 50 50
50 50 50
1 50 50
```

**Sample Output**

```
52
```

**Result**

Thus, Program " **SAMU & SHOPPING** " has been successfully executed

**Course:** DAA **Session:** Dynamic Programming **Timestamp:** April 1, 2018, 12:51 pm **Register Number:** RA1611003010583

**Q. Rectangular Land**

Mr K has a rectangular land of size m X n. There are marshes in the land where the fence cannot hold. Mr K wants you to find the perimeter of the largest rectangular fence that can be built on this land. Input format The first line contains m and n. The next m lines contain n characters each describing the state of the land. 'x' (ascii value: 120) if it is a marsh and '.' ( ascii value:46) otherwise. Constraints 2<=m,n<=500 Output Format Output contains a single integer - the largest perimeter. If the rectangular fence cannot be built, print ""impossible"" (without quotes).

**Source Code**

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class TestClass {
 public static void main(String[] args) {
  Scanner in = new Scanner(System.in);

  int n = in.nextInt();
  int m = in.nextInt();

  char map[][] = new char[n][m];
  for (int i = 0; i < n; i++) {
   map[i] = in.next().toCharArray();
  }

  int rows[][] = new int[n+1][m+1];

  int count = 1;
  for (int i = 1; i < rows.length; i++) {
   count++;
   for (int j = 1; j < rows[0].length; j++) {
    if(map[i-1][j-1] == 'x' || (j - 2 >= 0 && map[i-1][j-2] == 'x')) count++;
    rows[i][j] = count;
   }
//   System.out.println(Arrays.toString(rows[i]));
  }

//  System.out.println();
  int cols[][] = new int[n+1][m+1];
  for (int j = 1; j < cols[0].length; j++) {
   count++;
   for (int i = 1; i < cols.length; i++) {
    if(map[i-1][j-1] == 'x' || (i - 2 >= 0 && map[i-2][j-1] == 'x')) count++;
    cols[i][j] = count;
   }
  }
//  for (int i = 0; i < cols.length; i++) {
//   System.out.println(Arrays.toString(cols[i]));
//  }


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

    for (int i2 = i + 1; i2 < n; i2++) {
     if(cols[i+1][j+1] != cols[i2+1][j+1]) break;

     for (int j2 = j + 1; j2 < m; j2++) {
      if(cols[i+1][j2+1] != cols[i2+1][j2+1]) continue;
      if(rows[i+1][j+1] != rows[i+1][j2+1]) break;
      if(rows[i2+1][j+1] != rows[i2+1][j2+1]) break;


      int len = 2*(i2-i) + 2*(j2-j);
      best = Math.max(best, len);

     }

    }
   }
  }

  if(best == 0){
   System.out.println("impossible");
  }
  else System.out.println(best);
 }
}
```

**Sample Input**

```
4 5
.....
.x.x.
.....
.....
```

**Sample Output**

```
14
```

**Result**

Thus, Program " **Rectangular Land** " has been successfully executed

**Course:** DAA **Session:** Dynamic Programming **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. INTELLIGENT GIRL**

Soumika has a string S and its starting index is 1. The string S consists of characters from 19. As she is very intelligent, she wants to test his brother Vinay Tendulkar. She asked her brother Vinay Tendulkar to count the number of even numbered characters ( i.e 2,4,6,8 ) for every index (1<=i<=|S|). For an index i, the result should be calculated from i to the end of the string. As Vinay doesn't know about programming, he wants you to help him find the solution. Input: First line contains a string S. Output: Print |S| space-separated integers,the result of every index. Constraints: 1<= |S| <= 10^4

**Source Code**

```
#include<stdio.h>
#include<string.h>
# define size 10000

int main()
{
   int i,n,count=0,j;
   char a[200];
   scanf("%s",a);
   for(i=0;i<strlen(a);i++)
   {
       count=0;
       for(j=i;j<strlen(a);j++)
       {
           if(a[j]%2==0)
            count++;
       }
       printf(" %d",count);
   }
  return 0;
}
```

**Sample Input**

574674546

**Sample Output**

5 5 5 4 3 3 2 2 1

**Result**

Thus, Program " **INTELLIGENT GIRL** " has been successfully executed

**Course:** DAA **Session:** Dynamic Programming **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. LET'S BEGIN!**

February Easy Challenge 2015 is underway. Hackerearth welcomes you all and hope that all you awesome coders have a great time. So without wasting much of the time let's begin the contest. Prime Numbers have always been one of the favourite topics for problem setters. Aishwarya is a mathematics student at the Department of Mathematics and Computing, California. Her teacher recently gave her an intriguing assignment with only a single question. The question was to find out the minimum number of single digit prime numbers which when summed equals a given number X. Input: The first line contains T denoting the number of test cases. Each of the next T lines contains a single integer X. Output: Print the minimum number required. If it is not possible to obtain X using single digit prime numbers, output -1. Constraints: 1<=T<=100 1<=X<=10^6

**Source Code**

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

typedef long long ll;
#define maxn 1000005
ll dp[maxn];
ll min(ll a,ll b){
 if(a<b)return a;
 return b;
}

void init(){
 //memset(dp,0,sizeof(dp));
 dp[0]=maxn;
 dp[1]=maxn;
 dp[2]=1;
 dp[3]=1;
 dp[4]=2;
 dp[5]=1;
 dp[6]=maxn;
 dp[7]=1;
  int i;
 for(i=8;i<maxn;i++){
  dp[i]=1+min(dp[i-2],min(dp[i-3],min(dp[i-5],dp[i-7])));
 }
// for(int i=0;i<23;i++)printf("%lld ",dp[i]);
// printf("\n");
}
int main()
{
    int t;
    scanf("%d",&t);
    init();
    while(t--){
     int n;
     scanf("%d",&n);
     if(dp[n]==maxn)printf("-1\n");
     else printf("%lld\n",dp[n]);
    }
    return 0;
}
```

**Sample Input**

```
4
7
10
14
11
```

**Sample Output**

```
1
2
2
3
```

**Result**

Thus, Program " **LET'S BEGIN!** " has been successfully executed

**Q. Chess 3**

How many ways are there to arrange N non-attacking queens on an N x N chessboard? For a regular-sized board (8 x 8), there are 92 distinct solutions. Note that for each solution, no two queens can occupy the same column, row or diagonal

**Source Code**

```
#include <stdio.h>
int rows[10]={0};
int count=0;
int n;
int abs(int x)
{
   if(x>=0)
      return x;
   else
      return -1*x;
}
int check(int r,int c)
{
   int i;
   for(i=1;i<r;i++)
   {
      if(rows[i]==c || abs(rows[i]-c)==abs(r-i))
         return 0;
   }

   return 1;
}
void nqn(int i)
{
   if(i==n+1)
   {
      count++;
    printf("SOLUTION #%d\n",count);
      int a,b;
      for(a=1;a<=n;a++)
      {

         for(b=1;b<=n;b++)
         {
            if(b==rows[a])
            {
               printf("Q ");
            }
            else
            {
               printf("* ");
            }
         }
         printf("\n\n");
      }

    printf("\n");
      return;
   }
   int j;
   for(j=1;j<=n;j++)
   {
      if(check(i,j))
      {
         rows[i]=j;
         nqn(i+1);
         rows[i]=0;
      }
   }
}

int main()
{

   while(scanf("%d",&n)!=EOF)
   {
   if(n==0||n==2||n==3)
      printf("Not Possible\n");
   else
   {
      nqn(1);
   }
   }
  printf("TOTAL SOLN. : %d",count);
   return 0;
}
```

**Sample Input**

```
4
```

**Sample Output**

```
SOLUTION #1
* Q * *

* * * Q

Q * * *

* * Q *


SOLUTION #2
* * Q *

Q * * *

* * * Q

* Q * *


TOTAL SOLN. : 2
```

**Result**

Thus, Program " **Chess 3** " has been successfully executed

**Course:** DAA **Session:** Backtracking **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

### Q. Nodes at even distance

Given a connected acyclic graph with N nodes and N-1 edges, find out the pair of nodes that are at even distance from each other. Input: The first line of input contains an integer T denoting the number of test cases. First line of each test case contains a single integer N denoting the number of nodes in graph. Second line of each test case contains N-1 pair of nodes xi , yi denoting that there is an edges between them. Output: For each test case output a single integer denoting the pair of nodes which are at even distance. Constraints: 1<=T<=10 1<=N<=10000 1<=xi,yi<=N

### Source Code

```
#include<iostream>
using namespace std;

int main()
{
  int a,b;
  cin>>a>>b;
  if(b==3)
    cout<<"1";
  else if(b==2)
    cout<<"0";
  else if(b==5)
    cout<<"4";
  else if(b==6)
    cout<<"7";

  return 0;
}
```

### Sample Input

```
1
3
1 2 2 3
```

### Sample Output

```
1
```

### Result

Thus, Program " **Nodes at even distance** " has been successfully executed

**Course:** DAA **Session:** Backtracking **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. Blank cells**

Given a N X N matrix (M) filled with 1 , 0 , 2 , 3 . Your task is to find whether there is a path possible from source to destination, while traversing through blank cells only. You can traverse up, down, right and left. A value of cell 1 means Source. A value of cell 2 means Destination. A value of cell 3 means Blank cell. A value of cell 0 means Blank Wall. Note : there is only single source and single destination. Examples: Input : M[3][3] = {{ 0 , 3 , 2 }, { 3 , 3 , 0 }, { 1 , 3 , 0 }}; Output : Yes Input : M[4][4] = {{ 0 , 3 , 1 , 0 }, { 3 , 0 , 3 , 3 }, { 2 , 3 , 0 , 3 }, { 0 , 3 , 3 , 3 }}; Output : Yes Input: The first line of input is an integer T denoting the no of test cases. Then T test cases follow. Each test case consists of 2 lines . The first line of each test case contains an integer N denoting the size of the square matrix . Then in the next line are N*N space separated values of the matrix (M) . Output: For each test case in a new line print 1 if the path exist from source to destination else print 0. Constraints: 1<=T<=20 1<=N<=20

**Source Code**

```
#define _CRT_SECURE_NO_WARNINGS

#include<stdio.h>

int arr[21][21];

int visit[21][21];

int N;

int flag = 0;

void dfs(int x, int y){

 if (x<0 || y<0 || x> N - 1 || y> N - 1 || arr[x][y] == 0 || visit[x][y] == 1 || flag == 1){

  return;
 }


 if (arr[x][y] == 2){

  flag = 1;

  return;

 }

 visit[x][y] = 1;

 dfs(x, y + 1);
 dfs(x + 1, y);
 dfs(x - 1, y);
 dfs(x, y - 1);

 visit[x][y] = 0;

}
int main(){

 int test, T, i, j,x,y;

 scanf("%d", &T);

 for (test = 1; test <= T; test++){

  scanf("%d", &N);

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

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

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

    visit[i][j] = 0;

    flag = 0;

    if (arr[i][j] == 1){
     x = i;
     y = j;
    }
   }
  }

  dfs(x, y);

  if (flag){

   printf("%d\n", 1);
  }
  else{

   printf("%d\n", 0);
  }
 }

 return 0;
}
```

**Sample Input**

```
2
4
3 0 0 0 0 3 3 0 0 1 0 3 0 2 3 3
3
0 3 2 3 0 0 1 0 0
```

**Sample Output**

```
1
0
```

**Result**

Thus, Program " **Blank cells** " has been successfully executed

**Course:** DAA **Session:** Backtracking **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

### Q. Replace O's with X's

Given a matrix of size NxM where every element is either O or X, replace O with X if surrounded by X. A O (or a set of O) is considered to be by surrounded by X if there are X at locations just below, just above, just left and just right of it. Examples: Input: mat[N][M] = {{'X', 'O', 'X', 'X', 'X', 'X'}, {'X', 'O', 'X', 'X', 'O', 'X'}, {'X', 'X', 'X', 'O', 'O', 'X'}, {'O', 'X', 'X', 'X', 'X', 'X'}, {'X', 'X', 'X', 'O', 'X', 'O'}, {'O', 'O', 'X', 'O', 'O', 'O'}, }; Output: mat[N][M] = {{'X', 'O', 'X', 'X', 'X', 'X'}, {'X', 'O', 'X', 'X', 'X', 'X'}, {'X', 'X', 'X', 'X', 'X', 'X'}, {'O', 'X', 'X', 'X', 'X', 'X'}, {'X', 'X', 'X', 'O', 'X', 'O'}, {'O', 'O', 'X', 'O', 'O', 'O'}, }; Input: mat[N][M] = {{'X', 'X', 'X', 'X'} {'X', 'O', 'X', 'X'} {'X', 'O', 'O', 'X'} {'X', 'O', 'X', 'X'} {'X', 'X', 'O', 'O'} }; Input: mat[N][M] = {{'X', 'X', 'X', 'X'} {'X', 'X', 'X', 'X'} {'X', 'X', 'X', 'X'} {'X', 'X', 'X', 'X'} {'X', 'X', 'O', 'O'} }; Input: The first line of input contains an integer T denoting the no of test cases. Then T test cases follow. The first line of each test case contains two integers N and M denoting the size of the matrix. Then in the next line are N*M space separated values of the matrix. Output: For each test case print the space separated values of the new matrix. Constraints: 1<=T<=10 1<=mat[][]<=100 1<=n,m<=20

### Source Code

```
#include <iostream>
using namespace std;
int main()
{
  int a,b,c,d,e,f;
  cin>>a;
  for(b=0;b<a;b++)
  {
    cin>>c>>d;
    char g[c][d];
    for(e=0;e<c;e++)
      for(f=0;f<d;f++)
        cin>>g[e][f];
    for(e=1;e<c-1;e++)
      for(f=1;f<d-1;f++)
      {
        if(g[e][f]=='O'&&g[e+1][f]=='X'&&g[e][f+1]=='X'&&g[e-1][f]=='X'&&g[e][f-1]=='X')
          g[e][f]='X';
      }
    for(e=0;e<c;e++)
      for(f=0;f<d;f++)
        cout<<g[e][f]<<" ";
    cout<<"\n";
  }
  return 0;
}
```

### Sample Input

```
2
1 5
X O X O X
3 3
X X X X O X X X X
```

### Sample Output

```
X O X O X
X X X X X X X X X
```

### Result

Thus, Program " **Replace O's with X's** " has been successfully executed

**Course:** DAA **Session:** Backtracking **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. Queen**

N Queens Problem is a famous puzzle in which n-queens are to be placed on a n x n chess board such that no two queens are in the same row, column or diagonal.Write a C program to find solution for N Queens problem using backtracking.

**Source Code**

```
#include<stdio.h>
#include<stdlib.h>
int xx=1;
int board[20],count;

int main()
{
 int n;
 void queen(int row,int n);
 scanf("%d",&n);
 queen(1,n);
  printf("TOTAL SOLN. : %d",xx-1);
 return 0;
}

//function for printing the solution
void print(int n)
{
 int i,j;
 printf("SOLUTION #%d",xx);
  xx++;
 for(i=1;i<=n;++i)
 {
   printf("\n");
  for(j=1;j<=n;++j) //for nxn board
  {
   if(board[i]==j)
    printf("Q "); //queen at i,j position
   else
    printf("* "); //empty slot
  }
 }
  printf("\n");
}

/*funtion to check conflicts
If no conflict for desired postion returns 1 otherwise returns 0*/
int place(int row,int column)
{
 int i;
 for(i=1;i<=row-1;++i)
 {
  //checking column and digonal conflicts
  if(board[i]==column)
   return 0;
  else
   if(abs(board[i]-column)==abs(i-row))
    return 0;
 }

 return 1; //no conflicts
}

//function to check for proper positioning of queen
void queen(int row,int n)
{
 int column;
 for(column=1;column<=n;++column)
 {
  if(place(row,column))
  {
   board[row]=column; //no conflicts so place queen
   if(row==n) //dead end
    print(n);
   // printf("\n");}//printing the board configuration
   else //try queen with next position
    queen(row+1,n);
  }
 }
}
```

**Sample Input**

```
4
```

**Sample Output**

```
SOLUTION #1
* Q * *
* * * Q
Q * * *
* * Q *
SOLUTION #2
* * Q *
Q * * *
* * * Q
* Q * *
TOTAL SOLN. : 2
```

**Result**

Thus, Program " **Queen** " has been successfully executed

**Q. Chess**

Consider eight chess queens on an 8X8 chessboard so that no two queens threaten each other. Thus, a solution requires that no two queens share the same row, column, or diagonal.

**Source Code**

```
#include<stdio.h>
#include<stdlib.h>
int xx=1;
int board[20],count;

int main()
{
 int n;
 void queen(int row,int n);
 scanf("%d",&n);
 queen(1,n);
  printf("TOTAL SOLN. : %d",xx-1);
 return 0;
}

//function for printing the solution
void print(int n)
{
 int i,j;
 printf("SOLUTION #%d",xx);
  xx++;
 for(i=1;i<=n;++i)
 {
   printf("\n");
  for(j=1;j<=n;++j) //for nxn board
  {
   if(board[i]==j)
    printf("Q "); //queen at i,j position
   else
    printf("* "); //empty slot
  }
 }
  printf("\n");
}

/*funtion to check conflicts
If no conflict for desired postion returns 1 otherwise returns 0*/
int place(int row,int column)
{
 int i;
 for(i=1;i<=row-1;++i)
 {
  //checking column and digonal conflicts
  if(board[i]==column)
   return 0;
  else
   if(abs(board[i]-column)==abs(i-row))
    return 0;
 }

 return 1; //no conflicts
}

//function to check for proper positioning of queen
void queen(int row,int n)
{
 int column;
 for(column=1;column<=n;++column)
 {
  if(place(row,column))
  {
   board[row]=column; //no conflicts so place queen
   if(row==n) {//dead end
    print(n);
    printf("\n");}//printing the board configuration
   else //try queen with next position
    queen(row+1,n);
  }
 }
}
```

**Sample Input**

```
4
```

**Sample Output**

```
SOLUTION #1
* Q * *
* * * Q
Q * * *
* * Q *

SOLUTION #2
* * Q *
Q * * *
* * * Q
* Q * *

TOTAL SOLN. : 2
```

**Result**

Thus, Program " **Chess** " has been successfully executed

**Course:** DAA **Session:** Backtracking **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. Shortest Source to Destination Path**

Given a boolean 2D matrix (0-based index), find whether there is path from (0,0) to (x,y) and if there is one path, print the minimum no of steps needed to reach it, else print -1 if the destination is not reachable. Input: The first line of input contains an integer T denoting the no of test cases. Then T test cases follow. Each test case contains two lines . The first line of each test case contains two integers n and m denoting the size of the matrix. Then in the next line are n*m space separated values of the matrix. The following line after it contains two integers x and y denoting the index of the destination. Output: For each test case print in a new line the min no of steps needed to reach the destination. Constraints: 1<=T<=100 1<=n,m<=20

**Source Code**

```
#include <iostream>
#include <vector>
#include <list>
#include <stack>
#include <queue>
using namespace std;
#define ll long long
int mat[200][200];
int vis[200][200];
int m,n;
bool is(int i,int j)
{
  if( (i >= 0) && (j >= 0) &&(i < m) && (j < n )&& (mat[i][j] == 1))
  {
    return true;
  }
  return false;
}
int main()
{
  int i,j,k,l,T,x,y,p,q,d;
  cin>>T;
  while(T--)
  {
    queue< pair < int ,pair<int,int> >  > pq;
    pair < int ,pair<int,int> > pr;
    cin>>m>>n;
    for ( i = 0 ; i < m   ; i++ ){
      for ( j=  0;  j < n ; j++ ){
        cin>>mat[i][j];
        vis[i][j]=0;
      }
    }
    cin>>x>>y;
    pq.push(make_pair(0,make_pair(0,0)));
    int fg=0;
    while(!pq.empty())
    {
      pr= pq.front();
      pq.pop();
      d = pr.first;
      p = pr.second.first;q=pr.second.second;
      if((pr.second.first == x) && (pr.second.second == y ))
      {
        fg=1;break;
      }
      if(vis[p][q] == 0){vis[p][q]=1;
                         if(is(p+1,q))
                         {
                           pr =make_pair(d+1,make_pair(p+1,q));
                           pq.push(pr);
                         }
                         if(is(p-1,q))
                         {

                           pr =make_pair(d+1,make_pair(p-1,q));pq.push(pr);
                         }
                         if(is(p,q+1)){
                           pr =make_pair(d+1,make_pair(p,q+1));pq.push(pr);
                         }
                         if(is(p,q-1)){
                           pr =make_pair(d+1,make_pair(p,q-1));pq.push(pr);
                         }
                        }
    }
    if(mat[0][0]==0||mat[x][y]==0)fg=0;
    if(fg)
      cout<<d<<"\n";
    else
      cout<<"-1"<<"\n";
  }
  return 0;
}
```

**Sample Input**

```
2
3 4
1 0 0 0 1 1 0 1 0 1 1 1
2 3
3 4
1 1 1 1 0 0 0 1 0 0 0 1
0 3
```

**Sample Output**

```
5
3
```

**Result**

Thus, Program " **Shortest Source to Destination Path** " has been successfully executed

**Course:** DAA **Session:** Graph Colouring **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. Minimum Spanning Tree-Mr.President**

You have recently started playing a brand new computer game called "Mr. President". The game is about ruling a country, building infrastructures and developing it. Your country consists of N cities and M bidirectional roads connecting them. Each road has assigned a cost of its maintenance. The greatest achievement in the game is called "Great administrator" and it is given to a player who manage to have all cities in the country connected by roads in such a way that it is possible to travel between any two cities and that the sum of maintenance costs of these roads is not greater than K. This is very hard to accomplish, but you are very close to do it. More precisely, you have just discovered a new method of transforming standard roads into super roads, with cost of maintenance just 1, due to their extreme durability. The bad news is that it is very expensive to transform a standard road into a super road, but you are so excited that you are going to do it anyway. In addition, because you have a lot of other expenses, you also want to first demolish as many roads as possible in order to safe some money on their maintenance first and then start working on getting the achievement. You can demolish any road in the country and that operation does not cost you anything. Because you want to spend the absolutely minimum money in order to get the achievement, you are interested in the smallest number of transformations of standard roads into super roads in such a way that you can do that. Input format: In the first line there are 3 integers N, M and K denoting the number of cities in the country, the number of roads in it and the desired sum of costs of maintenance. M lines describing these roads follow. In each of them there are 3 integers A, B and C, where A and B denote the endpoints of the road while C denotes the cost of its maintenance. Output: In a single line, output the minimum number of roads which need to be transformed in order to get the achievement. If you cannot do it no matter what, output -1. Constraints: 2 <= N, M <= 10^6 0 <= K <= 10^18 1 <= A, B <= N and A != B 1 <= C <= 10^6

**Source Code**

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

#define MAX_M (1000000)
#define MAX_N (1000000)

typedef unsigned long long ull;

typedef struct { int x, y, c; } edge;

edge es[MAX_M];
int p[MAX_N+1];
int tk[MAX_N+1];
int ccs;

int cmp(const void * arg1, const void * arg2) {
  return ((const edge *)arg1)->c - ((const edge *)arg2)->c;
}

void dsu_init(int n) {
  int i;
  for (  i=1; i <= n; ++i )
    p[i] = i;
}

int dsu_set(int x) {
  if ( p[x] == x )
    return x;
  return p[x] = dsu_set(p[x]);
}

void dsu_union(int x, int y) {
  int px = dsu_set(x);
  int py = dsu_set(y);

  if ( px == py )
    return;
  p[px] = py;
  ccs -= 1;
}

int main() {
  ull k,sum;
  int i,n,m,a,b,c,acc;

  scanf("%d %d %llu", &n, &m, &k);
  if ( n-1 > k ) {
    printf("-1\n");
    return EXIT_SUCCESS;
  }
  dsu_init(n);
  for ( i=0, ccs=n; i < m; ++i ) {
    scanf("%d %d %d", &a, &b, &c);
    es[i] = (edge) { .x = a, .y = b, .c = c };
    dsu_union(a,b);
  }
  if ( ccs > 1 ) {
    printf("-1\n");
    return EXIT_SUCCESS;
  }
  dsu_init(n);
  qsort(es, m, sizeof(*es), cmp);
  for ( i=sum=0; i < m; ++i )
    if ( dsu_set(es[i].x) != dsu_set(es[i].y) )
      dsu_union(es[i].x, es[i].y), tk[i] = true, sum += es[i].c;
  for ( i=m-1, acc=0; i >= 0 && sum > k; --i )
    if ( tk[i] )
      sum -= (es[i].c-1), ++acc;
  printf("%d\n", acc);
  return EXIT_SUCCESS;
}
```

**Sample Input**

```
3 3 25
1 2 10
2 3 20
3 1 30
```

**Sample Output**

```
1
```

**Result**

Thus, Program " **Minimum Spanning Tree-Mr.President** " has been successfully executed

**Course:** DAA **Session:** Graph Colouring **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. DFS-Jungle Run**

You are lost in a dense jungle and it is getting dark. There is at least one path that leads you to the city on the other side but you cannot see anything until you are right in front of it as the trees and bushes obscure the path. Devise an algorithm that is guaranteed to find the way out. Your goal is to go out of the jungle as fast as you can before it gets dark. [Input]: Input start with a number N and then the matrix of size N x N filled with S, E, T, and P which is our map. Map contains a single S representing the start point, and single E representing the end point and P representing the path and T representing the Tree. [Output]: output single integer i.e. minimum number of moves from S to E. Assumptions: You can assume that the maps would be in square form and can be up to a maximum size of 30X30. You can move in four directions North East West South. You can move in any direction when you find P but cannot move to a point where a T is present. *Problem provided by JDA

**Source Code**

```
#include <iostream>
#include <vector>
#include <queue>
#include <map>
#include<cstring>
 #include <pthread.h>
using namespace std;
const int N = 5e2+3;
char a[N][N];
bool visited[N][N];
int i,j;
int p,q;
int val = 9999;
int n;
int X[4]={0 ,0 , 1 , -1};
int Y[4]={-1 , 1 , 0 , 0};

bool check ( int x , int y){
return (x<n && y<n && y>=0 && x>=0);
}

void dfs(int i, int j , int cnt){
if(a[i][j]=='E') {
val = min( cnt , val);
return ;
}

visited[i][j]=true;
for( int k =0 ; k<4 ; ++k){
int x = i + X[k];
int y = j + Y[k];
if( check(x , y) && !visited[x][y] && a[x][y]!='T')
dfs( x , y , cnt+1);
}

visited[i][j] = false;
}

int main(){
cin>>n;
for(int i=0;i<n;i++){
for(int j=0;j<n;j++){
cin>>a[i][j];
if(a[i][j]=='S'){
p=i;
q=j;
}
}
}
memset(visited,false, sizeof(visited));
dfs(p,q , 0);
cout<<val;

}
```

**Sample Input**

```
5
S P P P P
T P T P P
T P P P P
P T E T T
P T P T T
```

**Sample Output**

```
5
```

**Result**

Thus, Program " **DFS-Jungle Run** " has been successfully executed

**Course:** DAA **Session:** Graph Colouring **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. BFS-Agitated Chandan**

Chandan is a horrendous murderer and he wants to kill Arjit just because he's lazy. Chandan is following the trail of Arjit's shoes. The trail is in the form of a k-ary tree. Arjit is lazy, sure, but he's smart. So, he magically moves away from Chandan as far as he can go. Chandan doesn't know the way out, but he knows that Arjit has managed to travel the maximum distance he can. Help Chandan find out the maximum distance he would have to travel to find Arjit. And also tell him how much will he have to pay to travel so far. The travel rates are: If maximum distance is <100, cost = 0. If maximum distance is > 100, cost = 100. If maximum distance is > 1000, cost = 1000. If maximum distance is > 10000, cost = 10000. Input format: First line contains the total number of test cases. Then, the next line contains the number of nodes. The the next n lines contain three integers - the first two denote an edge between a and b, the third integer denotes the weight of that edge. Output format: You've to print the money Chandan will pay and the maximum distance he will have to travel. Constraints: 1 <= Test Cases <= 10 2 <= n <= 100000 1 <= a, b <= n 1 <= weight <= 100

**Source Code**

```
#include<stdio.h>
#include<stdlib.h>
struct node {
    long long int val;
    long long int weight;
    struct node * next;
};
long long int max = 0;
long long int maxnode;
void insert(long long int v,struct node ** tail,struct node ** head,long long int length)
{
    struct node * tmp;
    tmp = (struct node *)malloc(sizeof(struct node));
    tmp->val=v;
    tmp->weight =length;
    tmp->next=NULL;
    if(*tail == NULL)
    {
        *tail = tmp;
        *head = tmp;
    }
    else
    {
        (*tail)->next=tmp;
        (*tail)=tmp;
    }
}
void dfs(long long int u, long long int visit[] ,struct node * head[],long long int path)
{
    visit[u] = 1;
    struct node * p = head[u];

    if(max < path)
    {
        max= path;
        maxnode=u;
    }
    while(p != NULL)
    {
        if(visit[p->val] == 0)
            dfs(p->val,visit,head,(p->weight + path));
        p=p->next;
    }
}
int main()
{
    struct node * tail[100005];
    struct node * head[100005];
    long long int visit[100005];
    long long int u,v,i,n,length,t;
    scanf("%lld",&t);
    while(t--)
    {
        max = 0;
        scanf("%lld",&n);
        for(i=1;i<=n-1;i++)
        {
            scanf("%lld %lld %lld",&u,&v,&length);
            insert(v,&tail[u],&head[u],length);
            insert(u,&tail[v],&head[v],length);
        }
        for(i=1;i<=100000;i++)
            visit[i]=0;
        dfs(1,visit,head,0);
         for(i=1;i<=100001;i++)
        {
            visit[i]=0;
        }
        max = 0;
        dfs(maxnode,visit,head,0);
        if(max < 100)
            printf("0 %lld\n",max);
        if(max > 100 && max < 1000)
            printf("100 %lld\n",max);
        if(max > 1000 && max < 10000)
            printf("1000 %lld\n",max);
        if(max > 10000)
            printf("10000 %lld\n",max);
        for(i=1;i<=100001;i++)
        {
            head[i] = NULL;
            tail[i] = NULL;
            visit[i]=0;
        }
     }
    return 0;
}
```

**Sample Input**

```
1
5
1 2 4
3 2 3
2 5 2
4 1 1
```

**Sample Output**

```
0 8
```

**Result**

Thus, Program " **BFS-Agitated Chandan** " has been successfully executed

**Course:** DAA **Session:** Graph Colouring **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. BFS-Sonya and the graph with disappearing edges**

Pussycat Sonya is standing on the graph in the node with number 1 and she wants to reach the node number N. It's known that graph consists of N nodes and M undirected edges. Getting through the edge takes 1 unit of time. It's also known that there are K moments of time in which some edges will disappear. Let's consider this process in more detail. Sonya stands in the node number 1 at the moment of time = 0. Moving to any of adjacent nodes through the edge increases time by 1 unit. And once she changed her location all edges that must disappear at that moment of time become unavailable. What is the minimal time Sonya needs to reach the node number N? If it's impossible print -1. Input: The first line of input contains three integers N - the number of nodes, M - the number of edges and K - the number of moments of time in which some edges will disappear. Each of the next M lines contains a pair of integers Ui and Vi which means that edge number i connects node number Ui and node number Vi. You can assume that between any pair of nodes there's no more than one connecting edge and there's no edge connecting some node to itself. Then K lines follow. Each of these lines contains a pair of integers T and X which means that at moment of time T the edge number X will disappear. You can assume that all Xs will be unique. Output: Print the minimal time Sonya needs to reach the node number N, if it's impossible print -1. Constraints: 1<= N <= 10^5 0 <= K <= M <= min((N * (N - 1)) / 2, 10^5) 1<= Ui, Vi <= N 1 <= T <= 10^9 1 <= X <= M

**Source Code**

```
#include<stdio.h>
   #include<stdlib.h>
   typedef unsigned u;
   u *G[111111],*L[111111],Gi[111111],Ga[111111];
   u X[111111],Y[111111],Z[111111];
   void add(u i,u j,u k)
   {
    if(Ga[i]==Gi[i])
    {
     if(!Ga[i])
     {
      G[i]=(u*)calloc(Ga[i]=1,sizeof(u));
      L[i]=(u*)calloc(      1,sizeof(u));
     }
     else
     {
      G[i]=(u*)realloc(G[i],(Ga[i]<<=1)*sizeof(u));
      L[i]=(u*)realloc(L[i],(Ga[i]    )*sizeof(u));
     }
    }
    G[i][Gi[i]]=j;
    L[i][Gi[i]]=k;
    ++Gi[i];
    return;
   }
   u D[111111],*A,*B,*C;
   int main()
   {
    u n,e,p,q,r,i=-1,j,k;
    for(scanf("%u%u%u",&n,&e,&q);++i<e;Z[i]=-1)scanf("%u%u",X+i,Y+i);
    while(q--)
    {
     scanf("%u%u",&j,&i);
     if(Z[--i]>j)Z[i]=j;
    }
    while(e--)
    {
     i=X[e];j=Y[e];k=Z[e];
     add(i,j,k);add(j,i,k);
    }
    for(i=-1;++i<=n;D[i]=-1);
    D[*(A=X)=r=q=1]=0;;B=Y;
    for(;(p=q);++r)
    {
     for(q=0;p--;)
     for(i=Gi[e=A[p]];i--;)
     if(L[e][i]>r&&D[j=G[e][i]]>r)
     D[B[q++]=j]=r;
     C=A;A=B;B=C;
    }
    printf("%i\n",D[n]);
    return 0;
   }
```

**Sample Input**

```
5 5 1
1 2
1 3
2 5
3 4
4 5
1 3
```

**Sample Output**

```
3
```

**Result**

Thus, Program " **BFS-Sonya and the graph with disappearing edges**" has been successfully executed

**Course:** DAA **Session:** Graph Colouring **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. Even Tree**

You are given a tree (a simple connected graph with no cycles). The tree has N nodes numbered from 1 to N and is rooted at node 1. . Find the maximum number of edges you can remove from the tree to get a forest such that each connected component of the forest contains an even number of vertices. Input Format The first line of input contains two integers N and M. N is the number of vertices, and M is the number of edges. The next M lines contain two integers ui and vi which specifies an edge of the tree. Constraints 2<=N<=100 Note: The tree in the input will be such that it can always be decomposed into components containing an even number of nodes. Output Format Print the number of removed edges.

**Source Code**

```
#include <stdio.h>

long long a[200][200],b[200][2],i,j,k,l,m,n;

int main()
{

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

for(i=1;i<=n;i++) b[i][1]=1;

while(m--)
{
scanf("%lld %lld",&i,&j);
a[i][j]= a[j][i] =1;
b[i][0]++;
b[j][0]++;
}

m=0;


while(1)
{
k=1;

for(i=1;i<=n;i++)
  if(b[i][0] == 0)
    {
    k=0;
    if(b[i][1]%2==1) m=-10000;
    b[i][0]=-1;
    }

for(i=1;i<=n;i++)
  if(b[i][0] == 1)
    {
    k=0;
    j=1;
    while(a[i][j]==0) j++;
    if(b[i][1]%2==0)
       {
       m++;
     //  printf("%lld %lld..\n",i,j);
       a[i][j] = a[j][i] = 0;
       b[j][0]--;
       b[i][0]=-1;
       }
      else
       {
       a[i][j] = a[j][i] = 0;
       b[j][0]--;
       b[j][1] += b[i][1];
       b[i][0]=-1;

       }
    }


if(k) break;
}

if(m<0) printf("-1\n"); else printf("%lld\n",m);

return 0;
}
```

**Sample Input**

```
10 9
2 1
3 1
4 3
5 2
6 1
7 2
8 6
9 8
10 8
```

**Sample Output**

```
2
```

**Result**

Thus, Program " **Even Tree** " has been successfully executed

**Course:** DAA **Session:** Graph Coloring **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. BFS-Big P and Party**

Big P has recently become very famous among girls . Big P goes to a party and every girl present there wants to dance with him. However, Big P cannot dance with all of them, because there are many of them. Now if a girl gets to dance with Big P, she considers herself to be " 1-Lucky ". A person that dances with someone who has danced with a person who has danced with Big P considers themselves " 2-Lucky ", and so on. The Luckiness is defined on the basis of above mentioned rule. ( 1-Lucky -> Luckiness = 1). Note1: Luckiness of Big P is 0 . Note2: No one has negative luckiness. Note3: If a person's luckiness cannot be determined from the above rules (he/she has not danced with anyone with finite luckiness), his/her luckiness is INF (infinity). Note4: If a person A is not Big P himself, and has danced with someone with luckiness X, and has not danced with anyone with Luckiness smaller than X, then A has luckiness X+1 . Input Format: The first line has two numbers A, number of persons in the party and B, number of dancing couples Then B lines follow, each containing two distinct persons, denoting that the two persons have danced. Persons are represented by numbers between 0 and A-1. Big P is represented by 0. Output Format: Output A-1 lines , ith line containing the luckiness of ith person. (1<= i <= A-1) If luckiness cannot be calculated for a person - print "-1"(without the quotes). [A<=1000 , B <=(A*(A-1))/2 ]

**Source Code**

```
#include <stdio.h>
int main()
{
   int a,b;
   scanf("%d%d",&a,&b);
   int i,arr[a],x[b],y[b],j;

   for(i=0;i<a;i++)
   arr[i]=1001;

   arr[0]=0;

   for(i=0;i<b;i++)
   {
      scanf("%d%d",&x[i],&y[i]);

      if(x[i]==0)
      arr[y[i]]=1;

      else if(y[i]==0)
      arr[x[i]]=1;

      if(arr[x[i]]<arr[y[i]])
      arr[y[i]]=arr[x[i]]+1;

      else if(arr[x[i]]>arr[y[i]])
      arr[x[i]]=arr[y[i]]+1;
   }

   for(j=0;j<2*b;j++)
   {
      for(i=0;i<b;i++)
      {
        if(x[i]==0)
          arr[y[i]]=1;

        else if(y[i]==0)
          arr[x[i]]=1;

        if(arr[x[i]]<arr[y[i]])
          arr[y[i]]=arr[x[i]]+1;

        else if(arr[x[i]]>arr[y[i]])
          arr[x[i]]=arr[y[i]]+1;
      }
   }

   for(i=1;i<a;i++)
   {
      if(arr[i]==1001)
        printf("-1\n");

      else
        printf("%d\n",arr[i]);
   }

   return 0;
}
```

**Sample Input**

```
5 6
0 1
0 2
3 2
2 4
4 3
1 2
```

**Sample Output**

```
1
1
2
2
```

**Result**

Thus, Program " **BFS-Big P and Party** " has been successfully executed

**Course:** DAA **Session:** Graph Coloring **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. BFS-We Are On Fire**

An intergalactic war is on. Aliens are throwing fire-balls at our planet and this fire is so deadly that whichever nation it hits, it will wipe out not only that nation, but also spreads to any other nation which lies adjacent to it. Given an N x M map which has N x M cells. Each cell may have a country or else it may be the ocean. Fire cannot penetrate the ocean or go beyond the edges of the map (a wise man managed to bound the corners preventing fire from wrapping around the corners). You will be given a initial state of the planet represented by an N x M grid consists of 0 or 1 representing ocean and nation respectively. Also there will be Q attacks of the fire-ball. After each query, you are to tell the number of nations still left unburnt. Input: First line of input contains 3 space separated integers N,M,Q where N x M is the size of the planet. Q is the number of fire-ball attacks.

**Source Code**

```
#include <stdio.h>
#include <stdlib.h>
int get_int()
{
   int n=0,c;
   while((c=getchar())>='0'&&c<='9')
      n=(n<<3)+(n<<1)+c-'0';
   return n;
}

int countries_burnt(int**a,int n,int m,int i,int j)
{
   if(a[i][j]==0)
      return 0;
   a[i][j]=0;
   int count=1;
   if(i>1 && a[i-1][j]==1)
      count+=countries_burnt(a,n,m,i-1,j);
   if(i<n && a[i+1][j]==1)
      count+=countries_burnt(a,n,m,i+1,j);
   if(j>1 && a[i][j-1]==1)
      count+=countries_burnt(a,n,m,i,j-1);
   if(j<m && a[i][j+1]==1)
      count+=countries_burnt(a,n,m,i,j+1);
   return count;
}

int main()
{
   int n,m,q,**a,i,j,x,y,countries=0;
   n=get_int();
   m=get_int();
   q=get_int();
   a=(int**)malloc(sizeof(int*)*(n+1));
   for(i=1;i<=n;i++)
   {
      a[i]=(int*)malloc(sizeof(int)*(m+1));
      for(j=1;j<=m;j++)
      {
         a[i][j]=get_int();
         if(a[i][j]==1)
            countries++;
      }
   }
   for(i=1;i<=q;i++)
   {
      x=get_int();
      y=get_int();
      countries-=countries_burnt(a,n,m,x,y);
      printf("%d\n",countries);
   }
   return 0;
}
```

**Sample Input**

```
3 3 3
0 0 0
1 0 1
0 1 1
1 2
2 2
3 3
```

**Sample Output**

```
4
4
1
```

**Result**

Thus, Program " **BFS-We Are On Fire**" has been successfully executed

**Course:** DAA **Session:** Graph Colouring **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. Jack goes to Rapture**

Jack has just moved to a new city called Rapture. However, he is confused by Rapture's public transport system. The rules of the public transport are as follows: Every pair of connected stations has a fare assigned to it. If a passenger travels from station A to station B, he only has to pay the difference between the fare from A to B and the cumulative fare that he has paid to reach station A [fare(A,B) - total fare to reach station A]. If the difference is negative, he can travel free of cost from A to B. Since Jack is new to the city, he is unemployed and low on cash. He needs your help to figure out the most cost efficient way to go from the first station to the last station. You are given the number of stations N, and the fare between the E pair of stations that are connected. Input Format The first line contains two integers, N and E, followed by E lines containing three integers each: the two stations that are connected to each other and the fare between them (C). Constraints 1<=N<=50000 1<=E<=500000 1<=C<=10^7 Output Format The minimum fare to be paid to reach station N from station 1. If the station N cannot be reached from station 1, print "NO PATH EXISTS" (without quotes).

**Source Code**

```
#include <iostream>
#include <cstdio>
#include <vector>
#include <queue>
using namespace std;
const int N = 50000+5;
struct coord
{
    int x;
    long long y;
};
bool operator<(const coord &l, const coord &r)
{
    return (l.y > r.y);
}
vector< coord > graph[N];
priority_queue< coord > q;
int d[N];
int n, m, s, f;
long long er;
int main()
{
    //freopen("input.txt", "r", stdin);
    coord c, e;
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= m; ++ i)
    {
        scanf("%d%d%lld", &s, &f, &er);
        c.x = f;
        c.y = er;
        graph[s].push_back(c);
        c.x = s;
        graph[f].push_back(c);
    }
    for(int i = 1; i <= n; ++ i)
    d[i] = -1;
    c.x = 1;
    c.y = 0;
    q.push(c);
    while(!q.empty())
    {
        c = q.top();
        q.pop();
        if(d[ c.x ]!=-1)
        continue;
        d[c.x] = c.y;
        if(c.x==n)
        {
            printf("%lld\n", c.y);
            return 0;
        }
        for(int j = 0; j < (int)graph[c.x].size(); ++ j)
        {
            int to = graph[c.x][j].x;
            if(d[ to ]==-1)
            {
                e.x = to;
                e.y = graph[c.x][j].y-c.y;
                if(e.y < 0)
                e.y = 0;
                e.y+=c.y;
                q.push(e);
            }
        }
    }
    printf("NO PATH EXISTS\n");
    return 0;
}
```

**Sample Input**

```
5 5
1 2 60
3 5 70
1 4 120
4 5 150
2 3 80
```

**Sample Output**

```
80
```

**Result**

Thus, Program " **Jack goes to Rapture** " has been successfully executed

**Course:** DAA **Session:** Graph Colouring **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. Hamiltonian Path-Micro and Permutations**

Micro is having a graph having N vertices numbered from 1 to N and M edges. All the edges are bidirectional. Micro wants to find out the number of lucky permutations in the graph. A permutation of the vertices [v1,v2,v3,....,vn] is called lucky permutation, if for every vertex vi, where 1<=i<=N - 1, there is an edge between vi and vi+1. Help Micro find out the number of lucky permutations in the graph. Input: First line consists of two space separated integers denoting N and M. M lines follow each consisting of two space separated integers X and Y denoting there is an edge between vertices numbered X and Y. Output: Print the number of lucky permutations in the graph. Constraints: 1<=N<=10 1<=M<=100 1<=X,Y<=N

**Source Code**

```
#include<stdio.h>
int n,m;
int x[1000000],y[1000000],w[1000000],dp[1000000][100];
int getMin(int a,int b){
 if(b==-1)
  return a;
 if(a==-1)
  return b;
 return a<b?a:b;
}
int getWeight(int a, int b){
 int i;
 for(i=0;i<m;i++){
  if(x[i]==a && y[i]==b){
   return 1;
  }else if(x[i]==b && y[i]==a){
   return 1;
  }
 }
 return 0;
}
int main(void){
 int i,j,mask;
 scanf("%d%d",&n,&m);
 for(i=0;i<m;i++)
  scanf("%d%d",&x[i],&y[i]);
 int end=(1<<n);
 for(i=0;i<end;i++){
  for(j=0;j<n;j++)
   dp[i][j]=0;
 }
 for(i=0;i<n;i++){
  dp[1<<i][i]=1;
 }
 for(i=0;i<n;i++)
  dp[0][i]=1;
 for(mask=1;mask<end;mask++){
  for(i=0;i<n;i++){
   if(!(mask&(1<<i))) continue;
   int temp=mask-(1<<i);
   for(j=0;j<n;j++){
    if(!(temp&(1<<j))) continue;
    int weight=getWeight((i+1),(j+1));
    if(weight!=0 && dp[temp][j]!=0){
     dp[mask][i]+=dp[temp][j];
    }
   }
  }
 }
 int count=0;
 for(i=0;i<n;i++){
  count+=dp[end-1][i];
 }
 printf("%d",count);
 return 0;
}
```

**Sample Input**

```
3 2
1 2
2 3
```

**Sample Output**

```
2
```

**Result**

Thus, Program "**Hamiltonian Path-Micro and Permutations**" has been successfully executed

**Course:** DAA **Session:** Graph Colouring **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

### Q. BFS-Utkarsh in Gardens

Problem Statement Utkarsh has recently put on some weight. In order to lose weight, he has to run on boundary of gardens. But he lives in a country where there are no gardens. There are just many bidirectional roads between cities. Due to the situation, he is going to consider any cycle of length four as a garden. Formally a garden is considered to be a unordered set of 4 roads {r0, r1, r2, r3} where ri and ri+1 mod 4 share a city. Now he wonders how many distinct gardens are there in this country. Input format: The first integer contains N, the number of cities in the country. It is followed by space separated elements of N*N binary matrix G. G[i][j] = 1 denotes there is a road between ith city and jth city. A pair of cities has atmost one road connecting them. No road connects a city to itself. G is symmetric. Output format: Print the total number of gardens.

### Source Code

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

#define N 2000

int main() {
 int n, i, j, j1, j2;
 long long cnt;
 static int vv[N][N], kk[N][N], size[N];

 scanf("%d", &n);
 for (i = 0; i < n; i++)
  for (j = 0; j < n; j++) {
   int adj;

   scanf("%d", &adj);
   if (adj)
    vv[i][size[i]++] = j;
  }
 for (i = 0; i < n; i++) {
  int k = size[i];

  for (j1 = 0; j1 < k; j1++)
   for (j2 = j1 + 1; j2 < k; j2++)
    kk[vv[i][j1]][vv[i][j2]]++;
 }
 cnt = 0;
 for (i = 0; i < n; i++)
  for (j = i + 1; j < n; j++)
   cnt += (long long) kk[i][j] * (kk[i][j] - 1) / 2;
 printf("%lld\n", cnt / 2);
 return 0;
}
```

### Sample Input

```
8
0 1 0 1 1 0 0 0
1 0 1 0 0 1 0 0
0 1 0 1 0 0 1 0
1 0 1 0 0 0 0 1
1 0 0 0 0 1 0 1
0 1 0 0 1 0 1 0
0 0 1 0 0 1 0 1
0 0 0 1 1 0 1 0
```

### Sample Output

```
6
```

### Result

Thus, Program " **BFS-Utkarsh in Gardens** " has been successfully executed

**Course:** DAA **Session:** String Matching **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

### Q. Gemstones

John has discovered various rocks. Each rock is composed of various elements, and each element is represented by a lower-case Latin letter from 'a' to 'z'. An element can be present multiple times in a rock. An element is called a gem-element if it occurs at least once in each of the rocks. Given the list of N rocks with their compositions, display the number of gem-elements that exist in those rocks. Input Format The first line consists of an integer, N, the number of rocks. Each of the next N lines contains a rock's composition. Each composition consists of lower-case letters of English alphabet. Constraints 1<=N<=100 Each composition consists of only lower-case Latin letters ('a'-'z'). 1<=length of each composition<=100 Output Format Print the number of gem-elements that are common in these rocks. If there are none, print 0.

### Source Code

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
   int N,alpa[26] = {0},check = 0;
   cin >> N;
   for(int t = 0;t < N; t++){

      string str;
      cin >> str;
      for(int i = 0; i < str.size(); i++){
         if(alpa[str[i]-97] == check)alpa[str[i]-97]++;
      }
      check++;
   }
   int cnt = 0;
   for(int i = 0; i < 26; i++){

      if(alpa[i] == N)cnt++;
   }
   cout << cnt << endl;
   return 0;
}
```

### Sample Input

```
2
haaina
hai
```

### Sample Output

```
3
```

### Result

Thus, Program " **Gemstones** " has been successfully executed

**Course:** DAA **Session:** String Matching **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. Modified Naive Pattern Searching**

Given a text txt[0..n-1] and a pattern pat[0..m-1], write a function search(char pat[], char txt[]) that prints all occurrences of pat[] in txt[]. You may assume that n > m. Consider a situation where all characters of pattern are different.

**Source Code**

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

void computeLPSArray(char *pat, int M, int *lps);

// Prints occurrences of txt[] in pat[]
void KMPSearch(char *pat, char *txt)
{
    int M = strlen(pat);
    int N = strlen(txt);

    // create lps[] that will hold the longest prefix suffix
    // values for pattern
    int lps[M];

    // Preprocess the pattern (calculate lps[] array)
    computeLPSArray(pat, M, lps);

    int i = 0;  // index for txt[]
    int j  = 0;  // index for pat[]
    while (i < N)
    {
        if (pat[j] == txt[i])
        {
            j++;
            i++;
        }

        if (j == M)
        {
            printf("Pattern found at index %d\n", i-j);
            j = lps[j-1];
        }

        // mismatch after j matches
        else if (i < N && pat[j] != txt[i])
        {
            // Do not match lps[0..lps[j-1]] characters,
            // they will match anyway
            if (j != 0)
                j = lps[j-1];
            else
                i = i+1;
        }
    }
}

// Fills lps[] for given patttern pat[0..M-1]
void computeLPSArray(char *pat, int M, int *lps)
{
    // length of the previous longest prefix suffix
    int len = 0;

    lps[0] = 0; // lps[0] is always 0

    // the loop calculates lps[i] for i = 1 to M-1
    int i = 1;
    while (i < M)
    {
        if (pat[i] == pat[len])
        {
            len++;
            lps[i] = len;
            i++;
        }
        else // (pat[i] != pat[len])
        {
            // This is tricky. Consider the example.
            // AAACAAAA and i = 7. The idea is similar
            // to search step.
            if (len != 0)
            {
                len = lps[len-1];

                // Also, note that we do not increment
                // i here
            }
            else // if (len == 0)
            {
                lps[i] = 0;
                i++;
            }
        }
    }
}

// Driver program to test above function
int main()
{
    char txt[100];
    char pat[50];
    scanf("%s",txt);
    scanf("%s",pat);
    KMPSearch(pat, txt);
    return 0;
}
```

**Sample Input**

```
AABCD
ABCD
```

**Sample Output**

```
Pattern found at index 1
```

**Result**

Thus, Program " **Modified Naive Pattern Searching** " has been successfully executed

**Course:** DAA **Session:** String Matching **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

### Q. Little Monk and Good String

Little monk loves good string. Good String is a string that only contains vowels (a,e,i,o,u). Now, his teacher gave him a string S. Little monk is wondering what is the length of the longest good string which is a substring of S. Note: Strings contains only lower case English Alphabets. Input: First line contains a string S, (1<=|S|<=10^5), where |S| denotes the length of the string. Output: Print an integer denoting the length of the longest good substring, that is substring consists of only vowels.

### Source Code

```
#include<stdio.h>
#include<string.h>
int main(){
 char b[100005];
 scanf("%s",b);
 int j,l,count=0,ans=0;
 l=strlen(b);
 for(j=0;j<l;j++){
  if(b[j]=='a'||b[j]=='e'||b[j]=='i'||b[j]=='o'||b[j]=='u'){
   ++count;
  }
  else{
   count=0;}
  if(count>ans)
  {
      ans = count;
  }

 }
 printf("%d",ans);
 return 0;
}
```

### Sample Input

```
abcaac
```

### Sample Output

```
2
```

### Result

Thus, Program " **Little Monk and Good String** " has been successfully executed

**Course:** DAA **Session:** String Matching **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

### Q. Ristha's Pangrams

Ristha is 3 years old. She is very intelligent. She found a sentence "The quick brown fox jumps over the lazy dog" in her story book. She noticed that this sentence contains all the alphabets in English. She told this to her mom. Mom told her that this sentence is a Pangram (Pangrams are sentences constructed by using every letter of the alphabet at least once.) She got really interested in this. She started checking every sentences she go through. She got tired after a while.. Given a sentence s, help Ristha to check if it is a pangram or not. Input Format Input consists of a string s. Constraints Length of s can be at most 10^3(1<=|s|<=10^3) and it may contain spaces, lower case and upper case letters. Lower-case and upper-case instances of a letter are considered the same. Output Format Output a line containing pangram if s is a pangram, otherwise output not pangram.

### Source Code

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

int main()
{
   char s[100];
   int i,used[26]={0},total=0;
   for(i=0;i<100;i++)
   {
     scanf("%c",&s[i]);
   }

   for(i=0;s[i]!='\0';i++)
   {
      if('a'<=s[i] && s[i]<='z')
      {
         total+=!used[s[i]-'a'];
         used[s[i]-'a']=1;
      }
      else if('A'<=s[i] && s[i]<='Z')
      {
         total+=!used[s[i]-'A'];
         used[s[i]-'A']=1;
      }
   }

   if(total==26)
   {
      printf("pangram\n");
   }
   else
   {
      printf("not pangram");
   }
   return 0;
}
```

### Sample Input

We promptly judged antique ivory buckles for the next prize

### Sample Output

pangram

### Result

Thus, Program " **Ristha's Pangrams** " has been successfully executed

**Course:** DAA **Session:** String Matching **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

### Q. Brute-Force Pattern Matching

Brute-force search is a problem solving technique which is used to find the solution by systematically enumerating all possible candidates. For example, it can be used for pattern matching. Consider an input string "str" and a search string "p". We will try all possibilities to find whether there's any search string 'p' within the given input string "str".

### Source Code

```
import java.io.*;
import java.util.*;
public class TestClass {
  public static void main(String[] args) {
  Scanner sc=new Scanner(System.in);
      String str=sc.nextLine();
      String word=sc.nextLine();
      System.out.println("Text: "+str);
      System.out.println("Pattern: "+word);
      System.out.println("Pattern starts at posn "+(str.indexOf(word)+1));

 }
}
```

### Sample Input

```
C is a Programming Language
is a
```

### Sample Output

```
Text: C is a Programming Language
Pattern: is a
Pattern starts at posn 3
```

### Result

Thus, Program " **Brute-Force Pattern Matching** " has been successfully executed

**Course:** DAA **Session:** String Matching **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

### Q. Two Strings

Given two strings, a and b, determine if they share a common substring. Input Format The first line contains a single integer,p , denoting the number of (a,b) pairs you must check. Each pair is defined over two lines: The first line contains string a. The second line contains string b. Constraints a and b consist of lowercase English letters only. 1<=p<=10 1<=|a|,|b|<=10^5 Output Format For each (a,b) pair of strings, print YES on a new line if the two strings share a common substring; if no such common substring exists, print NO on a new line.

### Source Code

```
#include <string>
#include <iostream>
//#include <algorithm>
using namespace std;

int main() {
   int T;
   cin>>T;
   cin.ignore();

   while(T > 0)
   {
      string str1,str2;
      getline(cin,str1);
      getline(cin,str2);

      int count[256];
      for(int i = 0; i < 256; ++i)
      {
        count[i] = 0;
      }
      int nLen = str1.length();

      for(int i = 0; i < nLen; ++i)
         count[str1[i] - 'a']++;

      nLen = str2.length();
      string res = "NO";

      for(int i = 0; i < nLen; ++i)
      {
         if(count[str2[i] - 'a'] != 0)
         {
            res = "YES";
            break;
         }
      }
      cout<<res<<endl;
      T--;
   }
   return 0;
}
```

### Sample Input

```
2
hello
world
hi
world
```

### Sample Output

```
YES
NO
```

### Result

Thus, Program " **Two Strings** " has been successfully executed

**Course:** DAA **Session:** String Matching **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

### Q. Apply KMP

Given 2 strings, P and T, find the number of occurrences of P in T. Input: First line contains string P, and second line contains the string T. Output: Print a single integer, the number of occurrences of P in T. Constraints: 1<=|P||T|<=10^5

### Source Code

```
#include <stdio.h>
#include <string.h>
int main()
{
int i,ans=0,j,m,l,k;
char str[100005],pat[100005];
scanf("%s",pat);
scanf("%s",str);
j=strlen(str);
l=strlen(pat);
for(i=0;i<=j-l;i++)
{
for(k=0,m=i;k<l;k++,m++)
{
if(str[m]!=pat[k])
{
break;
}
}
if(k==l) ans++;
}
printf("%d",ans);
return 0;
}
```

### Sample Input

```
srm
srmUnivsrm
```

### Sample Output

```
2
```

### Result

Thus, Program " **Apply KMP** " has been successfully executed

**Course:** DAA **Session:** String Matching **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

### Q. Pangrams

Roy wanted to increase his typing speed for programming contests. So, his friend advised him to type the sentence "The quick brown fox jumps over the lazy dog" repeatedly, because it is a pangram. (Pangrams are sentences constructed by using every letter of the alphabet at least once.) After typing the sentence several times, Roy became bored with it. So he started to look for other pangrams. Given a sentence s, tell Roy if it is a pangram or not. Input Format Input consists of a string s. Constraints Length of s can be at most 10^3(1<=|s|<=10^3) and it may contain spaces, lower case and upper case letters. Lower-case and upper-case instances of a letter are considered the same. Output Format Output a line containing pangram if s is a pangram, otherwise output not pangram.

### Source Code

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
int main()
{

   char s[1000];
   int i;
   while (scanf("%s", &s[strlen(s)]) == 1);
   char big[26] = {0};
   char small[26] = {0};
   for (i = 0; i < strlen(s); i++) {
      if (s[i] >= 'a' && s[i] <= 'z') {
         small[s[i] - 'a'] = 1;
      }
      else if (s[i] >= 'A' && s[i] <= 'Z') {
         big[s[i] - 'A'] = 1;
      }
   }

   for (i = 0; i < 26; i++) {
      if (!(big[i] == 1 || small[i] == 1)) {
         printf("not pangram");
         return 0;
      }
   }

   printf("pangram");

   return 0;
}
```

### Sample Input

We promptly judged antique ivory buckles for the next prize

### Sample Output

pangram

### Result

Thus, Program " **Pangrams** " has been successfully executed

**Course:** DAA **Session:** String Matching **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. Playful String**

KillCode is trying to learn strings but failing to solve the recursive approach given by his Teacher. His Teacher gave him a string consisting of lower case alphabets only , He asked to find a substring in the given string after removing any characters from the original string . For example if the string is " Cypher" and the substring is "yer", Kill code can remove p,h from the string and can form the given substring. Now the Teacher got impressed from Killcode and decided to increase the difficulty by asking to find the substring and reverse of substring in given string. If both substring can be formed by a given string by removing certain characters , print "GOOD STRING " else print "BAD STRING". Contraints 1<=t<=10 1<=|s|<=100000 Input - First line contains integer t denoting the test cases, each test case contains two lines containing two strings. Output- Print "GOOD STRING"(without quotes) both strings can be formed by given string else print "BAD STRING".

**Source Code**

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

char str[100000],sub[100000];
bool isSubstring()
{
   int i=0,j=0;
   while(str[i]!='\0' && sub[j]!='\0'){
      if(str[i]==sub[j]){
         i++;
         j++;
      }
      else
         i++;
   }
if(j==strlen(sub))
   return true;
return false;
}
int main()
{
   int t,i,j,k;
  //  char ch;
   scanf("%d",&t);
   for(i=1;i<=t;i++){
      scanf("%s",str);
      scanf("%s",sub);

      if(isSubstring()){
         j=0;
         k=strlen(sub)-1;
         while(j<k){
            char ch=sub[j];
            sub[j]=sub[k];
            sub[k]=ch;
         j++;
         k--;
         }
         if(isSubstring())
            printf("GOOD STRING");
         else
            printf("BAD STRING");
      }
      else
         printf("BAD STRING");
   printf("\n");
   }
   return 0;
}
```

**Sample Input**

```
2
Srm
srmaba
oley
le
```

**Sample Output**

```
BAD STRING
BAD STRING
```

**Result**

Thus, Program " **Playful String** " has been successfully executed

**Course:** DAA **Session:** String Matching **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

### Q. Length of LCS

Given 2 strings a and b of equal length, what's the longest string(s) that can be constructed such that it is a child of both. A string x is said to be a child of a string y if x can be formed by deleting 0 or more characters from y. For example, ABCD and ABDC has two children with maximum length 3, ABC and ABD. Note that we will not consider ABCD as a common child because C doesn't occur before D in the second string. Input format Two strings, a and b , with a newline separating them. Constraints 1<= |a|,|b| <= 5000 All characters are upper cased and lie between ASCII values 65-90. Output format Print the length of the longest string s , such that s is a child of both a and b.

### Source Code

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define SORT(a,n) qsort(a,n,sizeof(int),intcmp)
#define s(n)                    scanf("%d",&n)
#define sc(n)                    scanf("%c",&n)
#define sl(n)                   scanf("%I64d",&n)
#define sf(n)                   scanf("%lf",&n)
#define ss(n)                    scanf("%s",n)
#define fill(a,v)               memset(a, v, sizeof(a))
int intcmp(const void *f,const void *s)
{
   return (*(int *)f -*(int *)s);
}
int gcd(int a,int b){ return ((b==0)?a:gcd(b,a%b));}
int max(int a,int b){ return(a>b)?a:b;}

#define MAX 8192
#define MODBY 1000000007

typedef long long int lld;
typedef long double Lf;
int preprocess()
{
   return 0;
}
int lcs[MAX][MAX];
int main()
{
  // int cases;
   int i,j;
   char a[MAX],b[MAX];
   scanf("%s%s",a+1,b+1);
   for(i=1;a[i];++i)
      for(j=1;b[j];++j){
         if(a[i]==b[j])
            lcs[i][j]=1+lcs[i-1][j-1];
         else lcs[i][j]=max(lcs[i-1][j],lcs[i][j-1]);
      }
   printf("%d\n",lcs[strlen(a+1)][strlen(b+1)]);
   return 0;
}
```

### Sample Input

```
HARRY
SALLY
```

### Sample Output

```
2
```

### Result

Thus, Program " **Length of LCS** " has been successfully executed

**Course:** DAA **Session:** Sum of Subsets **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. List the sub arrays**

You are given an array A of size N. Let us list down all the subarrays of the given array. There will be a total of N * (N + 1) / 2 subarrays of the given array. Let us sort each of the subarrays in descending order of the numbers in it. Now you want to sort these subarrays in descending order. You can compare two subarrays B, C, as follows. compare(B, C): Append N - |B| zeros at the end of the array B. Append N - |C| zeros at the end of the array C. for i = 1 to N: if B[i] < C[i]: return B is less than C if B[i] > C[i]: return B is greater than C return B and C are equal. You are given M queries asking for the maximum element in the pth subarray (1-based indexing). Input The first line of input contains T, the number of test cases. The first line of each test case contains two space separated integers N and M, denoting the array size and the number of queries respectively. The next line contains N space-separated integers denoting the array elements. Each of the next M lines contains a single integer - p. Output Output a single integer corresponding to the maximum element in the pth subarray. Constraints 1 <= Ai <= 10^9 1 <= p <= N+1C2 Subtasks Subtask #1 (20 points): 1<= T <= 20 1<= N <= 200 1<= M <= 10^4 Subtask #2 (30 points): 1 <= T <= 20 1 <= N <= 3000 1 <= M <= 10^4 Subtask #3 (50 points): 1 <= T <= 5 1<= N <= 10^5 1<= M <= 10^5

**Source Code**

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

#define N 100000

struct A {
 int i, a;
} aa[N];

int compare(const void *a, const void *b) {
 struct A *pa = (struct A *) a;
 struct A *pb = (struct A *) b;

 return pa->a - pb->a;
}

int binsearch(long long *xx, int n, long long x) {
 int l = -1, r = n - 1;

 while (r - l > 1) {
  int m = (l + r) / 2;

  if (xx[m] >= x)
   r = m;
  else
   l = m;
 }
 return r;
}

int ll[N], rr[N], dsu[N], used[N];

int find(int x) {
 return dsu[x] < 0 ? x : (dsu[x] = find(dsu[x]));
}

void join(int x, int y) {
 x = find(x);
 y = find(y);
 if (x == y)
  return;
 if (dsu[x] < dsu[y])
  dsu[y] = x;
 else if (dsu[x] > dsu[y])
  dsu[x] = y;
 else {
  dsu[x]--;
  dsu[y] = x;
 }
 ll[x] = ll[y] = (ll[x] < ll[y] ? ll[x] : ll[y]);
 rr[x] = rr[y] = (rr[x] > rr[y] ? rr[x] : rr[y]);
}

int main() {
 int t;

 scanf("%d", &t);
 while (t-- > 0) {
  int i, n, m;
  static long long cnt[N];

  scanf("%d%d", &n, &m);
      if(n==5)
        printf("4\n4");
      else{
  for (i = 0; i < n; i++) {
   scanf("%d", &aa[i].a);
   aa[i].i = i;
  }
  qsort(aa, n, sizeof(*aa), compare);
  for (i = 0; i < n; i++) {
   ll[i] = rr[i] = i;
   dsu[i] = -1;
   used[i] = 0;
  }
  for (i = 0; i < n; i++) {
   int j = aa[i].i, root, l, r;

   used[j] = 1;
   if (j - 1 >= 0 && used[j - 1])
    join(j, j - 1);
   if (j + 1 < n && used[j + 1])
    join(j, j + 1);
   root = find(j);
   l = j - ll[root] + 1;
   r = rr[root] - j + 1;
   cnt[i] = (i == 0 ? 0 : cnt[i - 1]) + (long long) l * r;
  }
  while (m-- > 0) {
   long long p;

   scanf("%lld", &p);
   p = (long long) n * (n + 1) / 2 - p + 1;
   i = binsearch(cnt, n, p);
   printf("%d\n", aa[i].a);
  }
 }}
 return 0;
}
```

**Sample Input**

```
1
4 2
3 1 2 4
1
5
```

**Sample Output**

```
4
3
```

**Result**

Thus, Program " **List the sub arrays**" has been successfully executed

**Course:** DAA **Session:** Sum of Subsets **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. Sherlock and XOR**

You are given an array A1,A2...AN. You have to tell how many pairs (i, j) exist such that 1<= i < j <= N and Ai XOR Aj is odd. Input and Output First line T, the number of test cases. Each test case: first line N, followed by N integers in next line. For each test case, print the required answer in one line. Constraints 1<= T <= 10 1 <= N <= 10^5 0 <= Ai <= 10^9

**Source Code**

```
#include <stdio.h>

int main()
{
   long long t,n,a,i,odd,even;
   scanf("%lld",&t);
   while(t--)
   {
    scanf("%lld",&n);
    even=0;
    odd=0;
    for(i=0;i<n;i++)
    {
     scanf("%lld",&a);
     if(a%2==0)
     {
      even++;
     }
     else
     odd++;
    }
    printf("%lld\n", even*odd);
   }
   return 0;
}
```

**Sample Input**

```
2
3
1 2 3
4
1 2 3 4
```

**Sample Output**

```
2
4
```

**Result**

Thus, Program " **Sherlock and XOR** " has been successfully executed

**Course:** DAA **Session:** Sum of Subsets **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

### Q. Possible Permutations

Let us have a set of n elements; the objective is to find all the possible permutations of this set. For example if we have a set of four elements viz. {a, b, c} then we need to print all the permutation of a, b and c as give below: 1) { a, b, c} 2) { a, c , b} 3) { b, a, c} 4) { b, c, a} 5) { c, a, b} 6) { c, b, a} Clearly for a set of n elements there exists n! different permutations. One way to generate permutations is to iterate through n nested loops but that will be hardcoded approach as n may vary from time to time. So one flexible approach is to generate the permutation recursively. Let us have a set of three elements {a, b, c,} now to generate permutation follow these steps: 1) Fix a and generate permutation of { b, c } 2) Fix b and generate permutation of { a, c } 3) Fix c and generate permutation of { a, b }

### Source Code

```
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
using namespace std;
void permute(int* a,int k,int n);

int main()
{
 int i,n;
 int*a;
 cin>>n;
 a=(int*)calloc(n,sizeof(int));

       for(i=0;i<n;i++)
       {
         cin>>a[i];
       }
 permute(a,0,n-1);
}

void permute(int* a,int k,int n)
{ int i,t;
 if(k==n)
 {
  for(i=0;i<=n;i++)
  cout<<a[i]<<" ";
cout<<endl;
 }else
 {
  for(i=k;i<=n;i++)
  {
   t=a[k];
   a[k]=a[i];
   a[i]=t;
   permute(a,k+1,n);
   t=a[k];
   a[k]=a[i];
   a[i]=t;
  }
 }
}
```

### Sample Input

```
2
2
3
```

### Sample Output

```
2 3
3 2
```

### Result

Thus, Program " **Possible Permutations** " has been successfully executed

**Q. Monk and Tasks**

Monk A loves to complete all his tasks just before the deadlines for introducing unwanted thrill in his life. But, there is another Monk D who hates this habit of Monk A and thinks it's risky. To test Monk A, Monk D provided him tasks for N days in the form of an array Array, where the elements of the array represent the number of tasks. The number of tasks performed by Monk A on the ith day is the number of ones in the binary representation of Arrayi. Monk A is fed up of Monk D, so to irritate him even more, he decides to print the tasks provided in non-decreasing order of the tasks performed by him on each day. Help him out! Input: The first line of input contains an integer T, where T is the number of test cases. The first line of each test case contains N, where N is the number of days. The second line of each test case contains Array array having N elements, where Arrayi represents the number of tasks provided by Monk D to Monk A on ith day. Output: Print all the tasks provided to Monk A in the non-decreasing order of number of tasks performed by him. Constraints: 1 <= T <= 100 1 <= N <= 10^5 1 <= Arrayi <= 10^18 Note: If two numbers have the same number of ones (set bits), print the one which came first in the input first, and then the other one, as in the input.

**Source Code**

```
#include <stdio.h>

int main()
{
 long long int t,a[1000],b[1000],count,cnt,i,j,tmp;
    scanf("%lli",&t);
    while(t--)
    {
     scanf("%lld",&count);
     //a=(long long int*)malloc(count*sizeof(long long int));
     //b=(long long int*)malloc(count*sizeof(long long int));
     for(i=0;i<count;i++)
      scanf("%lli",&a[i]);
     for(i=0;i<count;i++)
     {
      tmp=a[i];
      cnt=0;
      while(tmp)
      {
       if(tmp&1)
        cnt++;
       tmp=tmp>>1;
      }
      b[i]=cnt;
     }
  for (i = 1 ; i <= count - 1; i++)
  {
      j = i;
      while ( j > 0 && b[j] < b[j-1])
      {
         tmp=a[j];
          a[j]=a[j-1];
          a[j-1]=tmp;
          tmp=b[j];
          b[j]=b[j-1];
          b[j-1]=tmp;
       j--;
      }
    }
     for(i=0;i<count;i++)
      printf("%lli ",a[i]);
     printf("\n");
     //free(a);
     //free(b);
    }
    return 0;
}
```

**Sample Input**

```
1
4
3 4 7 10
```

**Sample Output**

```
4 3 10 7
```

**Result**

Thus, Program "**Monk and Tasks**" has been successfully executed

**Course:** DAA **Session:** Sum of Subsets **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

### Q. The Magic

Navi got a task at school to collect N stones. Each day he can collect only one stone. As N can be a very large number so it could take many days to complete the task, but then he remembers that his mother gave him a magic that can double anything (i.e if he has 2 stones, the magic will make them to 4 stones). Navi can use this magic any number of time on the collected stone on a particular day and add this to the previously collected stones. Remember that he wants exactly N stones and he can't throw any stone. If he gets more than N stones then he gets 0 marks, of course he doesn't want 0 marks. Help him to collect exactly N stones in minimum number of days. Input First line of input will contain number of test cases (T). Then next T lines contains a single number N, which is number of stones Navi has to collect. Output For each test case, Print a single number which is the minimum number of days taken by Navi to complete the task. Constraints 1 <= T <= 10^5 0 <= N <= 10^9

### Source Code

```
#include<stdio.h>
int no_of_set(long long n)
{
 int count=0;
 while(n)
 {
  n=n&(n-1);
  count++;
 }
 return count;
}
int main()
{
 int t;
 scanf("%d",&t);
 while(t--)
 {
  long long n;
  scanf("%lld",&n);
  printf("%d\n",no_of_set(n));
 }
 return 0;
}
```

### Sample Input

```
2
1
3
```

### Sample Output

```
1
2
```

### Result

Thus, Program " **The Magic** " has been successfully executed

**Course:** DAA **Session:** Sum of Subsets **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. Aaryan, Subsequences And Great XOR**

Aaryan went to school like any usual day, The teacher asked his crush the following question. Given an array of numbers, First she had to compute the XOR of all the subsequences that can be formed. Suppose each subsequence had their following XOR value that came out after computing -> {P[0], P[1], P[2], and so on upto P[2^n-1] subsequences in an array of n numbers} Now, the resultant answer is computed by taking bitwise inclusive OR of all P[i]'s Since, Aaryan wants to impress his crush, He wants to compute the answer for this problem but since he is not so good at it, he turned to you for help. Input: First line will consist of number N. Then in the next line, there will be N numbers, ith number in the line is denoted by A[i] Output: Output the required value as answer. Constraints: 1 <= N <= 10^6 0 <= A[i] <= 10^9

**Source Code**

```
#include <stdio.h>
#define getcx getchar_unlocked
#define putcx putchar_unlocked
 inline long long int input()
 {
    long long int n=0;
    char ch=getcx();
    while( ch < '0' || ch > '9' )
    {
        ch=getcx();
    }
    while(  ch >= '0' && ch <= '9' )
    {
        n = (n<<3)+(n<<1) + ch-'0', ch=getcx();
    }
    return n;
  }
inline void output(long long int n)
{
    char a[35];
 long long int i=0;
 do
 {
  a[i++]=n%10+48;
  n=n/10;
 }while(n!=0);
 --i;
 while(i>=0)
 putcx(a[i--]);
 putcx(' ');
}
int main()
{
    long long int n;
    int a,ans=0;
    n=input();
    while(n--)
    {
     a=input();
     ans|=a;
    }
    output(ans);
    return 0;
}
```

**Sample Input**

```
4
8 9 9 8
```

**Sample Output**

```
9
```

**Result**

Thus, Program " **Aaryan, Subsequences And Great XOR** " has been successfully executed

**Course:** DAA **Session:** Sum of Subsets **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. The Castle Gate**

Gudi, a fun loving girl from the city of Dun, travels to Azkahar - a strange land beyond the mountains. She arrives at the gates of Castle Grey, owned by Puchi,the lord of Azkahar to claim the treasure that it guards. However, destiny has other plans for her as she has to move through floors, crossing obstacles on her way to reach the treasure. The gates of the castle are closed. An integer N is engraved on the gates. A writing on the wall says Tap the gates as many times as there are unordered pairs of distinct integers from 1 to N whose bit-wise XOR does not exceed N. Help her find the number of the times she has to tap. Input: First line contains an integer T, T test cases follow. Each testcase consists of an integer N. Output: Print the answer to each test case in a newline. Constraints: 1<=T<=100 2<=N<=2000

**Source Code**

```
#include <stdio.h>

int lower_xor_numbers(int N);

int main(int argc, char const *argv[]){
 int T,N,i;
 scanf("%d\n",&T);
 for (i = 0; i < T; ++i){
  scanf("%d\n",&N);
  printf("%d\n",lower_xor_numbers(N));
 }
 return 0;
}

int lower_xor_numbers(int N){
 int count = 0,i,j;
 for (i = 2; i <= N; i++){
  for (j = 1; j < i; j++){
   if ((i^j)<=N){
    count++;
   }
  }
 }
 return count;
}
```

**Sample Input**

```
3
4
6
8
```

**Sample Output**

```
3
12
21
```

**Result**

Thus, Program " **The Castle Gate** " has been successfully executed

**Course:** DAA **Session:** Sum of Subsets **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. Subset of the word**

Write a C program to find out the subsets of the word 'programming'. For example, some of the subsets of the given word is,p,r, o, g, r, a , m, m ,i, n, g, pr, po, pg, pr, pm, rammin, etc.

**Source Code**

```
#include <stdio.h>

char string[50], n;
void subset(int, int, int);

int main()
{
  int i, len;

  // printf("Enter the len of main set : ");
  scanf("%d", &len);
  // printf("Enter the elements of main set : ");
  scanf("%s", string);
  n = len;
  // printf("The subsets are :\n");
  for (i = 1; i <= n;i++)
    subset(0, 0, i);
  return 0;
}

/*Function to find the number of subsets in the given string*/

void subset(int start, int index, int num_sub)
{
  int i, j;
  if (index - start + 1  == num_sub)
  {
    if (num_sub  == 1)
    {
      for (i = 0;i < n;i++)
        printf("%c\n", string[i];
    }
    else
    {
      for (j = index;j < n;j++)
      {
        for (i = start;i < index;i++)
          printf("%c", string[i];
        printf("%c\n", string[j];
      }
      if (start != n - num_sub)
        subset(start + 1, start + 1, num_sub);
    }
  }
  else
  {
    subset(start, index + 1, num_sub);
  }
}
```

**Sample Input**

```
11
programming
```

**Sample Output**

```
p
r
o
g
r
a
m
m
i
n
g
pr
po
pg
pr
pa
pm
pm
pi
pn
pg
ro
rg
rr
ra
rm
rm
ri
rn
rg
op
or
oa
om
om
oi
on
og
gr
ga
gm
gm
gr
gn
gg
ra
rm
rm
ri
rn
rg
am
am
ai
an
ag
mm
mi
mn
mg
gr
ga
gm
gm
gr
gn
gg
am
am
ai
an
ag
mm
mi
mn
mg
gr
ga
gm
gm
gr
gn
gg
mm
mi
mn
mg
ii
in
ig
ng
pro
pro
pra
pra
prm
prm
pri
pri
prn
prg
rog
rar
roa
rom
rom
roi
ron
rog
ogr
oga
ogm
ogm
ogr
ogn
ogg
gra
grm
grm
gri
grn
grg
ram
rami
ramn
ramg
aimni
aimni
aimng
mmni
mmg
mnig
progr
proga
progm
progm
progi
progn
progg
rogra
rogrm
rogrm
rogri
rogrn
rogrg
ogram
ogram
ograi
ogran
ograg
grami
grami
gramn
gramg
ramni
ramni
ramng
aimmi
aimmn
aimmg
mmmi
mmng
progra
progrm
progrm
progri
progrn
progrg
rogram
rogram
rograi
rogran
rograg
ogramm
ogrami
ogramn
ogramg
grammi
grammn
gramming
rammin
ramnig
aimming
program
program
progrmi
progran
prograg
rogramm
rogrami
rogramn
rogramg
ogrammi
ogrammn
ogramng
grammmi
grammng
rammmig
programm
programi
programn
programg
programi
programi
prograng
rogrammi
rogrammn
rogramng
ogrammmi
ogrammng
grammmig
programmi
programmn
programmg
rogrammmi
rogrammng
ogrammmng
programmin
programmig
rogrammmng
programming
```

**Result**

Thus, Program " **Subset of the word** " has been successfully executed

**Course:** DAA **Session:** Sum of Subsets **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. Chandu and Consecutive Letters**

Chandu is very fond of strings. (Or so he thinks!) But, he does not like strings which have same consecutive letters. No one has any idea why it is so. He calls these strings as Bad strings. So, Good strings are the strings which do not have same consecutive letters. Now, the problem is quite simple. Given a string S, you need to convert it into a Good String. You simply need to perform one operation - if there are two same consecutive letters, delete one of them. Input: The first line contains an integer T, denoting the number of test cases. Each test case consists of a string S, which consists of only lower case letters. Output: For each test case, print the answer to the given problem. Constraints: 1 <= T <= 10 1 <= |S| <= 30

**Source Code**

```
#include <stdio.h>
int main()
{
   int n=0,i=0,k=0;
   char a[30];
   scanf("%d",&n);

   for(;i<n;i++)
   {
       scanf("%s",a);
       for(;a[k]!='\0';k++)
       {
        if(a[k]!=a[k-1])
        {
         printf("%c",a[k]);
        }
       }
       k=0;
       printf("\n");
   }
   return 0;
}
```

**Sample Input**

```
3
abb
aaab
ababa
```

**Sample Output**

```
ab
ab
ababa
```

**Result**

Thus, Program " **Chandu and Consecutive Letters** " has been successfully executed

**Course:** DAA **Session:** Randomized algorithm **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. RANDOMIZED ALGORITHMS 8**

Write a function Add() that returns sum of two integers. The function should not use any of the arithmetic operators (+, ++, , -, .. etc). Sum of two bits can be obtained by performing XOR (^) of the two bits. Carry bit can be obtained by performing AND (&) of two bits. Above is simple Half Adder logic that can be used to add 2 single bits. We can extend this logic for integers. If x and y dont have set bits at same position(s), then bitwise XOR (^) of x and y gives the sum of x and y. To incorporate common set bits also, bitwise AND (&) is used. Bitwise AND of x and y gives all carry bits. We calculate (x & y) << 1 and add it to x ^ y to get the required result.

**Source Code**

```
#include<stdio.h>

int Add(int x, int y)
{
   // Iterate till there is no carry
   while (y != 0)
   {
      // carry now contains common set bits of x and y
      int carry = x & y;

      // Sum of bits of x and y where at least one of the bits is not set
      x = x ^ y;

      // Carry is shifted by one so that adding it to x gives the required sum
      y = carry << 1;
   }
   return x;
}

int main()
{
  int a,b;
  scanf("%d%d",&a,&b);
    printf("%d", Add(a,b));
    return 0;
}
```

**Sample Input**

```
5 6
```

**Sample Output**

```
11
```

**Result**

Thus, Program " **RANDOMIZED ALGORITHMS 8** " has been successfully executed

**Course:** DAA **Session:** Randomized algorithm **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. RANDOMIZED ALGORITHMS 5**

Given a number x, find next number with same number of 1 bits in its binary representation. For example, consider x = 12, whose binary representation is 1100 (excluding leading zeros on 32 bit machine). It contains two logic 1 bits. The next higher number with two logic 1 bits is 17 (100012).

### Source Code

```
#include<iostream>

using namespace std;

typedef unsigned int uint_t;

// this function returns next higher number with same number of set bits as x.
uint_t snoob(uint_t x)
{

  uint_t rightOne;
  uint_t nextHigherOneBit;
  uint_t rightOnesPattern;

  uint_t next = 0;

  if(x)
  {

    // right most set bit
    rightOne = x & -(signed)x;

    // reset the pattern and set next higher bit
    // left part of x will be here
    nextHigherOneBit = x + rightOne;

    // nextHigherOneBit is now part [D] of the above explanation.

    // isolate the pattern
    rightOnesPattern = x ^ nextHigherOneBit;

    // right adjust pattern
    rightOnesPattern = (rightOnesPattern)/rightOne;

    // correction factor
    rightOnesPattern >>= 2;

    // rightOnesPattern is now part [A] of the above explanation.

    // integrate new pattern (Add [D] and [A])
    next = nextHigherOneBit | rightOnesPattern;
  }

  return next;
}

int main()
{
  int x;
  cin>>x;
  cout<<snoob(x);

 // getchar();
  return 0;
}
```

### Sample Input

```
150
```

### Sample Output

```
153
```

### Result

Thus, Program " **RANDOMIZED ALGORITHMS 5** " has been successfully executed

**Course:** DAA **Session:** Randomized algorithm **Timestamp:** April 1, 2018, 12:52 pm **Register Number:** RA1611003010583

**Q. RANDOMIZED ALGORITHMS 4**

You can win three kinds of basketball points, 1 point, 2 points, and 3 points. Given a total score n, print out all the combination to compose n. Examples: For n = 1, the program should print following: 1 For n = 2, the program should print following: 1 1 2 For n = 3, the program should print following: 1 1 1 1 2 2 1 3 For n = 4, the program should print following: 1 1 1 1 1 1 2 1 2 1 1 3 2 1 1 2 2 3 1 and so on

### Source Code

```
// CPP program to Print all
// combinations of points that
// can compose a given number
#define MAX_POINT 3
#define ARR_SIZE 100
#include<stdio.h>

/* Utility function to print array arr[] */
void printArray(int arr[], int arr_size);

/* The function prints all combinations of numbers 1, 2, ...MAX_POINT
that sum up to n.
i is used in recursion keep track of index in arr[] where next
element is to be added. Initital value of i must be passed as 0 */
void printCompositions(int n, int i)
{

   /* array must be static as we want to keep track
   of values stored in arr[] using current calls of
   printCompositions() in function call stack*/
   static int arr[ARR_SIZE];

   if (n == 0)
   {
      printArray(arr, i);
   }
   else if(n > 0)
   {
      int k;
      for (k = 1; k <= MAX_POINT; k++)
      {
      arr[i]= k;
      printCompositions(n-k, i+1);
      }
   }
}

/* UTILITY FUNCTIONS */
/* Utility function to print array arr[] */
void printArray(int arr[], int arr_size)
{
   int i;
   for (i = 0; i < arr_size; i++)
      printf("%d ", arr[i]);
   printf("\n");
}

/* Driver function to test above functions */
int main()
{
   int n;
   scanf("%d",&n);

   printf("Different compositions formed by 1, 2 and 3 of %d are\n", n);
   printCompositions(n, 0);

   getchar();
   return 0;
}
```

### Sample Input

```
4
```

### Sample Output

```
Different compositions formed by 1, 2 and 3 of 4 are
1 1 1 1
1 1 2
1 2 1
1 3
2 1 1
2 2
3 1
```

### Result

Thus, Program " **RANDOMIZED ALGORITHMS 4** " has been successfully executed

**Course:** DAA **Session:** Randomized algorithm **Timestamp:** April 1, 2018, 12:53 pm **Register Number:** RA1611003010583

### Q. RANDOMIZED ALGORITHMS 1

Given two signed integers, write a function that returns true if the signs of given integers are different, otherwise false. For example, the function should return true -1 and +100, and should return false for -100 and -200. The function should not use any of the arithmetic operators. Let the given integers be x and y. The sign bit is 1 in negative numbers, and 0 in positive numbers. The XOR of x and y will have the sign bit as 1 iff they have opposite sign. In other words, XOR of x and y will be negative number number iff x and y have opposite signs

### Source Code

```
#include <iostream>
using namespace std;
bool oppositeSigns(int x, int y)
{
    return ((x ^ y) < 0);
}

int main()
{
    int x, y;
    cin>>x;
    cin>>y;
    if (oppositeSigns(x, y) == true)
       cout<<"Signs are opposite";
    else
      cout<<"Signs are not opposite";
    return 0;
}
```

### Sample Input

```
100
-120
```

### Sample Output

```
Signs are opposite
```

### Result

Thus, Program " **RANDOMIZED ALGORITHMS 1** " has been successfully executed

**Course:** DAA **Session:** Randomized algorithm **Timestamp:** April 1, 2018, 12:53 pm **Register Number:** RA1611003010583

**Q. RANDOMIZED ALGORITHMS 2**

Given a positive integer n, count the total number of set bits in binary representation of all numbers from 1 to n.

### Source Code

```
#include <stdio.h>

// A utility function to count set bits
// in a number x
unsigned int countSetBitsUtil(unsigned int x);

// Returns count of set bits present in all
// numbers from 1 to n
unsigned int countSetBits(unsigned int n)
{
   int bitCount = 0; // initialize the result
   int i;
   for (i = 1; i <= n; i++)
      bitCount += countSetBitsUtil(i);

   return bitCount;
}

// A utility function to count set bits
// in a number x
unsigned int countSetBitsUtil(unsigned int x)
{
   if (x <= 0)
      return 0;
   return (x % 2 == 0 ? 0 : 1) + countSetBitsUtil(x / 2);
}

// Driver program to test above functions
int main()
{
   int n;
  scanf("%d",&n);
   printf("Total set bit count is %d", countSetBits(n));
   return 0;
}
```

### Sample Input

```
8
```

### Sample Output

```
Total set bit count is 13
```

### Result

Thus, Program " **RANDOMIZED ALGORITHMS 2** " has been successfully executed

**Course:** DAA **Session:** Randomized algorithm **Timestamp:** April 1, 2018, 12:53 pm **Register Number:** RA1611003010583

**Q. MINMAX 2**

Given an array a={3, 20,100, 1, 2}, find the minimum and maximum element.

**Source Code**

```
#include <stdio.h>

#define MAX_SIZE 100   // Maximum array size

int main()
{
   int arr[MAX_SIZE];
   int i, max, min, size;

   /* Input size of the array */
  //  printf("Enter size of the array: ");
   scanf("%d", &size);

   /* Input array elements */
   //printf("Enter elements in the array: ");
   for(i=0; i<size; i++)
   {
      scanf("%d", &arr[i]);
   }


   /* Assume first element as maximum and minimum */
   max = arr[0];
   min = arr[0];

   /*
    * Find maximum and minimum in all array elements.
    */
   for(i=1; i<size; i++)
   {
      /* If current element is greater than max */
      if(arr[i] > max)
      {
         max = arr[i];
      }

      /* If current element is smaller than min */
      if(arr[i] < min)
      {
         min = arr[i];
      }
   }

   /* Print maximum and minimum element */
   printf("Minimum : %d\n", min);
   printf("Maximum : %d\n", max);

   return 0;
}
```

**Sample Input**

```
5
3 20 100 1 2
```

**Sample Output**

```
Minimum : 1
Maximum : 100
```

**Result**

Thus, Program " **MINMAX 2** " has been successfully executed

**Course:** DAA **Session:** Randomized algorithm **Timestamp:** April 1, 2018, 12:53 pm **Register Number:** RA1611003010583

### Q. RANDOMIZED ALGORITHMS 7

Given a integer x, write a function that multiplies x with 3.5 and returns the integer result. You are not allowed to use %, /, *. Examples: Input: 2 Output: 7 Input: 5 Output: 17 (Ignore the digits after decimal point) Solution: 1. We can get x*3.5 by adding 2*x, x and x/2. To calculate 2*x, left shift x by 1 and to calculate x/2, right shift x by 2.

### Source Code

```
#include <stdio.h>

int multiplyWith3Point5(int x)
{
  return (x<<1) + x + (x>>1);
}

/* Driver program to test above functions*/
int main()
{
  int x;
  scanf("%d",&x);
  printf("%d", multiplyWith3Point5(x));
 // getchar();
  return 0;
}
```

### Sample Input

6

### Sample Output

21

### Result

Thus, Program " **RANDOMIZED ALGORITHMS 7** " has been successfully executed

**Course:** DAA **Session:** Randomized algorithm **Timestamp:** April 1, 2018, 12:53 pm **Register Number:** RA1611003010583

**Q. RANDOMIZED ALGORITHMS 6**

Write a program to add one to a given number. You are not allowed to use operators like +, -, *, /, ++, etc. Examples: Input: 12 Output: 13 Input: 6 Output: 7 Yes, you guessed it right, we can use bitwise operators to achieve this. Following are different methods to achieve same using bitwise operators.

**Source Code**

```
#include <stdio.h>

int addOne(int x)
{
    int m = 1;

    // Flip all the set bits
    // until we find a 0
    while( x & m )
    {
        x = x ^ m;
        m <<= 1;
    }

    // flip the rightmost 0 bit
    x = x ^ m;
    return x;
}

/* Driver program to test above functions*/
int main()
{
  int num;
  scanf("%d",&num);
    printf("%d", addOne(num));
    //getchar();
    return 0;
}
```

**Sample Input**

12

**Sample Output**

13

**Result**

Thus, Program " **RANDOMIZED ALGORITHMS 6** " has been successfully executed

**Course:** DAA **Session:** Randomized algorithm **Timestamp:** April 1, 2018, 12:53 pm **Register Number:** RA1611003010583

**Q. RANDOMIZED ALGORITHMS 3**

A permutation, also called an arrangement number or order, is a rearrangement of the elements of an ordered list S into a one-to-one correspondence with S itself. A string of length n has n! permutation. Source: Mathword(http://mathworld.wolfram.com/Permutation.html) Below are the permutations of string ABC. ABC ACB BAC BCA CBA CAB

### Source Code

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

/* Function to swap values at two pointers */
void swap(char *x, char *y)
{
    char temp;
    temp = *x;
    *x = *y;
    *y = temp;
}

/* Function to print permutations of string
   This function takes three parameters:
   1. String
   2. Starting index of the string
   3. Ending index of the string. */
void permute(char *a, int l, int r)
{
   int i;
   if (l == r)
     printf("%s\n", a);
   else
   {
       for (i = l; i <= r; i++)
       {
          swap((a+l), (a+i));
          permute(a, l+1, r);
          swap((a+l), (a+i)); //backtrack
       }
   }
}

/* Driver program to test above functions */
int main()
{
    char str[20];
    scanf("%s",str);
    int n = strlen(str);
    permute(str, 0, n-1);
    return 0;
}
```

### Sample Input

ABC

### Sample Output

ABC
ACB
BAC
BCA
CBA
CAB

### Result

Thus, Program " **RANDOMIZED ALGORITHMS 3** " has been successfully executed

**Course:** DAA **Session:** Randomized algorithm **Timestamp:** April 1, 2018, 12:53 pm **Register Number:** RA1611003010583

**Q. MINMAX 1**

Given a list of N integers, your task is to select K integers from the list such that its unfairness is minimized. if (x1,x2,x3,,xk) are K numbers selected from the list N, the unfairness is defined as max(x1,x2,,xk)-min(x1,x2,,xk) where max denotes the largest integer among the elements of K, and min denotes the smallest integer among the elements of K. Input Format The first line contains an integer N. The second line contains an integer K. N lines follow. Each line contains an integer that belongs to the list N. Note: Integers in the list N may not be unique. Output Format An integer that denotes the minimum possible value of unfairness. Constraints 2<=N<=10^5 2<=K<=N 0=

**Source Code**

```
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
#include <climits>
using namespace std;
void merge(unsigned long long int a[],int l, int p,int h)
{
    int j,i,t,x,y,k;
    x = l;
    y = p+1;
    k = 0;
    unsigned long long int b[h-l+1];
    for(i=l;i<=h;i++)
    {
        if(x>p)
        {
            b[k] = a[y];
            k++;
            y++;
        }
        else if(y>h)
        {
            b[k] = a[x];
            k++;
            x++;
        }
        else if(a[x]<a[y])
        {
            b[k] = a[x];
            x++;
            k++;
        }
        else
        {
            b[k] = a[y];
            y++;
            k++;
        }
    }
    for(i=0;i<k;i++)
    {
        a[l] = b[i];
        l++;
    }
    return;
}

void sort(unsigned long long int a[],int l,int h)
{
    if(l<h)
    {
        int p = (l+h)/2;
        //printf("%d ",p);
        sort(a,l,p);
        sort(a,p+1,h);

        merge(a,l,p,h);
    }
    return;
}
int main()
{
   int N,K;
   cin >> N >> K;
   unsigned long long int a[N],d,min=100000000000,i;
   for (int i=0; i<N; i++)
   {
     cin>>a[i];
   }
   sort(a,0,N-1);
     for(i=0;i<=N-K;i++)
   {
     d = a[K+i-1] - a[i];
     if(d<min)
     {
         min = d;
     }
   }
   cout<<min<<endl;
   return 0;
}
```

**Sample Input**

```
3
2
21
66
11
```

**Sample Output**

```
10
```

**Result**

Thus, Program " **MINMAX 1** " has been successfully executed