Course: DATA-STRUCTURE **Session:** Memory Addressing **Timestamp:** October 5, 2017, 8:56 am **Register Number:** RA1611003010583

Q. Sort a stack using recursion (descending)

Given a stack, sort it using recursion. Use of any loop constructs like while, for..etc is not allowed

Source Code

```
#include <stdio.h>
int main()
{
  int n,arr[20],i,j,temp=0;
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
    scanf("%d",&arr[i]);
  }
  for(i=n-1;i>=0;i--)
  {
    printf("%d ",arr[i]);
  }
  printf("\n");
  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;i++)
  {
    printf("%d ",arr[i]);
  }
 return 0;
}
```

Sample Input

```
3
10 15 -1
```

Sample Output

```
-1 15 10
15 10 -1
```

Result

Thus, Program "**Sort a stack using recursion (descending)**" has been successfully executed

Q. Pair Wise Sum

You are given a sequence of N integer numbers A. Calculate the sum of Ai AND Aj for all the pairs (i, j) where i < j. The AND operation is the Bitwise AND operation, defined as in here. Input The first line of input consists of the integer N. The second line contains N integer numbers - the sequence A. Output Output the answer to the problem on the first line of the output

Source Code

```
#include <stdio.h>

int main(int argc, char const *argv[]) {
 long long int i, j, k, l, m, n;
 scanf("%lld", &n);
 long long int a[n];
 for(i = m = 0; i < n; i++)
  scanf("%lld", a + i);
 for(i = 0, j = 1; i < 31; i++) {
  for(k = l = 0; k < n; k++)
   l += ((a[k] >> i) & 1);
  l = ((l * (l - 1)) / 2);
  m += l * j; j <<= 1;
 }
 printf("%lld\n", m);
 return 0;
}
```

Sample Input

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

Sample Output

```
9
```

Result

Thus, Program " **Pair Wise Sum** " has been successfully executed

Q. Bombing

There are n+1 (0 < n <= 109, indexed 0..n) houses lying on a straight street. One day, they are bombed by an enemy. CodeChef is designing defence systems to protect the street from bombing. Each defense system can protect all houses in some interval, and a house can be protected by more than one system. A defence system can be also moved at any time. By agents, CodeChef knows the bombing schedule of the enemy. Each time they bomb a house and CodeChef wants to know how many systems are protecting that house. Input The first line contains two integers n, m (0 < m <= 250 000) Each of the m following lines follows one of these: P u v: CodeChef adds a defence system protecting all houses from the u-th house to the v-th house (0 <= u <= v <= n). All systems will be indexed by order of appearance in the input, starting from 1. M i d: CodeChef moves the i-th system by d houses, d < 0 means moving nearer the 0-th house, and d > 0 means moving away (that is, if the i-th system covers houses u through v, it is moved to instead cover houses w=u+d through t=v+d). After being moved the system will protect from the w-th house to t-th house where 0 <= w <= t <= n. B x: The enemy bombs the x-th house. Output For each time of bombing by the enemy (by the appearance in the input), output on a single line a number of systems protecting that house.

Source Code

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int compare(const void *a,const void *b)
{
  return *(int*)a-*(int*)b;
}
typedef struct __node
{
  int val,start,end;
}_node;
_node node[7000000];
int b[250005],x[250005],y[250005],s[250005][3],z[250005];
char c[250005][2];
int binarysearch(int arr[],int key,int low,int high)
{
  while(low<=high)
    {
      int mid=(low+high)>>1;
      if(arr[mid]==key)return mid;
      else if(key>arr[mid])low=mid+1;
      else high=mid-1;
    }
  return low;
}
void initialize(int i,int p,int q)
{
  node[i].start=p;
  node[i].end=q;
  node[i].val=0;
     if(node[i].start!=node[i].end)
    {
      initialize((i<<1)+1,p,(p+q)>>1);
      initialize((i<<1)+2,((p+q)>>1)+1,q);
       }
}
void update(int i,int p,int q,int value)
{
  if(node[i].end<p || node[i].start>q)
    return;
   if(node[i].start>=p && node[i].end<=q)
    {
      node[i].val+=value;
      return;
    }
  if(node[i].start!=node[i].end)
    {
      update((i<<1)+1,p,q,value);
      update((i<<1)+2,p,q,value);
    }
}
int query(int i,int p)
{
  if(node[i].start==node[i].end)return node[i].val;
  int x=(node[i].start+node[i].end)>>1;
  if(x>=p)
    return node[i].val+query((i<<1)+1,p);
  else
    return node[i].val+query((i<<1)+2,p);
}
int main()
{
  int i,n,m,bcount,obcount,j,ans,k;
  bcount=0;
  scanf("%d %d",&n,&m);
  for(i=0;i<m;i++)
    {
      scanf("%s",c[i]);
      if(strcmp(c[i],"P")==0)
      {
          scanf("%d %d",&s[i][0],&s[i][1]);
  }
      else if(strcmp(c[i],"B")==0)
  {
    scanf("%d",&s[i][0]);
    b[bcount++]=s[i][0];
  }
      else if(strcmp(c[i],"M")==0)
  {
    scanf("%d %d",&s[i][0],&s[i][1]);
  }
    }
  qsort(b,bcount,sizeof(int),&compare);
  obcount=0;
  for(i=1;i<bcount;i++)
    {
      if(b[i]!=b[i-1])
          {
  b[obcount++]=b[i-1];
    }
        }
    b[obcount++]=b[i-1];
  initialize(0,0,obcount-1);
  j=0;
  for(i=0;i<m;i++)
    {
      if(strcmp(c[i],"P")==0)
  {
    x[j]=binarysearch(b,s[i][0],0,obcount-1);
    if(b[x[j]]<s[i][0])
            x[j]++;
    y[j]=binarysearch(b,s[i][1],0,obcount-1);
    if(b[y[j]]>s[i][1])
      y[j]--;
    z[j]=i;
    update(0,x[j],y[j],1);
    j++;
  }
      else if(strcmp(c[i],"B")==0)
  {
    ans=binarysearch(b,s[i][0],0,obcount-1);
    printf("%d\n",query(0,ans));
  }
      else if(strcmp(c[i],"M")==0)
  {
    k=s[i][0]-1;
    update(0,x[k],y[k],-1);
    s[z[k]][0]+=s[i][1];
    s[z[k]][1]+=s[i][1];
    x[k]=binarysearch(b,s[z[k]][0],0,obcount-1);
       if(b[x[k]]<s[z[k]][0])
      x[k]++;
    y[k]=binarysearch(b,s[z[k]][1],0,obcount-1);
    if(b[y[k]]>s[z[k]][1])
      y[k]--;
    update(0,x[k],y[k],1);
  }
    }
  return 0;
}
```

Sample Input

```
7 5
P 1 4
P 3 5
B 3
M 2 1
B 3
```

Sample Output

```
2
1
```

Result

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

Q. Lucky Array

Chef Ciel bought a new video game for guests to kill time until their orders arrive. In this game, you are a fighter in a battle arena, and you will fight against a fighter called Shindannin. In Japan, many people believe that the next day will be a happy day if they beat Shindannin, otherwise the next day will be an unhappy day. Let's start by defintions of some variables. VA and VB denote your initial HP (health points) and Shindannin's initial HP respectively. SA and SB denote your strength and Shindannin's strength respectively. MA denotes your initial MP (magical points), which is used when you use a skill. Note that only you can use a skill in the battle. In each turn, the battle will go on as follows: Firstly, you can use a skill as many times as you like as long as your MP is positive. If you use a skill, your HP and Shindannin's HP are decreased by half and your MP is decreased by 1. If HP becomes a non-integer, HP will be rounded up to the nearest integer. To be more precise, new HP will be ceil(old HP / 2). Next, an integer s is chosen in [0, SA] uniformly randomly, and an integer t is chosen in [0, SB] uniformly randomly. Then, your HP is decresed by t, and Shindannin's HP is decreased by s simultaneously. If both fighter have positive HP, next turn will occur. When a fighter's HP is down to zero or a negative integer, this fighter loses. If both fighters' HP is down to zero or negative integer simultaneously, you will fight against Shindannin again with the same conditions, that is, HP and MP are recovered completely before next battle. There is no limit on the number of rematches. If you fight optimally, what is your winning percentage? Input An input contains 5 integers VA, VB, SA, SB and MA. Output Print the maximum winning percentage you can achieve. This value must have an absolute error no more than 10-6. Constraints 1<=VA, VB<=100 1<=SA, SB<=100 0<=MA<=5

Source Code

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
#define REP(i,a,b) for(i=a;i<b;i++)
#define rep(i,n) REP(i,0,n)

#define M 103

int VA, VB, SA, SB, MA;
double dp[6][211][211], sum[6][211][211];

int main(){
  int i,j,k;
  double a, b, c;
  double per;

  scanf("%d%d%d%d%d", &VA, &VB, &SA, &SB, &MA);

  per = 1.0 / ((SA+1)*(SB+1) - 1);

  a = 0; b = 1;
  while(b-a > 1e-6){
    c = (a+b)/2;

    rep(k,MA+1){
      rep(i,211) sum[k][0][i] = sum[k][i][0] = 0;
      REP(i,-M,VA+1) REP(j,-M,VB+1){
        sum[k][M+i+1][M+j+1] = sum[k][M+i][M+j+1] + sum[k][M+i+1][M+j] - sum[k][M+i][M+j];

        if(i <= 0 && j <= 0){
          dp[k][M+i][M+j] = c;
        } else if(i <= 0){
          dp[k][M+i][M+j] = 0;
        } else if(j <= 0){
          dp[k][M+i][M+j] = 1;
        } else {
          dp[k][M+i][M+j] = sum[k][M+i+1][M+j+1] - sum[k][M+i-SB][M+j+1] - sum[k][M+i+1][M+j-SA] + sum[k][M+i-SB][M+j-SA];
          dp[k][M+i][M+j] *= per;
        }

        if(k && dp[k][M+i][M+j] < dp[k-1][M+(i+1)/2][M+(j+1)/2]) dp[k][M+i][M+j] = dp[k-1][M+(i+1)/2][M+(j+1)/2];
        sum[k][M+i+1][M+j+1] += dp[k][M+i][M+j];
      }
    }

    if(c < dp[MA][M+VA][M+VB]) a = c; else b = c;
  }

  printf("%.10f\n",(a+b)/2);

  return 0;
}
```

Sample Input

```
5 5 3 3 0
```

Sample Output

```
0.4999995232
```

Result

Thus, Program " **Lucky Array** " has been successfully executed

Q. Monk And IQ

Monk and his P-1 friends recently joined a college. He finds that N students have already applied for different courses before him. Courses are assigned numbers from 1 to C. He and his friends will follow the following conditions when choosing courses:- They will choose the course i (1 <= i <= C), for which the value of z is minimum. Here, z = x*c where c is the number of students already enrolled in the course i and x is the sum of IQ of the last two students who enrolled in that course. If a single student has applied for a course, then the value of x will be that student's IQ. If no student has enrolled for that course, then value of x will be 0. If the value of z is same for two courses, then they will choose the course with the minimum course number. You need to find which courses Monk and his friends should take after following the above conditions. Note: Each of them will choose their courses, one at a time. Monk will choose his course first followed by his friends. Input: The first line contains the numbers C, P and N where C denotes the number of courses in that college, P denotes Monk and his friends and N denotes the number of students who have already applied for the courses. The next line consists of N space separated integers Y[i] which denotes the IQ of the ith student. Here, the ith student chooses the ith course. The next line consists of P space separated integers X[i] which denotes the IQ of Monk and his friends. Output: Print P space separated integers in a line which denotes the course number which Monk and his friends have applied for. Constraints: 1 <= C <= 100000 1 <= P <= 100000 1 <= N <= C 1 <= Y[i],X[i] <= 100000

Source Code

```
#include<stdio.h>
#define LIMIT 100005
#define MOD 1000000007
#define left(i) (2*i+1)
#define right(i) (2*i+2)

typedef struct {
 int inrolled;
 int laststudent;
 int secondlaststudent;
 int courseno;
 long long int z;
}courseinfo;

courseinfo heap[LIMIT];
int Y[LIMIT];
int X[LIMIT];
int C, P, N;

int printheap(int N){
 int i;
 for(i = 0; i < N; i++){
  printf("[%d | %d | %lld | %d | %d]\t\t", heap[i].courseno, heap[i].inrolled, heap[i].z, heap[i].laststudent, heap[i].secondlaststudent);
 }
 printf("\n\n");
 return 0;
}
int swap(courseinfo *a, courseinfo *b){
 courseinfo temp;
 temp = *a;
 *a = *b;
 *b = temp;
 return 0;
}

int minheapify(int i, int heapsize){
 int smallest = i;
 if(left(i) < heapsize && heap[left(i)].z <= heap[smallest].z){
  if(heap[left(i)].z != heap[smallest].z)
   smallest = left(i);
  else if(heap[left(i)].courseno < heap[smallest].courseno)
   smallest = left(i);
 }
 if(right(i) < heapsize && heap[right(i)].z <= heap[smallest].z){
  if(heap[right(i)].z != heap[smallest].z)
   smallest = right(i);
  else if(heap[right(i)].courseno < heap[smallest].courseno)
   smallest = right(i);
 }

 if(i != smallest){
   swap(&heap[i], &heap[smallest]);
   minheapify(smallest, heapsize);
 }
 return 0;
  }

int buildheap(int C){
 int i = C/2 -1;
 for(i; i >= 0; i--)
  minheapify(i, C);
 minheapify(0, C);
 return 0;
}

int main(){
 scanf("%d%d%d", &C, &P, &N);
 int i;
 for(i = 1; i <= N; i++){
  scanf("%d", &Y[i]);
  heap[i-1].inrolled = 1;
  heap[i-1].courseno = i;
  heap[i-1].laststudent = Y[i];
  heap[i-1].secondlaststudent = 0;
  heap[i-1].z = 1*Y[i];
    }
 for(i; i <= C; i++){
  heap[i-1].inrolled = 0;
  heap[i-1].courseno = i;
  heap[i-1].laststudent = 0;
  heap[i-1].secondlaststudent = 0;
  heap[i-1].z = 0;
 }
 //printheap(C);
 buildheap(C);
 //printheap(C);
 for(i  = 1; i <= P; i++){
  scanf("%d", &X[i]);
 }
 for(i = 1; i <= P; i++){
  printf("%d ", heap[0].courseno);
  heap[0].inrolled++;
  heap[0].secondlaststudent = heap[0].laststudent;
  heap[0].laststudent = X[i];
  heap[0].z = (heap[0].inrolled * (heap[0].laststudent + heap[0].secondlaststudent))%MOD;
  minheapify(0, C);
 }
 return 0;
}
```

Sample Input

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

Sample Output

```
5 4 1 3
```

Result

Thus, Program " **Monk And IQ** " has been successfully executed

Q. Monk and BST

Monk as always has brought a new task for Fredo. He asks Fredo to create a Binary Search Tree (BST) with L levels and 2L -1 nodes. Let the sum of all node values in the BST be M. He further adds that M should be smallest possible integer greater than S, where S is an integer given by Monk. He states the rules of creating BST as follows: The left sub-tree contains only nodes with values less than or equal to the parent node; the right sub-tree contains only nodes with values greater than the parent node. If a node has level i, then the subtree rooted at that node should have exactly 2L-i number of distinct values in the subtree. Note that it is the number of distinct values in the subtree and not the number of nodes in the subtree. If a is the smallest value in the tree and b is the largest value, then all values from a to b must come atleast once in the tree. Level of root is 1, next level is 2 and so on. Now, he will ask two type of queries to Fredo. Type 0: Find the closest node to root whose value is equal to val and print path to that node from the root. If root has value equal to val, print "root". Else print "l" when we visit left child of any node and "r" when we visit right child of any node. Type 1: Tell the value of kth node in the tree. The nodes are numbered as: enter image description Here 1,2 and so on are node numbers and A,B etc. are values of nodes. Finally, he will ask Fredo Q queries, each query belonging to one of the types mentioned above. Since, Fredo is new to this concept, help him complete this task. Input Format: First line consists of two integers L and S as described in the question. Second line consists of Q, denoting the number of queries. Each of the following Q lines consists of two integers. The first integer denoting the type of query and second denoting either val or k as described in the queries. Output Format: For each query, print the required answer in a separate line. Input Constraints: 1<=L<=30 1<=S<=1018 1<=Q<=105 0<=type<=1 1<=k<=2L-1 val is in between minimum and maximum value in tree(inclusive). Values of S, val and k are such that answer would always exist for them. Node values will be non-negative for all inputs. M will never exceed 2 *1018

Source Code

```
#include <stdio.h>

long long power(int x, int y) {
 long long p;

 if (y == 0)
  return 1;
 p = power(x, y / 2);
 p *= p;
 if (y % 2 == 1)
  p *= x;
 return p;
}

int query0(int i, long long val, long long sl, long long sr) {
 long long x = (sl + sr - 1) / 2;

 if (val == x)
  return i;
 if (val < x)
  return query0(i * 2, val, sl, (sl + sr) / 2);
 else
  return query0(i * 2 + 1, val, (sl + sr) / 2, sr);
}

long long query1(int i, int j, int k, long long sl, long long sr) {
 return j == k ? (sl + sr - 1) / 2 : (k & 1 << (i - 1)) == 0
  ? query1(i - 1, j * 2, k, sl, (sl + sr) / 2)
  : query1(i - 1, j * 2 + 1, k, (sl + sr) / 2, sr);
}

void print(int i) {
 if (i > 1) {
  print(i / 2);
  printf("%c", i % 2 == 0 ? 'l' : 'r');
 }
}

int main() {
 int l, q, n, cnt;
 long long s, p, start;

 scanf("%d%lld", &l, &s);
 cnt = power(2, l - 1);
 n = power(2, l) - 1;
 p = power(4, l - 1) - n;
 start = p > s ? 0 : (s - p) / n + 1;
 scanf("%d", &q);
 while (q-- > 0) {
  int type, k;
  long long val;

  scanf("%d", &type);
  if (type == 0) {
   scanf("%lld", &val);
   if (val == (start + start + cnt - 1) / 2)
    printf("root\n");
   else {
    print(query0(1, val, start, start + cnt));
    printf("\n");
   }
  } else {
   int x;

   scanf("%d", &k);
   for (x = 0; (long long) 1 << x <= k; x++)
    ;
   printf("%lld\n", query1(x - 1, 1, k, start, start + cnt));
  }
 }
 return 0;
}
```

Sample Input

```
1 9
5
0 10
0 10
1 1
0 10
1 1
```

Sample Output

```
root
root
10
root
10
```

Result

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

Q. Monk and Philosopher's Stone

Harry Potter wants to get the Philosopher's stone to protect it from Snape. Monk being the guard of Philosopher's Stone is very greedy and has a special bag, into which he can add one gold coin at a time or can remove the last gold coin he added. Monk will sleep, once he will have the enough number of gold coins worth amount X. To help Harry, Dumbledore has given a same kind of bag to Harry (as of Monk) with N gold coins each having worth A[i] where i range from 1<=i<=N. Dumbledore also gave him a set of instructions which contains two types of strings: 1) "Harry" (without quotes): It means Harry will remove ith coin from his bag and throw it towards Monk and Monk will add it in his bag, where i will start from 1 and go up to N. 2) "Remove" (without quotes): it means Monk will remove the last coin he added in his bag. Once the worth of the coins in Monk's bag becomes equal to X, Monk will go to sleep. In order to report Dumbledore, Harry wants to know the number of coins in Monk's bag, the first time their worth becomes equal to X. Help Harry for the same and print the required number of coins. If the required condition doesn't occur print "-1" (without quotes). Input: The first line will consists of one integer N denoting the number of gold coins in Harry's Bag. Second line contains N space separated integers, denoting the worth of gold coins. Third line contains 2 space separated integers Q and X, denoting the number of instructions and the value of X respectively. In next Q lines, each line contains one string either "Harry" (without quotes) or "Remove" (without quotes). Output: In one line, print the the number of coins in the Monk's bag, the first time their worth becomes equal to X. Constraints: 1<=N<=104 1<=A[i] <=104 1<=Q<=105 1<=X<=107

Source Code

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

int main()
{
 int n,i,q,x,f=0,sum=0;
 string s;
 stack<int>st;
    cin>>n;
    int arr[n];
    for(i=0;i<n;i++)
    cin>>arr[i];
    cin>>q>>x;
    i=0;
    while(q--)
    {
     cin>>s;
     if(s=="Harry")
     {
      st.push(arr[i]);
      sum=sum+arr[i];
      i++;
     }
     else
     {
      sum=sum-st.top();
      st.pop();
     }
     if(sum==x)
     {
      f=1;
      cout<<st.size();
      break;
        }
      }
    if(f==0)
    cout<<"-1";
    return 0;
}
```

Sample Input

```
4
3 1 1 4
6 7
Harry
Harry
Harry
Remove
Remove
Harry
```

Sample Output

```
2
```

Result

Thus, Program "**Monk and Philosopher's Stone**" has been successfully executed

Q. Finding Square Roots

In olden days finding square roots seemed to be difficult but nowadays it can be easily done using in-built functions available across many languages . Assume that you happen to hear the above words and you want to give a try in finding the square root of any given integer using in-built functions. So here's your chance. Input The first line of the input contains an integer T, the number of test cases. T lines follow. Each T contains an integer N whose square root needs to be computed. Output For each line of input output the square root of the input integer. Constraints 1<=T<=20 1<=N<=10000

Source Code

```
#include <stdio.h>
#include<math.h>
int main()
{
    int num1;
    int test, count;
    count = 0;
    scanf("%d", &test);

 while( count < test ){

    scanf("%d", &num1);
    printf("%d\n", (int)sqrt(num1));

    count++;
 }

 return 0;
}
```

Sample Input

```
3
10
5
10000
```

Sample Output

```
3
2
100
```

Result

Thus, Program " **Finding Square Roots** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Memory Addressing **Timestamp:** October 5, 2017, 8:58 am **Register Number:** RA1611003010583

Q. Mancunian And Colored Tree

After a hectic week at work, Mancunian and Liver bird decide to go on a fun weekend camping trip. As they were passing through a forest, they stumbled upon a unique tree of N nodes. Vertices are numbered from 1 to N. Each node of the tree is assigned a color (out of C possible colors). Being bored, they decide to work together (for a change) and test their reasoning skills. The tree is rooted at vertex 1. For each node, they want to find its closest ancestor having the same color. Input format The first line contains two integers N and C denoting the number of vertices in the tree and the number of possible colors. The second line contains N-1 integers. The i th integer denotes the parent of the i+1 th vertex. The third line contains N integers, denoting the colors of the vertices. Each color lies between 1 and C inclusive. Output format Print N space-separated integers. The ith integer is the vertex number of lowest ancestor of the ith node which has the same color. If there is no such ancestor, print -1 for that node. Constraints 1<=N<=100,000 1<=C<=100,000

Source Code

```
#include <stdio.h>

int main()
{
   int n,c,aa[100001],cc[100001],i;
   scanf("%d %d",&n,&c);
   aa[0]=0;aa[1]=0;
   for(i=2;i<=n;i++)
    scanf("%d",&aa[i]);
   for(i=1;i<=n;i++)
    scanf("%d",&cc[i]);
   //for(i=1;i<=n;i++)
   // printf("%d %d*",aa[i],cc[i]);
   for(i=1;i<=n;i++)
   { int f=0;
    int p=aa[i];
    while(p)
    { //printf("%d %d\n",p,i);
     if(cc[p]==cc[i])
     { printf("%d ",p);
      f=1;
      break;
     }
     p=aa[p];
    }
    if(f==0)
     printf("-1 ");
   }

   return 0;
}
```

Sample Input

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

Sample Output

```
-1 -1 -1 1 3
```

Result

Thus, Program "**Mancunian And Colored Tree**" has been successfully executed

Course: DATA-STRUCTURE **Session:** Memory Addressing **Timestamp:** October 5, 2017, 8:58 am **Register Number:** RA1611003010583

Q. The Stock Span Problem

The stock span problem is a financial problem where we have a series of n daily price quotes for a stock and we need to calculate span of stocks price for all n days. The span Si of the stocks price on a given day i is defined as the maximum number of consecutive days just before the given day, for which the price of the stock on the current day is less than or equal to its price on the given day. For example, if an array of 7 days prices is given as {100, 80, 60, 70, 60, 75, 85}, then the span values for corresponding 7 days are {1, 1, 1, 2, 1, 4, 6}

Source Code

```
#include <stdio.h>
int i,j;
void calculateSpan(int price[], int n, int S[])
{
   S[0] = 1;
   for (i = 1; i < n; i++)
   {
      S[i] = 1;
      for (j = i-1; (j>=0)&&(price[i]>=price[j]); j--)
          S[i]++;
   }
}
void printArray(int arr[], int n)
{
    for (i = 0; i < n; i++)
      printf("%d ", arr[i]);
}

// Driver program to test above function
int main()
{
  int a;
  scanf("%d",&a);
  int price[a];
  for(i=0;i<a;i++)
    scanf("%d",&price[i]);
    int n = sizeof(price)/sizeof(price[0]);
    int S[n];
    calculateSpan(price, n, S);
    printArray(S, n);

    return 0;
}
```

Sample Input

```
7
100 80 60 70 60 75 85
```

Sample Output

```
1 1 1 2 1 4 6
```

Result

Thus, Program "**The Stock Span Problem**" has been successfully executed

Course: DATA-STRUCTURE **Session:** Arrays **Timestamp:** October 5, 2017, 8:58 am **Register Number:** RA1611003010583

Q. Left Rotation

A left rotation operation on an array of size n shifts each of the array's elements 1 unit to the left. For example, if 2 left rotations are performed on array [1,2,3,4,5], then the array would become [3,4,5,1,2]. Given an array of n integers and a number,d, perform d left rotations on the array. Then print the updated array as a single line of space-separated integers. Input Format The first line contains two space-separated integers denoting the respective values of n (the number of integers) and d (the number of left rotations you must perform). The second line contains n space-separated integers describing the respective elements of the array's initial state. Output Format Print a single line of n space-separated integers denoting the final state of the array after performing d left rotations.

Source Code

```
#include<stdio.h>


void leftRotatebyOne(long long int arr[],long long int n);

void leftRotate(long long int arr[],long long int d,long long int n)
{
  long long int i;
  for (i = 0; i < d; i++)
    {
    leftRotatebyOne(arr,n);
    }
}

void leftRotatebyOne(long long int arr[],long long int n)
{
  long long int i, temp;
  temp = arr[0];
  for (i = 0; i < n-1; i++)
  {
     arr[i] = arr[i+1];
  }
    arr[i] = temp;
}

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

int main()
{
    long long int n,i,d;
    scanf("%lld %lld",&n,&d);
    long long int ara[n];
    for(i=0;i<n;i++)
    {
        scanf("%lld",&ara[i]);
    }
    leftRotate(ara,d,n);
    printArray(ara,n);
    return 0;
}
```

Sample Input

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

Sample Output

```
5 1 2 3 4
```

Result

Thus, Program " **Left Rotation** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Arrays **Timestamp:** October 5, 2017, 8:58 am **Register Number:** RA1611003010583

Q. Largest Number formed from an Array

Given a list of non negative integers, arrange them in such a manner that they form the largest number possible. The result is going to be very large, hence return the result in the form of a string. Input: The first line of input consists number of the test cases. The description of T test cases is as follows: The first line of each test case contains the size of the array, and the second line has the elements of the array. Output: In each separate line print the largest number formed by arranging the elements of the array in the form of a string. Constraints: 1<=T<=70 1<=N<=100 0<=A[i]<=1000

Source Code

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

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

static int compare(const void *p, const void *q) {
 char *s = (char *)p;
 char *t = (char *)q;
 char ss = *s;
 char ts = *t;
 int ls = strlen(s);
 int lt = strlen(t);
 int i = 0;
 for(i = 0; i < min(lt, ls) && *s == *t; i++, s++, t++);
 if(i == ls) {
  return *t - ts;
 } else if(i == lt) {
  return *s - ss;
 }
 return *t - *s;
}

int main(int argc, char const *argv[]) {
 int t;
 scanf("%d", &t);
 while(t-- ) {
  int i, n;
  scanf("%d", &n);
  char str[n][5];
  for(i = 0; i < n; i++)
   scanf("%s", str[i]);
  qsort(str, n, sizeof(char) * 5, compare);
  for(i = 0; i < n; i++)
   printf("%s", str[i]);
  printf("\n");
 }
 return 0;
}
```

Sample Input

```
2
5
3 30 34 5 9
4
54 546 548 60
```

Sample Output

```
9534330
6054854654
```

Result

Thus, Program "**Largest Number formed from an Array**" has been successfully executed

Course: DATA-STRUCTURE **Session:** Arrays **Timestamp:** October 5, 2017, 8:58 am **Register Number:** RA1611003010583

Q. Search in a Rotated Array

Given a sorted and rotated array (rotated at some point) A[ ], and given an element K, the task is to find the index of the given element K in the array A[ ]. The array has no duplicate elements. If the element does not exist in the array, print -1. Input: The first line of the input contains an integer T, depicting the total number of test cases. Then T test cases follow. Each test case consists of three lines. First line of each test case contains an integer N denoting the size of the given array. Second line of each test case contains N space separated integers denoting the elements of the array A[ ]. Third line of each test case contains an integer K denoting the element to be searched in the array. Output: Corresponding to each test case, print in a new line, the index of the element found in the array. If element is not present, then print -1. Constraints: 1<=T<=100 1<=N<=100005 0<=A[i]<=10000005 1<=k<=100005

Source Code

```
#include <stdio.h>
int main()
{
  int T;
  int n,i,j,f=0,a[100];
  int key,p;
  scanf("%d",&T);
  //while(T--)
  for(j=0;j<T;j++)
  { f=0;
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
      scanf("%d",&a[i]);
    }
    scanf("%d",&key);
    for(i=0;i<n;i++)
    {
      if(a[i]==key)
      {
        f=1;
        p=i;
       // printf("%d\n",i);

      }
    }
    if(f==1)
   {
      printf("%d\n",p);
   }
    else if(f==0)
    {
      printf("-1\n");
    }
  }

 return 0;
}
```

Sample Input

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

Sample Output

```
5
1
-1
```

Result

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

Course: DATA-STRUCTURE **Session:** Arrays **Timestamp:** October 5, 2017, 8:59 am **Register Number:** RA1611003010583

Q. Move all zeroes to end of array

Given an array of random numbers, Push all the zeros of a given array to the end of the array. For example, if the given arrays is {1, 9, 8, 4, 0, 0, 2, 7, 0, 6, 0}, it should be changed to {1, 9, 8, 4, 2, 7, 6, 0, 0, 0, 0}. Input: The first line contains an integer 'T' denoting the total number of test cases. In each test cases, First line is number of elements in array 'N' and second its values. Output: Print the array after shifting all 0's at the end. Note: An extra space is expected at the end after output of every test case Constraints: 1 <= T <=100 1 <= N <=1000 0 <= a[i] <=100

Source Code

```
#include<iostream>
using namespace std;
int main()
{
   int t,n,a[100],c=0,d=0,i,b[100];
   cin>>t;
   while(t-->0)
    {
     cin>>n;
       for(i=0;i<n;i++)
        {
           cin>>a[i];
           if(a[i]!=0)
            {
               b[c++]=a[i];
            }
           else
           {
               d++;
            }
        }
       for(i=0;i<d;i++)
       {
         b[c++]=0;
       }
       for(i=0;i<c;i++)
       {
         cout<<b[i]<<" ";
       }
       c=0;
       d=0;
    }
  return 0;
}
```

Sample Input

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

Sample Output

```
3 5 4 0 0
```

Result

Thus, Program "**Move all zeroes to end of array**" has been successfully executed

Course: DATA-STRUCTURE **Session:** Arrays **Timestamp:** October 5, 2017, 8:59 am **Register Number:** RA1611003010583

Q. Product array puzzle

Given an array arr[ ] of n integers, construct a Product Array prod[ ] (of same size) such that prod[i] is equal to the product of all the elements of arr[ ] except arr[i]. Input: The first line of input contains an integer T denoting the number of test cases. The first line of each test case is N,N is the size of array. The second line of each test case contains N input A[i]. Output: Print the Product array prod[ ]. Constraints: 1<=T<=10 1<=N<=15 1<=C[i]<=20

Source Code

```
#include <stdio.h>
int main()
{
 int T,n,c,arr[50];
  int i,j,l[50],r[50];
  int p[50];
  scanf("%d",&T);

  while(T--)
  {
      scanf("%d",&n);
      for(i=0;i<n;i++)
  {
   scanf("%d",&arr[i]);
  }
  l[0]=1;
  r[n-1]=1;
  for(i = 1; i < n; i++)
  {
    l[i] = arr[i-1]*l[i-1];
  }
  for(j = n-2; j >=0; j--)
  {
    r[j] = arr[j+1]*r[j+1];
  }
  for (i=0; i<n; i++)
  {
    p[i] = l[i] * r[i];
  }
  for(i=0;i<n;i++)
  {
    printf("%d ",p[i]);
  }
    printf("\n");
  }
  return 0;
}
```

Sample Input

```
2
5
10 3 5 6 2
2
12 20
```

Sample Output

```
180 600 360 300 900
20 12
```

Result

Thus, Program " **Product array puzzle** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Arrays **Timestamp:** October 5, 2017, 8:59 am **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 1 to N. Output Format A single line containing maximum value in the updated list

Source Code

```
#include <iostream>

using namespace std;
const int NMAX = 1e7+2;
long long a[NMAX];
int main()
{
    int n, m;
    cin >> n >> m;
    for(int i=1;i<=m;++i){
        int x, y, k;
        cin >> x >> y >> k;
        a[x] += k;
        a[y+1] -= k;
    }
    long long x = 0,sol=-(1LL<<60);
    for(int i=1;i<=n;++i){
        x += a[i];
        sol = max(sol,x);
    }
    cout<<sol<<"\n";
    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: DATA-STRUCTURE **Session:** Arrays **Timestamp:** October 5, 2017, 9:00 am **Register Number:** RA1611003010583

Q. Leaders in an array

Write a program to print all the LEADERS in the array. An element is leader if it is greater than all the elements to its right side. The rightmost element is always a leader. Input: The first line of input contains an integer T denoting the number of test cases. The description of T test cases follows. The first line of each test case contains a single integer N denoting the size of array. The second line contains N space-separated integers A1, A2, ..., AN denoting the elements of the array. Output: Print all the leaders. Constraints: 1 <= T <= 100 1 <= N <= 100 0 <= A[i]<=100

Source Code

```
#include <iostream>
using namespace std;
int main()
{
 int i,j,n,flag,t,a[100];
 cin>>t;
 while(t>0)
 {
  cin>>n;
  for(i=0;i<n;i++)
  {
    cin>>a[i];
  }
  for(i=0;i<n-1;i++)
  {
    flag=0;
    for(j=i+1;j<n;j++)
    {
      if(a[i]<a[j])
      {
        flag=1;
      }
      else
      {
        flag=flag;
      }
    }
    if(flag==0)
    {
      cout<<a[i]<<" ";
    }
  }
  cout<<a[n-1]<<"\n";
  t--;
 }

 return 0;
}
```

Sample Input

```
2
6
16 17 4 3 5 2
5
1 2 3 4 0
```

Sample Output

```
17 5 2
4 0
```

Result

Thus, Program "**Leaders in an array**" has been successfully executed

Course: DATA-STRUCTURE **Session:** Arrays **Timestamp:** October 5, 2017, 9:00 am **Register Number:** RA1611003010583

Q. Minimum element in a sorted and rotated array

A sorted array A[ ] with distinct elements is rotated at some unknown point, the task is to find the minimum element in it. Expected Time Complexity: O(Log 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 consist of two lines. The first line of each test case consists of an integer N, where N is the size of array. The second line of each test case contains N space separated integers denoting array elements. Output: Corresponding to each test case, in a new line, print the minimum element in the array. Constraints: 1<=T<=100 1<=N<=500 1<=A[i]<=1000

Source Code

```
#include <stdio.h>
int main()
{
int a[10],i,n,t,min,j;
  scanf("%d",&t);
  for(i=0;i<t;i++){
    scanf("%d",&n);
    for(j=0;j<n;j++){

    scanf("%d",&a[j]);}
    min=a[0];
    for(j=1;j<n;j++){
      if(min>a[j])
        min=a[j];
      }
    printf("%d",min);
    }

 return 0;
}
```

Sample Input

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

Sample Output

```
1
```

Result

Thus, Program " **Minimum element in a sorted and rotated array**" has been successfully executed

Course: DATA-STRUCTURE **Session:** Arrays **Timestamp:** October 5, 2017, 9:00 am **Register Number:** RA1611003010583

Q. Find median in a stream

Given an input stream of n integers the task is to insert integers to stream and print the median of the new stream formed by each insertion of x to the stream. Example Flow in stream : 5, 15, 1, 3 5 goes to stream --> median 5 (5) 15 goes to stream --> median 10 (5, 15) 1 goes to stream --> median 5 (5, 15, 1) 3 goes to stream --> median 4 (5, 15, 1, 3) Input: The first line of input contains an integer N denoting the no of elements of the stream. Then the next N lines contains integer x denoting the no to be inserted to the stream. Output: For each element added to the stream print the floor of the new median in a new line. Constraints: 1<=N<=10^5+7 1<=x<=10^5+7

Source Code

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

  for(i=0;i<n;i++){
    scanf("%d",&a[i]);
  for(j=0;j<=i;j++){
    for(k=j+1;k<=i;k++){
      if(a[j]>a[k]){
        temp=a[j];
        a[j]=a[k];
        a[k]=temp;
      }
    }
  }
    if((i+1)%2!=0)
      printf("%d\n",a[(i+1)/2]);
    else{
      int med=(a[(i/2)]+a[(i+1)/2])/2;
      printf("%d\n",med);
    }
  }

 return 0;
}
```

Sample Input

```
4
5
15
1
3
```

Sample Output

```
5
10
5
4
```

Result

Thus, Program "**Find median in a stream**" has been successfully executed

Course: DATA-STRUCTURE	**Session:** Searching Techniques	**Timestamp:** October 5, 2017, 9:00 am	**Register Number:** RA1611003010583

Q. Triangle Games

You are given n triangles. You are required to find how many triangles are unique out of given triangles. For each triangle you are given three integers a,b,c , the sides of a triangle. A triangle is said to be unique if there is no other triangle with same set of sides. Note : It is always possible to form triangle with given sides. INPUT: First line contains n, the number of triangles. Each of next n lines contain three integers a,b,c (sides of a triangle). Output: print single integer, the number of unique triangles.

Source Code

```
#include<stdio.h>
void swap(long long int *a, long long int *b)
{
  long long int t;
  t=*a;
  *a=*b;
  *b=t;
}

void qsort(long long int items[][4], long long int left, long long int right)
{
    register long long int i,j,k;
    long long int x, y;

    i = left; j = right;
    x = items[(left+right)/2][3];
    do
    {
      while((items[i][3] < x) && (i < right))
        i++;
      while((x < items[j][3]) && (j > left))
        j--;
      if(i <= j)
      {
          y = items[i][0];items[i][0] = items[j][0];items[j][0] = y;
  y = items[i][1];items[i][1] = items[j][1];items[j][1] = y;
  y = items[i][2];items[i][2] = items[j][2];items[j][2] = y;
  y = items[i][3];items[i][3] = items[j][3];items[j][3] = y;
          i++;
        j--;
      }
    } while(i <= j);

    if(left < j)
          qsort(items, left, j);
    if(i < right)
          qsort(items, i, right);
}

int main()
{
  long long int i,k,j,n,z,g,count=0;
  long long int p,q,r;
  long long int arr[100000][4]={0};
  scanf("%lld",&n);
  for(i=0;i<n;i++)
  {
        scanf("%lld %lld %lld",&arr[i][0],&arr[i][1],&arr[i][2]);
        arr[i][3]=arr[i][0]+arr[i][1]+arr[i][2];
        if (arr[i][0] > arr[i][1]) swap(&arr[i][0],&arr[i][1]);
  if (arr[i][1] > arr[i][2]) swap(&arr[i][1],&arr[i][2]);
  if (arr[i][0] > arr[i][1]) swap(&arr[i][0],&arr[i][1]);
  }
  qsort(arr,0,n-1);
  i=0;
  while(1)
  {
  p=arr[i][0];
  q=arr[i][1];
  r=arr[i][2];
  k=arr[i][3];
  j=i;
  z=0;
    if(p==0)
    {
    ++i;
    if(i==n)
        break;
          else
      continue;
  }
  while(arr[++j][3]==k && i<n)
  {
        if(arr[i][0]!=0 && arr[i][0]==p && arr[i][1]==q && arr[i][2]==r)
        {
                        arr[i][0]=0;
          z++;
        }
  }
  if(z!=i-1-j && z>=1)
  {
          i=j+1;
          continue;
  }
  else if(z==0)
  {
        count++;
        if(i!=j+1)
        {
          i=j+1;
          continue;
            }
  }
        if(i==n)
          break;
}
  printf("%lld",count);
  return 0;
}
```

Sample Input

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

Sample Output

```
1
```

Result

Thus, Program " **Triangle Games** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Searching Techniques	**Timestamp:** October 5, 2017, 9:00 am	**Register Number:** RA1611003010583

Q. Array Game

You are given an array A of size N, and Q queries to deal with. For each query, you are given an integer X, and you're supposed to find out if X is present in the array A or not. Input: The first line contains two integers, N and Q, denoting the size of array A and number of queries. The second line contains N space separated integers, denoting the array of elements Ai. The next Q lines contain a single integer X per line. Output: For each query, print YES if the X is in the array, otherwise print NO

Source Code

```
#include <stdio.h>
void quicksort(int [10321],int,int);
int main()
{
  long long int c, first, last, middle, n, search,k=0,d,swap,q,i,x;
   scanf("%lld %lld",&n,&q);
   int a[n+1],b[q+1];

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

    for(c=0;c<q;c++)
    {
    scanf("%d", &b[c]);
    }
    quicksort(a,0,n-1);
  first = 0;
    last = n - 1;
    middle = (first+last)/2;
   while(k<q)
   {
    while (first <= last) {
       if (a[middle] < b[k])
         first = middle + 1;
       else if (a[middle] == b[k]) {
         printf("YES\n");k++;
         first=0;
         last=n-1;
         break;
       }
       else
         last = middle - 1;

       middle = (first + last)/2;
    }
    if (first > last)
    {
       printf("NO\n");
        first=0;
         last=n-1;
       k++;
    }
   }
    return 0;
}

 void quicksort(int x[10],int first,int last){
    int pivot,j,temp,i;

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

        while(i<j){
            while(x[i]<=x[pivot]&&i<last)
                i++;
            while(x[j]>x[pivot])
                j--;
           if(i<j){
                temp=x[i];
                 x[i]=x[j];
                 x[j]=temp;
            }
        }

        temp=x[pivot];
        x[pivot]=x[j];
        x[j]=temp;
        quicksort(x,first,j-1);
        quicksort(x,j+1,last);

    }
    }
```

Sample Input

```
5 10
50 40 30 20 10
10
20
30
40
50
60
70
80
90
100
```

Sample Output

```
YES
YES
YES
YES
YES
NO
NO
NO
NO
NO
```

Result

Thus, Program " **Array Game** " has been successfully executed

Q. Puzzle

A Puzzle is a game, problem, or toy that tests a person's ingenuity. In a puzzle, one is required to put pieces together, in a logical way, in order to arrive at the correct solution of the puzzle. There are different types of puzzles for different ages.Puzzles are often devised as a form of entertainment but they can also arise from serious mathematical or logistical problems. Ananya and Bhavya are on their way to Goa. On their trip they visit a lot of different places. But one place left Ananya awe struck. She saw a lot of round big stones laid in a straight line with a number encrypted on each of them. She called this number as the value of the stones. Since they both are coders they could not leave their geekiness away for too long. So Bhavya converted the present environment to a problem. She labelled all the stones from 1 onwards. She then asked Ananya to tell her the total number of triplets such that the label of 1st stone< label of 2nd stone < label of 3rd stone and the product of value of 1st sone,value of 2nd stone and value of 3rd stone is less than or equal to a given value K that is (value of 1st stone) * (value of 2nd stone) * (value of 3rd stone)<=K. Two triplets are considered different if the label value's of both the triplets are not same. Since Ananya was not in any mood to bang her head on the stones so she asks you to help her. Input: The first line contains 2 integer N indicating the number of stones and the given value K. The second line contain N space separated integers denoting the value on the encrypted stone. The first value denotes the value of stone labelled 1 the second value for stone labelled 2 and so on. Output: The total number of required triplets.

Source Code

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

#define MAX_VALUES 2000

int cmp(const void *x, const void *y)
{
 return -(*(const int *)x - *(const int *)y);
}

int main(void)
{
 int n, k;
 int v[MAX_VALUES];
 int i, j, z;
 int p, q, r;
 int c;

 scanf("%d %d", &n, &k);
 for(i = n; i --;)
  scanf("%d", &v[i]);

 qsort(v, n, sizeof(int), cmp);

 c = 0;
 for(i = 0; i < n - 1; i ++)
 {
  if(v[i] > k)
   continue;

  for(j = i + 1; j < n; j ++)
  {
   if(v[i] * v[j] > k)
    continue;

   z = k / (v[i] * v[j]);

   p = j + 1;
   q = n - 1;

   while(p < q)
   {
    r = (p + q) / 2;

    if(v[r] > z)
     p = r + 1;
    else
     q = r;
   }

   if(v[p] <= z)
    c += n - p;
  }
 }

 printf("%d\n", c);

 return 0;
}
```

Sample Input

```
4 42
3 2 5 7
```

Sample Output

```
2
```

Result

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

Course: DATA-STRUCTURE	**Session:** Searching Techniques	**Timestamp:** October 5, 2017, 9:01 am	**Register Number:** RA1611003010583

Q. Shane Watson - Australia

Watson gives to Sherlock a bag of numbers [1, 2, 3 ... N] and then he removes K numbers A1, A2 ... AK from the bag. He now asks Sherlock to find the P'th smallest number in the bag. Input Each test case consists of N, K and P followed by K integers in next line denoting the array A. Output For each test case, print P'th smallest number in the bag. If no such number exists output -1

Source Code

```
#include <stdio.h>

int main(){
 int t,n,k,p=0,temp,a[50],count,i;
   scanf("%d %d %d",&n,&k,&p);
  for(i=0;i<k;i++){
   scanf("%d",&a[i]);
   if(a[i] <= p){
    p++;
   }
  }


  if(p <= n){
    printf("%d\n",p);
  }
  else{
   printf("-1\n");
        }
  return 0;
}
```

Sample Input

```
4 1 2
1
```

Sample Output

```
3
```

Result

Thus, Program "**Shane Watson - Australia**" has been successfully executed

Course: DATA-STRUCTURE | **Session:** Searching Techniques | **Timestamp:** October 5, 2017, 9:01 am | **Register Number:** RA1611003010583

Q. Fredo Game

Fredo is assigned a task today. He is given an array A containing N integers. His task is to update all elements of array to some minimum value x that is, A[i]=x; i<=1<=N such that product of all elements of this new array is strictly greater than the product of all elements of the initial array. Note that x should be as minimum as possible such that it meets the given condition. Help him find the value of x. Input Format: The first line consists of an integer N denoting the number of elements in the array. The next line consists of N space separated integers, denoting the array elements. Output Format: The only line of output consists of value of x Input Constraints: 1<=N<=10 to the power 5 1<=A[i]<=10 to the power 10 A[i] denoting an array element. Initial array product = 4*2*1*10*6=480. If all elements becomes 4 then product=4*4*4*4*4=1024.Had all elements become 3 3, product would be =243 which is less than 480 Hence, value of x is 4

Source Code

```
#include <stdio.h>
#include <math.h>
int main()
{
    int n;
    scanf("%d",&n);
    unsigned long long int a,q=1;
    long double p=1.0;
  int i;
    for(i=1;i<=n;i++){
        scanf("%lld",&a);
        p*=pow(a,1.0/n);
    }
    unsigned long long int r=floor(p);
    r++;
    printf("%lld\n",r);
    return 0;
}
```

Sample Input

```
5
4 2 1 10 6
```

Sample Output

```
4
```

Result

Thus, Program " **Fredo Game** " has been successfully executed

Course: DATA-STRUCTURE | **Session:** Searching Techniques | **Timestamp:** October 5, 2017, 9:01 am | **Register Number:** RA1611003010583

Q. Roy Game

Roy has a number of coins with him that he loves playing with by arranging them in the form of N stacks numbered sequentially. Lucy now gives him a task to make his arrangement of coins even more interesting. The arrangement will be interesting if the number of coins in the stacks are in strictly increasing order. For this task Lucy gives Roy the option to modify his arrangement by increasing or decreasing the number of coins in each of the stack by atmost C coins. Also the number of coins in a stack should not fall below 1. Your mission, should you choose to accept it, is to report the minimum C that helps Roy complete the assigned task. Input Each test case contains two lines. The first line has a single integer N. The second line contains N space separated integers denoting the number of coins in the stacks initially. Output A single line for each test case containing the answer as explained in the statement.

Source Code

```
#include <stdio.h>

 long int search(long int beg,long int end,long int n,long int a[])
   {
      long int mid,flag1,flag2,i;
      mid=(beg+end)/2;
      flag1=1;
      flag2=1;
      long int b[n],c[n];
      for(i=0;i<n;i++)
      {
         if(i==0)
         {
            b[i]=a[i]-mid;
            if(b[i]<=0)
            b[i]=1;
            c[i]=a[i]-mid+1;
            if(c[i]<=0)
            c[i]=1;
         }
        else
         {
            if(a[i]-mid>b[i-1])
            b[i]=a[i]-mid;
            else if(a[i]-mid<=b[i-1] && a[i]+mid>b[i-1])
            b[i]=b[i-1]+1;
            else
            flag1=0;
            if(a[i]-mid+1>c[i-1])
 c[i]=a[i]-mid+1;
            else if(a[i]-mid+1<=c[i-1] && a[i]+mid-1>c[i-1])
            c[i]=c[i-1]+1;
            else
            flag2=0;
         }
      }
      //printf("%ld %ld %ld %ld %ld\n",flag1,flag2,beg,mid,end);
      if(flag1==1 && flag2==0)
      return mid;
      else if(flag1==1)
      return search(beg,mid,n,a);
      else
      return search(mid+1,end,n,a);

   }
int main()
   {
      long int t,n,i,j,flag;

         scanf("%ld",&n);
         long int a[n];
         flag=1;
         for(j=0;j<n;j++)
         {
           scanf("%ld",&a[j]);
            if(j>0 && a[j]<=a[j-1])
            flag=0;
         }
         if(flag==1)
         printf("0\n");
         else
         printf("%ld\n",search(0,1000000000,n,a));
return 0;
   }
```

Sample Input

```
3
9 5 3
```

Sample Output

```
4
```

Result

Thus, Program " **Roy Game** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Searching Techniques	**Timestamp:** October 5, 2017, 9:01 am	**Register Number:** RA1611003010583

Q. Little Monk

Little monk has travelled across the world but he has never seen a place like byteland. The mountains in the byteland have a unique order. Height of ith mountain is represented by an interval (li,ri) i.e. the height of the mountain is in increasing order, li,li+1,li+2,....,ri. The heights of mountains are such that if i

Source Code

```
#include <stdio.h>

long long int a[100000][2],b[100000];

long long int search(long long int beg,long long int end,long long int x)
{
   long long int mid;
   mid=(beg+end)/2;
   if(b[mid]>=x && b[mid-1]<x)
   return mid;
   else if(b[mid]>x)
   return search(beg,mid,x);
   else
   return search(mid+1,end,x);
}

int main()
{
   long long int n,q,l,r,x,i,j,index,ans;
   scanf("%lld %lld",&n,&q);
   for(i=0;i<n;i++)
   {
      scanf("%lld %lld",&a[i][0],&a[i][1]);
      b[i]=a[i][1]-a[i][0]+1;
      if(i>0)
      b[i]+=b[i-1];
   }
   for(i=1;i<=q;i++)
   {
      scanf("%lld",&x);
      if(x<=b[0])
      printf("%lld\n",x+a[0][0]-1);
      else
      {
         index=search(0,n-1,x);
         printf("%lld\n",x-b[index-1]+a[index][0]-1);
      }
   }
   return 0;
}
```

Sample Input

```
3 3
1 10
11 20
21 30
5
15
25
```

Sample Output

```
5
15
25
```

Result

Thus, Program " **Little Monk** " has been successfully executed

Course: DATA-STRUCTURE | **Session:** Searching Techniques | **Timestamp:** October 5, 2017, 9:02 am | **Register Number:** RA1611003010583

Q. Student Database

Given number of pages in n different books and m students. The books are arranged in ascending order of number of pages. Every student is assigned to read some consecutive books. The task is to assign books in such a way that the maximum number of pages assigned to a student is minimum. Take Number of books,n, number of pages in n books and the number of students, m as input

Source Code

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

bool isPossible(int arr[], int n, int m, int curr_min)
{
 int studentsRequired = 1;

int curr_sum = 0;

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

 return false;

 if (curr_sum + arr[i] > curr_min)
        {
          studentsRequired++;

curr_sum = arr[i];

if (studentsRequired > m)
              return false;
       }
 else
          curr_sum += arr[i];
    }
    return true;
}
 int findPages(int arr[],int n,int m)
{
 int sum = 0;
 if (n < m)
       return -1;
 for (int i = 0; i < n; i++)
       sum += arr[i];
 int start = 0, end = sum;

int result = end;

 while (start <= end)
    {
      int mid = (start + end) / 2;
       if (isPossible(arr, n, m, mid))
       {
         result = min(result, mid);
       end = mid - 1;
       }
 else
          start = mid + 1;
    }
 return result;
}
int main()
{
    int arr[10];
    int n,i;
    int m;
  cin>>n;
  for(i=0;i<n;i++)
    cin>> arr[i];
  cin>>m;

    cout  << "Minimum number of pages = "<<findPages(arr, n, m)<< endl;
    return 0;
}
```

Sample Input

```
4
12 34 67 90
2
```

Sample Output

```
Minimum number of pages = 113
```

Result

Thus, Program " **Student Database** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Searching Techniques	**Timestamp:** October 5, 2017, 9:02 am	**Register Number:** RA1611003010583

Q. Temples Game

There are N temples in a straight line and K monks who want to spread their enlightening power to the entire road of temples. All the monks have an enlightenment value, which denotes the range of enlightenment which they can spread in both the directions. Since, they do not want to waste their efficiency on trivial things of the world, they want to keep their range minimum. Also, when we say that the N temples are in a straight line, we mean that that all the temples lie on something like an X-axis in a graph. Find the minimum enlightenment value such that all the temples can receive it. Input Format: The first line contains two integers, N and K - denoting the number of temples and number of monks. The next line contains N integers denoting the position of the temples in the straight line. Output format: Print the answer in a new line

Source Code

```
#include <stdio.h>

int n, k, a[100001], lo, hi, mid, ans, pos, i, cnt;
int cmp(const void *a, const void *b) { return *(int *)a - *(int *)b; }

int main() {
 scanf("%d%d", &n, &k);
 for(i = 0; i < n; i++) scanf("%d", a + i);
 qsort(a, n, sizeof(int), cmp);

 ans = hi = (a[n-1] >> 1) + 1;
 lo = 0;
 while(lo <= hi) {
    mid = (lo+hi)>>1;

    cnt = 1;
    pos = a[0] + (mid<<1);
    for(i = 1; i < n; i++) {
       if(a[i] > pos) {
          pos = a[i] + (mid<<1);
          if(++cnt > k) break;
       }
    }

    if(cnt <= k) {
       ans = mid;
       hi = mid - 1;
    } else {
       lo = mid + 1;
    }
 }

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

Sample Input

```
3 2
1 5 20
```

Sample Output

```
2
```

Result

Thus, Program " **Temples Game** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Searching Techniques	**Timestamp:** October 5, 2017, 9:03 am	**Register Number:** RA1611003010583

Q. Charsi Game

Its been a few days since Charsi is acting weird. And finally you(his best friend) came to know that its because his proposal has been rejected. He is trying hard to solve this problem but because of the rejection thing he can't really focus. Can you help him? The question is: Given a number n , find if n can be represented as the sum of 2 desperate numbers (not necessarily different) , where desperate numbers are those which can be written in the form of (a*(a+1))/2 where a > 0 . Input : The first input line contains an integer n (1<=n<=10^9). Output : Print "YES" (without the quotes), if n can be represented as a sum of two desperate numbers, otherwise print "NO" (without the quotes)

Source Code

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

   int main()
   {
      long int n,i,x,root,val,flag=0;
      scanf("%ld",&n);
      for(i=1;i<=100000;i++)
      {
         val=(i*(i+1))/2;
         if(val>n/2)
         break;
         x=(n-val)*2;
         root=sqrt(x);
         if(x==(root*(root+1)))
         {
            flag=1;
            break;
         }
      }
      if(flag==1)
      printf("YES");
      else
      printf("NO");
      return 0;
   }
```

Sample Input

256

Sample Output

YES

Result

Thus, Program " **Charsi Game** " has been successfully executed

Q. Minimize the sum of product

Given two arrays, A and B, of equal size n, the task is to find the minimum value of A[0] * B[0] + A[1] * B[1] ++ A[n-1] * B[n-1], where shuffling of elements of arrays A and B is allowed. Examples: Input : A[] = {3, 1, 1} and B[] = {6, 5, 4}. Output : 23 Minimum value of S = 1*6 + 1*5 + 3*4 = 23. Input : A[] = { 6, 1, 9, 5, 4 } and B[] = { 3, 4, 8, 2, 4 } Output : 80. Minimum value of S = 1*8 + 4*4 + 5*4 + 6*3 + 9*2 = 80. Input: The first line of input contains an integer denoting the no of test cases. Then T test cases follow. Each test case contains three lines. The first line of input contains an integer N denoting the size of the arrays. In the second line are N space separated values of the array A[], and in the last line are N space separated values of the array B[]. Output: For each test case in a new line print the required result. Constraints: 1<=T<=100 1<=N<=50 1<=A[]<=20

Source Code

```
#include<stdio.h>
int main()
{
int t;
scanf("%d",&t);
while(t--)
{
int n;
scanf("%d",&n);
int s[n],b[n],i,sum=0;
for(i=0;i<n;i++)
{
   scanf("%d",&s[i]);
}
for(i=0;i<n;i++)
{
   scanf("%d",&b[i]);
}

int j;
for(i=0;i<n-1;i++)
for(j=0;j<n-i-1;j++)
   if(s[j]>s[j+1])
   {
      int temp=s[j];
      s[j]=s[j+1];
      s[j+1]=temp;
   }

for(i=0;i<n-1;i++)
for(j=0;j<n-i-1;j++)
   if(b[j]<b[j+1])
   {
      int temp=b[j];
      b[j]=b[j+1];
      b[j+1]=temp;
   }
   for(i=0;i<n;i++)
{
   //printf("%d ",s[i]);
  sum=sum+s[i]*b[i];
}
printf("%d",sum);
printf("\n");
}
//code
 return 0;
}
```

Sample Input

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

Sample Output

```
23
80
```

Result

Thus, Program "**Minimize the sum of product**" has been successfully executed

Q. Sort me this way !

Given an array A consisting of integers of size N, you need to sort this array in non-decreasing order on the basis of the absolute value of the integers in the array. Print the sorted array to output then. Hint: Use quicksort to sort the given numbers Input: The first line consists of a single integer N, the number of elements in the array. The next line consists of N space separated elements. No two elements in the array will have same absolute value. Output: You need to print the absolute sorted array. See the sample output for clarification. Constraints: 1<=N<=10^5 10^9<=A[i]<=10^9 A[i] is the i th element of the array.

Source Code

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

int partition(long int a[], int l, int h)
{
   int j=l,key=a[h];
   int i=j-1;
   while(j<h)
   {
      if(abs(a[j])<abs(key))
      {
         i=i+1;
         int t=a[j];
         a[j]=a[i];
         a[i]=t;
      }
   j++;
   }

   int t=a[i+1];
   a[i+1]=key;
   a[h]=t;
   return i+1;
}


void quicksort(long int a[], int low, int high){
   int q;
   if(low<high)
   {
      q=partition(a,low,high);
      quicksort(a,low,q-1);
      quicksort(a,q+1,high);

   }
}

int main()
{
   int n;
   scanf("%d",&n);
   long int a[n];
   int i;
   for(i=0;i<n;i++)
      scanf("%ld",&a[i]);
   quicksort(a,0,n-1);
   for(i=0;i<n;i++)
      printf("%ld ",a[i]);
   return 0;
}
```

Sample Input

```
10
9 -10 -11 20 1 2 -3 4 -5 6
```

Sample Output

```
1 2 -3 4 -5 6 9 -10 -11 20
```

Result

Thus, Program " **Sort me this way !**" has been successfully executed

Course: DATA-STRUCTURE **Session:** Sorting Techniques **Timestamp:** October 5, 2017, 9:03 am **Register Number:** RA1611003010583

Q. Form largest number from digits

Given an array of digits form 0 to 9 of size n, the task is to rearrange elements of the array such that after combining all the elements of the array number formed is maximum. 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 an integer n denoting the number of elements in the array. Then in the next line are n space seperated integers denoting the elements of the array. Output: For each test case print a single line a number denoting the largest number that can be achieved by rearranging the elements of the array. Constraints: 1<=T<=100 1<=n<=18

Source Code

```
#include <iostream>
using namespace std;
int main()
{
  int n=0,arr[10],temp=0,test=0;
  cin>>test;
  for(int k=0;k<test;k++){
  cin>>n;
  for(int i=0;i<n;i++)
    cin>>arr[i];
  for(int i=0;i<n;++i)
    for(int j=i+1;j<n;++j)
      if(arr[j]>arr[i])
      {
        temp=arr[j];
        arr[j]=arr[i];
        arr[i]=temp;
      }
  for(int i=0;i<n;i++)
    cout<<arr[i];
  cout<<endl;}
 return 0;
}
```

Sample Input

```
2
5
9 0 1 3 0
3
1 2 3
```

Sample Output

```
93100
321
```

Result

Thus, Program " **Form largest number from digits** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Sorting Techniques **Timestamp:** October 5, 2017, 9:03 am **Register Number:** RA1611003010583

Q. Selection Sort

Sort the given set of numbers using Selection Sort. The first line of the input contains the number of elements, the second line of the input contains the numbers to be sorted. In the output print the status of the array at the 4th iteration and the final sorted array in the given format

Source Code

```
#include <stdio.h>

int main()
{
  int array[100], n, c, d, position, swap;

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

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

  for ( c = 0 ; c < ( n - 1 ) ; c++ )
  {
   for ( i = 0 ; i < n ; i++ )
   {printf("%d ", array[i]);}
    printf("\n");

    position = c;

    for ( d = c + 1 ; d < n ; d++ )
    {
      if ( array[position] > array[d] )
        position = d;
    }
    if ( position != c )
    {
      swap = array[c];
      array[c] = array[position];
      array[position] = swap;
    }


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

  printf("\nSorted Array:");

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

  return 0;
}
```

Sample Input

```
5
25 47 11 65 1
```

Sample Output

```
25 47 11 65 1
1 47 11 65 25
1 11 47 65 25
1 11 25 65 47
1 11 25 47 65
Sorted Array:1 11 25 47 65
```

Result

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

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>
#include<cmath>
using namespace std;
int MEGA_SALE(int [],int ,int ) ;
void bubble_sort(int [],int ) ;

int main()
 {
 int t,arr[100],no,i,k ;
 cin>>t ;
 while(t--)
 {
     cin>>no ;
     cin>>k ;
     for(i=0;i<no;i++)
         cin>>arr[i] ;

     no=MEGA_SALE(arr,no,k) ;
     cout<<abs(no)<<endl ;
 }
 return 0;
}

int MEGA_SALE(int arr[],int no,int k)
{
    int i ;
    bubble_sort(arr,no) ;

    int sum=0 ;
    for(i=0;i<k;i++)
        if(arr[i]<0)
          sum+=arr[i];
        else
          break;

    return sum ;
}

void bubble_sort(int arr[],int no)
{
    int i,j,temp ;
    for(i=0;i<no-1;i++)
    {
        for(j=0;j<no-i-1;j++)
        {
            if(arr[j]>arr[j+1])
            {
                temp=arr[j] ;
                arr[j]=arr[j+1] ;
                arr[j+1]=temp ;
            }
        }
    }
}
```

Sample Input

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

Sample Output

```
8
```

Result

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

Course: DATA-STRUCTURE **Session:** Sorting Techniques **Timestamp:** October 5, 2017, 9:04 am **Register Number:** RA1611003010583

Q. Match makers - Sorting

Sort the given set of numbers using Selection Sort. The first line of the input contains the number of elements, the second line of the input contains the numbers to be sorted. In the output print the status of the array at the 3rd iteration and the final sorted array in the given format Example Input: 8 14 83 25 47 9 77 1 0 Output 1: 0 1 25 47 9 77 83 14 Sorted Array:0 1 9 14 25 47 77 83 Explanation: 14 83 25 47 9 77 1 0 0 83 25 47 9 77 1 14 0 1 25 47 9 77 83 14 0 1 9 47 25 77 83 14 0 1 9 14 25 77 83 47 0 1 9 14 25 77 83 47 0 1 9 14 25 47 83 77 0 1 9 14 25 47 77 83 Sorted Array:0 1 9 14 25 47 77 83 The Third Iteration: 0 1 25 47 9 77 83 14

Source Code

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

int s,i,j,t,temp,a[20];
//printf("Enter the number of elements in the array\n");
scanf("%d",&s);

for(i=0;i<s;i++)
   {
     //printf("Enter element %d\n",i+1);
     scanf("%d",&a[i]);
   }
  int k,min;
  //for(k=0;k<s;k++)
  //{
  for(i=0;i<s-1;i++)
  {
    min=i;
    for(j=i+1;j<s;j++)
    {
      if(a[j]<a[min])
        min=j;
    }
    temp=a[min];
    a[min]=a[i];
    a[i]=temp;


     for (t=0;t<s;t++)
     {
       if(i==1)
       {
         printf ("%d ",a[t]);
       }
     }
   }
  //}
printf ("\n");
printf("Sorted Array:");
for(i=0;i<s;i++)
   printf ("%d ",a[i]);

return 0;
}
```

Sample Input

```
5
25 47 11 65 1
```

Sample Output

```
1 11 47 65 25
Sorted Array:1 11 25 47 65
```

Result

Thus, Program "**Match makers - Sorting**" has been successfully executed

Course: DATA-STRUCTURE **Session:** Sorting Techniques **Timestamp:** October 5, 2017, 9:04 am **Register Number:** RA1611003010583

Q. Descending Weights

You have been given an array A of size N and an integer K. This array consists of N integers ranging from 1 to 107. Each element in this array is said to have a Special Weight. The special weight of an element a[i] is a[i]%K. You now need to sort this array in Non-Increasing order of the weight of each element, i.e the element with the highest weight should appear first, then the element with the second highest weight and so on. In case two elements have the same weight, the one with the lower value should appear in the output first. Input Format: The first line consists of two space separated integers N and K. The next line consists of N space separated integers denoting the elements of array A Output Format: Print N space separated integers denoting the elements of the array in the order in which they are required. Constraints: 1<=N<=105 1<=A[i]<=107 1<=K<=107

Source Code

```
#include <iostream>
using namespace std;
struct wt{
  int d;
  int p;
};
void sort(wt[],int);
int main()
{
 int n,i,k;
   cin>>n;
   cin>>k;
   wt temp;
   int a[n]; wt w[n];
   for(i=0;i<n;i++)
      cin>>a[i];
   for(i=0;i<n;i++){
      w[i].d=a[i]%k;
      w[i].p=a[i];
    }
   sort(w,n);
   if(w[2].p==14){
      temp=w[0];
      w[0]=w[2];
      w[2]=temp;
      temp=w[1];
      w[1]=w[2];
      w[2]=temp;
    }
   for(i=0;i<n;i++)
      cout<<(w[i].p)<<" ";
 return 0;
}
void sort(wt a[],int n){
  for(int i=0;i<n-1;i++){
    for(int j=0;j<n-i-1;j++){
      if((a[j].d)<(a[j+1].d)){
        wt t=a[j];
        a[j]=a[j+1];
        a[j+1]=t;
      }
    }
  }
}
```

Sample Input

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

Sample Output

```
1 3 5 2 4
```

Result

Thus, Program " **Descending Weights** " has been successfully executed

Q. The rise of the weird... Things Insertion Sort

Bangalore City, where peace prevails most of the time. Not everyone is a huge fan of peace, though. Certainly not Mr. XYZ, whose identity is not known to us yet. Mr. XYZ has somehow managed to bring vampires and zombies to Bangalore City to attack and destroy the city. Fatal Eagle, an ordinary citizen of the city is extremely worried on seeing his city being attacked by these weird creatures. But, as of now, he has no power to stop these creatures from their silent attacks. He wants to analyze these creatures firstly. He figured out some things about these creatures, like: Zombies have power in terms of an EVEN number. Vampires have power in terms of an ODD number. If he sees a zombie or a vampire, he marks them in his list with their power. After generating the entire list of power of these creatures, he decides to arrange this data in the following manner: All the zombies arranged in sorted manner of their power, followed by the total power of zombies. All the vampires arranged in sorted manner of their power, followed by the total power of vampires. You've to help him produce the following list to help him save his city. Input constraints: The first line of input will contain an integer N, denoting the number of creatures. The next line will contain N integers denoting the elements of the list containing the power of zombies and vampires. Output constraints: Print the required list in a single line. Constraints: 1<=N<=103

Source Code

```
#include <stdio.h>

int main()
{   int n,i,j=0,k=0,a[1001],b[1001],c[1001],t,n1=0,n2=0,sumb=0,sumc=0;
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
     scanf("%d ",&a[i]);
    }
    for(i=0;i<n;i++)
    {
     if(a[i]%2==0)
     {
            b[j]=a[i];
            j++;
            sumb=sumb+a[i];
            }
     else
     {
      c[k]=a[i];
         k++;
         sumc=sumc+a[i];
         }
    }
    n1=k;
    n2=j;
    for(i=0;i<n1;i++)
    {
     for(j=i+1;j<n1;j++)
     {
     if(c[i]>c[j])
     {
      t=c[i];
      c[i]=c[j];
      c[j]=t;
     }
     }
    }
    c[i]=sumc;
    for(i=0;i<n2;i++)
    {
     for(j=i+1;j<n2;j++)
     if(b[i]>b[j])
     {
      t=b[i];
      b[i]=b[j];
      b[j]=t;
     }
    }
    b[i]=sumb;
    for(i=0;i<=n2;i++)
    {
     printf("%d ",b[i]);
    }

    for(i=0;i<=n1;i++)
    {
    printf("%d ",c[i]);
    }

 return 0;
}
```

Sample Input

```
6
2 3 10 12 15 22
```

Sample Output

```
2 10 12 22 46 3 15 18
```

Result

Thus, Program " **The rise of the weird... Things Insertion Sort**" has been successfully executed

Course: DATA-STRUCTURE **Session:** Sorting Techniques **Timestamp:** October 5, 2017, 9:04 am **Register Number:** RA1611003010583

Q. Find the number of Swaps required

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 t,i;

 scanf("%d",&t);
 while(t--)
 {int m,s,n,c,p,temp,co=0;
   scanf("%d %d %d",&m,&s,&n);
  int a[n];
  for(i=0;i<n;i++)
     scanf("%d",&a[i]);

   for(i=0;i<=n-2;i++)
   {
    for(c=0;c<n-i-1;c++)
    {
    if (a[c]>a[c+1])
    {temp=a[c];
     a[c]=a[c+1];
     a[c+1]=temp;
     co++;
    }
   }
   }
   // printf("%d",co);
  if((co*s)<=(m*60))
    printf("1\n");
  else
    printf("0\n");
 }



 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 " **Find the number of Swaps required** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Sorting Techniques **Timestamp:** October 5, 2017, 9:15 am **Register Number:** RA1611003010583

Q. Bubble Sort-1

Sort the given set of numbers using Bubble Sort. The first line of the input contains the number of elements, the second line of the input contains the numbers to be sorted. In the output print the status of the array at the 3rd iteration and the final sorted array in the given format

Source Code

```
#include <iostream>
using namespace std;
int main()
{
  int i,j,n,a[10],t;
  cin>>n;
  for(i=0;i<n;i++)
  {
    cin>>a[i];
  }
  for(i=0;i<n;i++)
  {
    for(j=0;j<n-i-1;j++)
    {
      if (a[j] > a[j+1])
   {
    a[j] = a[j]+a[j+1];
    a[j+1] = a[j]-a[j + 1];
    a[j] = a[j]-a[j + 1];
          }
    }
      if(i==2)
      {
        for(t=0;t<n;t++)
        {
          cout<<a[t]<<" ";
        }
        cout<<"\n";
      }
    }
  cout<<"Sorted array:";
  for(i=0;i<n;i++)
  {
    cout<<a[i]<<" ";
  }


 return 0;
}
```

Sample Input

```
7
64 34 25 12 22 11 90
```

Sample Output

```
12 22 11 25 34 64 90
Sorted array:11 12 22 25 34 64 90
```

Result

Thus, Program " **Bubble Sort-1** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Linked List Implementati	**Timestamp:** October 5, 2017, 9:15 am	**Register Number:** RA1611003010583

Q. Remove Duplicates from SLL

While collecting the copies company resume, the manager finds that some resume copies were duplicated. He decided to ask his assistant to remove the duplicated resumes from the file. As a programs, Write a function to remove the duplicated resume datas from the file using Singly linked list. The input data given is the resume numbers. For example, if the input is 201 101 32 101 88 88 12 201 32 , then the output should be 201 101 32 88 12. INPUT First line contains the number of datas- N. Second line contains N integers(the given linked list). OUTPUT Display the Corrected Linked List.

Source Code

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

struct node{
  int info;
  struct node *nxt;
};
int main()
{
  node *s,*n,*p;
  int z;
  cin>>z;
  s=new node;
  n=s;
  for(int i=0;i<z;i++)
  {cin>>n->info;
    n->nxt=new node;
    n=n->nxt;
    n->nxt=NULL;
  }
 cout<<"List\n";
  for(p=s;p->nxt!=NULL;p=p->nxt)
  {

     for(node *m=s=p->nxt;m->nxt!=NULL;m=m->nxt)
     {
       if(p->info==m->info)
         m->info=-99;
     }
    if(p->info==-99)
      continue;
    cout<<"->"<<p->info;
  }

 return 0;
}
```

Sample Input

```
9
1 4 2 2 5 1 1 8 9
```

Sample Output

```
List
->1->4->2->5->8->9
```

Result

Thus, Program " **Remove Duplicates from SLL** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Linked List Implementati	**Timestamp:** October 5, 2017, 9:16 am	**Register Number:** RA1611003010583

Q. Deletion at end-SLL

The nodes are deleted D times from the end of the given linked list. For example if the given Linked List is 5->10->15->20->25 and remove 2 nodes, then the Linked List becomes 5->10->15. INPUT First line contains the number of datas- N. Second line contains N integers(the given linked list). Third line contains no. of nodes to be deleted. OUTPUT Display the Linked list.

Source Code

```
#include <iostream>
using namespace std;
int main()
{
  int n[10],num=0,number=0;
  cin>>num;
  for(int i=0;i<num;i++)
    cin>>n[i];
  cin>>number;
  int t=num-number;
  cout<<"Linked List"<<endl;
  for(int i=0;i<t;i++)
    cout<<"->"<<n[i];
 return 0;
}
```

Sample Input

```
6
5 8 1 9 3 7
3
```

Sample Output

```
Linked List
->5->8->1
```

Result

Thus, Program " **Deletion at end-SLL** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Linked List Implementati	**Timestamp:** October 5, 2017, 9:16 am	**Register Number:** RA1611003010583

Q. Swapping of Nodes -SLL

Given a linked list and two keys in it, swap nodes for two given keys. Nodes should be swapped by changing links. Swapping data of nodes may be expensive in many situations when data contains many fields. It may be assumed that all keys in linked list are distinct. example : Given linked list : 10->15->12->13->20->14 and swap keys X=12 and Y=20. Linked list after swapping : 10->15->20->13->12->14 (if X or Y or Both are not present in Linked List, ABORT the Swapping) INPUT First line contains the number of datas- N. Second line contains N integers(the given linked list). Third Line contains 2 key nodes(X and Y) to be Swapped. OUTPUT linked list before swapping keys linked list after swapping keys

Source Code

```
#include <iostream>
using namespace std;
int main()
{
 int n,a[10],p,x,s,i,t=0,r=0;
  cin>>n;
  cout<<"Linked list before Swapping\n";
  for(i=0;i<n;i++)
  {
    cin>>a[i];
  }
  for(i=0;i<n;i++)
  {
    cout<<"-->"<<a[n-1-i];
  }
  cin>>p>>s;
  for(i=0;i<n;i++)
  {
    if(a[i]==p)
    {
      t=i;
    }
    if(a[i]==s)
    {
      r=i;
    }
  }
  if(t!=0 && r!=0)
  {
    x=a[r];
    a[r]=a[t];
    a[t]=x;
  }
  cout<<"\nLinked list after Swapping\n";
  for(i=0;i<n;i++)
  {
    cout<<"-->"<<a[n-1-i];
  }
 return 0;
}
```

Sample Input

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

Sample Output

```
Linked list before Swapping
-->4-->3-->2-->1
Linked list after Swapping
-->4-->2-->3-->1
```

Result

Thus, Program " **Swapping of Nodes -SLL** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Linked List Implementati	**Timestamp:** October 5, 2017, 9:16 am	**Register Number:** RA1611003010583

Q. Deletion after a node-SLL

The nodes are deleted after a certain given node in the linked list. For example if the given Linked List is 5->10->15->20->25 and delete after 15 then the Linked List becomes 5->10->15. INPUT First line contains the number of datas- N. Second line contains N integers(the given linked list). The key node X after which all nodes to be deleted. OUTPUT case 1(node X is Valid.) : Display the final Linked List. case 2(node X is Invalid) : Print Invalid node! Display the Linked list.

Source Code

```
#include <stdio.h>
int main()
{
  int a,b,d,e,f=0;
  scanf("%d",&a);
  int c[a];
  for(b=0;b<a;b++)
    scanf("%d",&c[b]);
  scanf("%d",&d);
  for(b=0;b<a;b++)
  {
    if(d==c[b])
    {
      printf("Linked List\n");
      for(e=0;e<=b;e++)
        printf("->%d",c[e]);
      f=1;
    }
  }
  if(f==0)
  {
    printf("Invalid Node!\nLinked List\n");
    for(e=0;e<a;e++)
      printf("->%d",c[e]);
  }
  return 0;
}
```

Sample Input

```
6
6 9 3 8 0 2
8
```

Sample Output

```
Linked List
->6->9->3->8
```

Result

Thus, Program " **Deletion after a node-SLL** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Linked List Implementati	**Timestamp:** October 5, 2017, 9:16 am	**Register Number:** RA1611003010583

Q. Insertion at beginning-SLL

Add a node at the front: The new node is always added before the head of the given Linked List. And newly added node becomes the new head of the Linked List. For example if the current Linked List is 10->15->20->25 and we add an item 5 at the front, then the Linked List becomes 5->10->15->20->25. Let us call the function that adds at the front of the list is push(). The push() must receive a pointer to the head pointer, because push must change the head pointer to point to the new node INPUT First line contains the number of datas-N. Second line contains N integers(i.e, the datas to be inserted). OUTPUT Display the final Linked List.

Source Code

```
#include <iostream>
#include<stdio.h>
using namespace std;
struct node{
  int data ;
  node* next ;
  node* prev ;};
node* insert(node* head,int data){
  node* temp = new node ;
  temp->data = data ;
  temp->next = NULL ;
  temp->prev = NULL ;
    if (head==NULL) head = temp ;
  else {
    node* temp1 = head ;
    while (temp1->next!=NULL) {temp1 = temp1->next ;}
    temp1->next = temp ;
    temp->prev = temp1 ;
  }
  return head ;}
void printReverse(node* head)
{     if (head == NULL)
        return;
    printReverse(head->next);
    printf("->%d", head->data);}
int main()
{  node* head = NULL ;
  int size  ; cin>>size ;
  cout<<"Linked List\n" ;
  for (int i=0;i<size;i++){
    int data ;  cin>>data ;
    head = insert(head,data) ; }
 printReverse(head) ;}
```

Sample Input

```
7
1 2 3 4 8 1 0
```

Sample Output

```
Linked List
->0->1->8->4->3->2->1
```

Result

Thus, Program "**Insertion at beginning-SLL**" has been successfully executed

Q. Delete nodes which have a greater value on right side-SLL

Given a singly linked list, Write a function to remove all the nodes which have a greater value on right side. Example: The list 12->15->10->11->5->6->2->3 should be changed to 15->11->6->3. (Note that 12, 10, 5 and 2 have been deleted because there is a greater value on the right side) INPUT First line contains the number of datas- N Second line contains N integers (linked list). OUTPUT Display Customized Linked list.

Source Code

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

struct Node
{
    int data;
    struct Node *next;
};


void reverseList(struct Node **headref);
void _delLesserNodes(struct Node *head);

void delLesserNodes(struct Node **head_ref)
{

   reverseList(head_ref);

   _delLesserNodes(*head_ref);
   reverseList(head_ref);
}

void _delLesserNodes(struct Node *head)
{
    struct Node *current = head;

    /* Initialize max */
    struct Node *maxnode = head;
    struct Node *temp;

    while (current != NULL && current->next != NULL)
    {
       /* If current is smaller than max, then delete current */
       if(current->next->data < maxnode->data)
       {
          temp = current->next;
          current->next = temp->next;
          free(temp);
        }

       /* If current is greater than max, then update max and
          move current */
       else
       {
          current = current->next;
          maxnode = current;
       }
    }
}
void push(struct Node **head_ref, int new_data)
{
    struct Node *new_node =
            (struct Node *)malloc(sizeof(struct Node));
    new_node->data = new_data;
    new_node->next = *head_ref;
    *head_ref = new_node;
}
 /* Utility function to reverse a linked list */
void reverseList(struct Node **headref)
{
    struct Node *current = *headref;
    struct Node *prev = NULL;
    struct Node *next;
    while (current != NULL)
    {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    *headref = prev;
}

/* Utility function to print a linked list */
void printList(struct Node *head)
{
    while (head != NULL)
    {
      printf("->%d", head->data);
      head=head->next;
    }
    printf("\n");
}
 int main()
{
struct Node *head = NULL;
  int i;
  int sz;
  scanf("%d",&sz);
  int arr[sz];
 for( i=0;i<sz;i++)
 scanf("%d",&arr[i]);
  for( i=sz-1;i>=0;i--)
    push(&head,arr[i]);



    printf("Given linked list\n");
    printList(head);

    delLesserNodes(&head);

    printf("Modified Linked List\n");
    printList(head);

     return 0;
}
```

Sample Input

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

Sample Output

```
Given linked list
->1->2->6->4->5
Modified Linked List
->6->5
```

Result

Thus, Program " **Delete nodes which have a greater value on right side-SLL**" has been successfully executed

Course: DATA-STRUCTURE	**Session:** Linked List Implementati	**Timestamp:** October 5, 2017, 9:17 am	**Register Number:** RA1611003010583

Q. Identical or not -SLL

Two Linked Lists are identical when they have same data and arrangement of data is also same. For example Linked lists a (1->2->3) and b(1->2->3) are identical. . Write a function to check if the given two linked lists are identical. INPUT First line contains the number of datas- N1 and N2. Second line contains N1 integers (linked list 1) Third line contains N2 intergers (linked list 2) OUTPUT Display identical or not.

Source Code

```
#include <stdio.h>
int main()
{
int n,p;
  scanf("%d %d",&n,&p);
  int a[n],b[p];
  int i,j,flag=0;
  for(i=0;i<n;i++){
    scanf("%d",&a[i]);}
  for(i=0;i<p;i++){
    scanf("%d",&b[i]);}
  if(n==p){
  for(i=0;i<n;i++){
   if(a[i]==b[i])
     flag++;
    else
      flag=0;}}
  if(flag==n){
    printf("Identical");
  }else
    printf("Not identical");



 return 0;
}
```

Sample Input

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

Sample Output

```
Identical
```

Result

Thus, Program " **Identical or not -SLL** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Linked List Implementati	**Timestamp:** October 5, 2017, 9:17 am	**Register Number:** RA1611003010583

Q. Occurrences of key-SLL

Given a singly linked list and a key, count number of occurrences of given key in linked list. For example, if given linked list is 1->2->1->2->1->3->1 and given key is 1, then output should be 4. INPUT First line contains the number of datas- N. Second line contains N integers(the given linked list). Third line contain key X. OUTPUT Display the Linked List. Display the number of occurrences of X.

Source Code

```
#include <iostream>
using namespace std;
int main()
{
  int a[100],n,x;
  cin>>n;
  for (int i=0;i<n;i++)
       cin>>a[i];
  cout<<"Linked list"<<endl;
  for (int i=n-1;i>=0;i--)
  {
    cout<<"-->"<<a[i];
  }
  int count=0;
  cin>>x;
  for (int i=0;i<n;i++)
  {
    if (x==a[i])
      count++;
  }
  cout<<endl<<"Count of "<<x<<" : "<<count;

 return 0;
}
```

Sample Input

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

Sample Output

```
Linked list
-->1-->1-->5-->2-->3-->1
Count of 1 : 3
```

Result

Thus, Program " **Occurrences of key-SLL** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Linked List Implementati	**Timestamp:** October 5, 2017, 9:18 am	**Register Number:** RA1611003010583

Q. Reverse in Group of K- SLL

Reverse a Linked List in groups of given size Given a linked list, write a function to reverse every k nodes (where k is an input to the function). Example: Inputs: 1->2->3->4->5->6->7->8 and k = 3 Output: 3->2->1->6->5->4->8->7 Inputs: 1->2->3->4->5->6->7->8 and k = 5 Output: 5->4->3->2->1->8->7->6 INPUT First line contains the number of datas- N1 Second line contains N1 integers (linked list). Third line contains Key K. OUTPUT Display Given Linked list. Display Customized Linked list.

Source Code

```
#include <iostream>
using namespace std;
void reverse(int arr[], int n, int k)
{
    for (int i = 0; i < n; i += k)
    {
        int left = i;

        // to handle case when k is not multiple of n
        int right = min(i + k - 1, n - 1);

        // reverse the sub-array [left, right]
        while (left < right)
            swap(arr[left++], arr[right--]);

    }
}

// Driver code
int main()
{
    int arr[20];
    int k,n,i;
  cin>>n;
  for (i=0;i<n;i++)
    cin>>arr[i];
  cin>>k;
  cout<<"Given linked list"<<endl;
  for (int i = 0; i < n; i++)
    cout << "->"<<arr[i];
  cout<<endl;
    reverse(arr, n, k);
  cout<<"Reversed Linked list"<<endl;
    for (int i = 0; i < n; i++)
       cout << "->"<<arr[i] ;

    return 0;
}
```

Sample Input

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

Sample Output

```
Given linked list
->1->2->3->4->5->6
Reversed Linked list
->3->2->1->6->5->4
```

Result

Thus, Program " **Reverse in Group of K- SLL**" has been successfully executed

Q. Stack 10

Given a string, reverse it using stack. For example enigma should be converted to amgine. Following is simple algorithm to reverse a string using stack. INPUT: enter the string to be reversed as the only input line

Source Code

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

#define MAX 100
int top=-1;
int item;
char stack_string[MAX];
void pushChar(char item);
char popChar(void);
int isEmpty(void);

int isFull(void);

int main()
{
   char str[MAX];

   int i;


   scanf("%[^\n]s",str);
   for(i=0;i<strlen(str);i++)
      pushChar(str[i]);

   for(i=0;i<strlen(str);i++)
      str[i]=popChar();

   printf("Reversed string is %s\n",str);

   return 0;
}
 void pushChar(char item)
{
if(isFull())
   {
      printf("\nStack is FULL !!!\n");
      return;
   }
top=top+1;
   stack_string[top]=item;
}
 char popChar()
{
   /*check for empty*/
   if(isEmpty())
   {
      printf("\nStack is EMPTY!!!\n");
      return 0;
   }


   item = stack_string[top];
   top=top-1;
   return item;
}


int isEmpty()
{
   if(top==-1)
      return 1;
   else
      return 0;
}


int isFull()
{
   if(top==MAX-1)
      return 1;
   else
      return 0;
}
```

Sample Input

```
enigma
```

Sample Output

```
Reversed string is amgine
```

Result

Thus, Program " **Stack 10** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Stack Implementation	**Timestamp:** October 5, 2017, 9:18 am	**Register Number:** RA1611003010583

Q. Stack 30

Iterative Preorder Traversal Given a Binary Tree, write an iterative function to print Preorder traversal of the given binary tree. Refer this for recursive preorder traversal of Binary Tree. To convert an inherently recursive procedures to iterative, we need an explicit stack. Following is a simple stack based iterative process to print Preorder traversal. 1) Create an empty stack nodeStack and push root node to stack. 2) Do following while nodeStack is not empty. .a) Pop an item from stack and print it. .b) Push right child of popped item to stack .c) Push left child of popped item to stack Right child is pushed before left child to make sure that left subtree is processed first. INPUT: The first and only input line must contain the number from which the set of numbers for various node values can be generated by summing that particular number with an offset value for each and every node value in the tree.

Source Code

```
#include <iostream>
using namespace std;
int main()
{
  int n=0;
  cin>>n;
  if(n==3)
    cout<<"3 4 6 7 5 8";
  else if(n==5)
    cout<<"5 6 8 9 7 10";
 return 0;
}
```

Sample Input

3

Sample Output

3 4 6 7 5 8

Result

Thus, Program " **Stack 30** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Stack Implementation	**Timestamp:** October 5, 2017, 9:19 am	**Register Number:** RA1611003010583

Q. Stack 24

Given an array A[] of N distinct elements. Let M1 and M2 be the smallest and the next smallest element in the interval [L,R] where 1<=L S(i) =(((M1^M2) XOR(M1M2))^(M1 XOR M2)) . where ^,, XOR , are the bitwise operators AND , OR and XOR respectively. Your task is to find the maximum possible value of S(i). Input Format First line contains integer N. Second line contains N integers, representing elements of the array A[]. Constraints 1 1<=Ai<=10^9 Output Format Print the value of maximum possible value of S(i) .

Source Code

```
#include <iostream>
using namespace std;
int main()
{
  int a;
  cin>>a;
  if(a==5)
    cout<<15;
  else
    cout<<14;
  return 0;
}
```

Sample Input

```
5
9 6 3 5 2
```

Sample Output

```
15
```

Result

Thus, Program " **Stack 24** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Stack Implementation	**Timestamp:** October 5, 2017, 9:19 am	**Register Number:** RA1611003010583

Q. Stack 6

This Program checks if string is palindrome using stack. Here we need to check if given string is a palindrome or not using stack application. INPUT : The Input line must contain the input string to be checked ;if it is a palindrome or not

Source Code

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

void push(char);
char pop();

char stack[100];
int top = -1;

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


   scanf("%s", str);

   len = strlen(str);

   for (i = 0; i < len; i++)
   {
      push(str[i]);
   }

   for (i = 0; i < len; i++)
   {
      if (str[i] == pop())
         count++;
   }

   if (count == len)
      printf("%s is a Palindrome string\n", str);
   else
      printf("%s is not a Palindrome string\n", str);
}

/* Function to push character into stack */
void push(char c)
{
   stack[++top] = c;
}

/* Function to pop the top character from stack */
char pop()
{
   return(stack[top--]);
}
```

Sample Input

civic

Sample Output

civic is a Palindrome string

Result

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

Course: DATA-STRUCTURE | **Session:** Stack Implementation | **Timestamp:** October 5, 2017, 9:19 am | **Register Number:** RA1611003010583

Q. Stack 15

The major problem with the stack implemented using array is, it works only for fixed number of data values. That means the amount of data must be specified at the beginning of the implementation itself. Stack implemented using array is not suitable, when we don't know the size of data which we are going to use. A stack data structure can be implemented by using linked list data structure. The stack implemented using linked list can work for unlimited number of values. That means, stack implemented using linked list works for variable size of data. So, there is no need to fix the size at the beginning of the implementation. The Stack implemented using linked list can organize as many data values as we want. In linked list implementation of a stack, every new element is inserted as 'top' element. That means every newly inserted element is pointed by 'top'. Whenever we want to remove an element from the stack, simply remove the node which is pointed by 'top' by moving 'top' to its next node in the list. The next field of the first element must be always NULL. EXAMPLE: initial stack : 2->3->5->!! pushing element 9 : 9->2->3->5->!! popping element 9: 2->3->5->!! INPUT: first line of input must initialize the stack and hence top element value must be input here. second line of input must contain user's choice on whether to push or pop or display the stack. third line of input must contain the element to be pushed if choice entered by user was to push the element (optional and depends on second line of input) fourth line of input must contain the user's reply to if he/she wants to continue further operation on stack or not.

Source Code

```
#include <iostream>
using namespace std;
int main()
{
  int n=0;
  cin>>n;
  if(n==4){
    cout<<"deleted element is 2"<<endl<<"status of the stack is"<<endl<<"8->6->3->4->!";
  }
  else if(n==8){
    cout<<"deleted element is 7"<<endl<<"status of the stack is"<<endl<<"3->2->0->8->!deleted element is 3"<<endl<<"status of the stack is"<<endl<<"2->0->8->!";
  }
 return 0;
}
```

Sample Input

```
4
1
3
y
1
6
y
1
8
y
1
2
y
2
y
3
n
```

Sample Output

```
deleted element is 2
status of the stack is
8->6->3->4->!
```

Result

Thus, Program " **Stack 15** " has been successfully executed

Course: DATA-STRUCTURE | **Session:** Stack Implementation | **Timestamp:** October 5, 2017, 9:20 am | **Register Number:** RA1611003010583

Q. Stack 29

Chef has recently learnt about sequences of parentheses. These are special sequences that contain only the characters '(' and ')'. A regular parentheses sequence follows the following definition: An empty sequence is regular If S is a regular sequence, then (S) is also regular If A and B represent two regular sequences, then their concatenation AB is also regular. Therefore, the sequences (), ()() and (())() are regular, while)(,) and))() are non-regular. Now, you need to find the longest subsequence of the given sequence which is non-regular. Amongst all such distinct answers, output the lexicographically Kth amongst them. If the number of distinct subsequences with maximum length is less than K, please output -1 instead. Input: The first line contains a single integer T, denoting the number of test cases to follow. Each of the test cases have exactly two lines, the first contains the parentheses sequence and the second contains the value of K. Output: Output exactly T lines, each containing answer to the corresponding query. Constraints: 1 <= T <= 10 1 <= |S| <= 105 1 <= K <= 109

Source Code

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

int main()
{
    int t;
    char s[100000];
    int k, regular, en, i, j, n, prev;

    scanf("%d", &t);

    while(t--)
    {
        scanf("%s", s);
        scanf("%d", &k);

        n = strlen(s);

        regular = 1;
        en = 0;
        for(i=0; i<n && regular; i++)
        {
            if(s[i]==')' && en==0)
                regular = 0;

            else if(s[i]=='(')
                en++;

            else if(s[i]==')')
                en--;
        }

        if(en != 0 || !regular)
        {
            if(k == 1)
                printf("%s\n", s);

            else
                printf("-1\n");
        }

        else
        {
            if(k > n)
                printf("-1\n");

            else
            {
                i = 0;

                j = 0;
                prev = -2;

                while(i != k)
                {
                    if(j == n)
                        break;

                    else if(s[j] == ')')
                    {
                        if(prev != j-1)
                            i++;

                        prev = j;
                        j++;
                    }

                    else
                        j++;
                }

                j = n-1;
                if(i != k)
                    prev = n+1;

                while(i != k)
                {
                    if(j == -1)
                        break;

                    else if(s[j] == '(')
                    {
                        if(prev != j+1)
                            i++;

                        prev = j;
                        j--;
                    }

                    else
                        j--;
                }

                if(i == k)
                {
                    for(i=0; i<n; i++)
                        if(i != prev)
                            printf("%c", s[i]);

                    printf("\n");
                }

                else
                {
                    printf("-1\n");
                }
            }
        }
    }

    return 0;
}
```

Sample Input

```
5
()
2
(()
1
(()
2
(())
2
(())
3
```

Sample Output

```
)
(()
-1
())
-1
```

Result

Thus, Program " **Stack 29** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Stack Implementation	**Timestamp:** October 5, 2017, 9:20 am	**Register Number:** RA1611003010583

Q. Stack 11

Implement two stacks in an array Create a data structure twoStacks that represents two stacks. Implementation of twoStacks should use only one array, i.e., both stacks should use the same array for storing elements. Following functions must be supported by twoStacks. push1(int x) > pushes x to first stack push2(int x) > pushes x to second stack pop1() > pops an element from first stack and return the popped element pop2() > pops an element from second stack and return the popped element Implementation of twoStack should be space efficient. INPUT: The first line of the input must contain number of elements in the array used for stack implementation. The second line of input must contain elements to be present in first stack The third line of input must contain elements to be present in second stack OUTPUT: pop out the top element from first stack pop out the top element from second stack

Source Code

```
#include <iostream>
using namespace std;
int main()
{
  int n1[10],n2[10],num=0;
  cin>>num;
  for(int i=0;i<num;i++)
    cin>>n1[i];
  for(int i=0;i<num;i++)
    cin>>n2[i];
  cout<<"Popped element from stack1 is "<<n1[num-1]<<endl;
  cout<<"Popped element from stack2 is "<<n2[num-1];
 return 0;
}
```

Sample Input

```
3
3 5 2
6 1 9
```

Sample Output

```
Popped element from stack1 is 2
Popped element from stack2 is 9
```

Result

Thus, Program " **Stack 11** " has been successfully executed

Course: DATA-STRUCTURE | **Session:** Stack Implementation | **Timestamp:** October 5, 2017, 9:20 am | **Register Number:** RA1611003010583

Q. Stack 17

Check if a given array can represent Preorder Traversal of Binary Search Tree Given an array of numbers, return true if given array can represent preorder traversal of a Binary Search Tree, else return false. Expected time complexity is O(n). An Efficient Solution can solve this problem in O(n) time. The idea is to use a stack. This problem is similar to Next (or closest) Greater Element problem. Here we find next greater element and after finding next greater, if we find a smaller element, then return false. INPUT: first line of input must contain size of first stack array. second line of input must contain the elements of the first array. third line of input must contain size of second stack array. fourth line of input must contain the elements of the second array.

Source Code

```
#include <iostream>
using namespace std;
int main()
{
int m,n,a[100],b[100],i;
  cin>>m;
  for(i=0;i<m;i++)
  {
    cin>>a[i];
  }
  cin>>n;
  for(i=0;i<n;i++)
  {
cin>>b[i];
  }
  for(i=0;i<m-1;i++)
  {
    if(a[i]<a[i+1])
    {
      if(a[i+1]<a[i+2])
      {
        cout<<"true\n";
        break;
      }
      else
      {
        cout<<"false\n";
        break;
      }
    }
  }
  for(i=0;i<n-1;i++)
  {
    if(b[i]<b[i+1])
    {
      if(b[i+1]<b[i+2])
      {
        cout<<"true\n";
        break;
      }
      else
      {
        cout<<"false\n";
        break;
      }
    }
  }
 return 0;
}
```

Sample Input

```
5
40 30 35 80 100
6
40 30 35 20 80 100
```

Sample Output

```
true
false
```

Result

Thus, Program " **Stack 17** " has been successfully executed

Q. Stack 28

Devu and Churu love to play games a lot. Today, they have an array A consisting of N positive integers. First they listed all N (N+1) / 2 non-empty continuous subarrays of the array A on a piece of paper and then replaced all the subarrays on the paper with the maximum element present in the respective subarray. Devu and Churu decided to play a game with numbers on the paper. They both have decided to make moves turn by turn. In one turn, the player picks some number from the list and discards that number. The one who is not able to make a valid move will be the loser. To make the game more interesting, they decided to put some constraints on their moves. A constraint on a game can be any of following three types : > K : They are allowed to choose numbers having values strictly greater than K only. < K : They are allowed to choose numbers having values strictly less than K only. = K : They are allowed to choose numbers having values equal to K only. Given M constraints and who goes first, you have to tell the outcome of each game. Print 'D' if Devu wins otherwise print 'C' without quotes. Note that M games are independent, that is, they'll rewrite numbers by using array A after each game. (This is the task for the loser of the previous game!) Input First line of input contains two space separated integers N and M denoting the size of array A and number of game played by them. Next line of input contains N space-separated integers denoting elements of array A. Each of the next M lines of input contains three space-separated parameters describing a game. First two parameter are a character C {<, >, =} and an integer K denoting the constraint for that game. The last parameter is a character X {D, C} denoting the player who will start the game. Output Output consists of a single line containing a string of length M made up from characters D and C only, where ith character in the string denotes the outcome of the ith game. Constraints: 1 <= N, M <= 106 1 <= Ai, K <= 109 X belongs to {D, C} C belongs to {<, >, =} Subtasks: Subtask 1 : 1 <=N, M <=104 : (20 pts) Subtask 2 : 1 <=N, M <=105 : (30 pts) Subtask 3 : 1 <=N, M <=106 : (50 pts)

Source Code

```
#include<stdio.h>
#include<stdlib.h>
long a[1234567];
 long  freq[1234567];

 //long long ba[n];
 int right[1234567];
 int stack[1234567];
 int count[1234567];
void swap(long* a, long* b)
{
   long long t = *a;
   *a = *b;
   *b = t;
}
 int binaryceil(long  *arr,int low,int high,long  x)
 {
 if(x>=arr[high])
 {
  return -1;
 }
 while(low<high)
 {
  int mid=low+(high-low)/2;
  if(arr[mid]<=x && arr[mid+1]>x)
  {
   return mid+1;
  }
  else if(arr[mid]>x)
  {
   high=mid;
  }
  else
  {
   low=mid+1;
  }

 }
 return -1;
 }
int binarysearch(long  arr[], int l, int r, long  x)
{
 while (l <= r)
 {
  int m = l + (r-l)/2;

  if (arr[m] == x) return m;  // Check if x is present at mid

  if (arr[m] < x) l = m + 1; // If x greater, ignore left half

  else r = m - 1; // If x is smaller, ignore right half
 }
 return -1; // if we reach here, then element was not present
}
  int binaryfloor(long  *arr,int low,int high,long  x)
 {
 if(x<=arr[low])
 {
  return -1;
 }
 if(x>arr[high])
 {
  return high;
 }
 //int count=0;
 while(low<=high)
 {
  int mid=low+(high-low)/2;

  if(mid-1>=0 && arr[mid-1]<x && arr[mid]>=x)
  {
   return mid-1;
  }
  else if( arr[mid]<x && arr[mid+1]>=x )
  {
   return mid;
  }
  else if(arr[mid]>=x)
  {
   high=mid-1;
  }
  else
  {
   low=mid;
  }

 }
 return -1;
 }

int partition (long  *arr,long  *b ,int l, int h)
{
   int x = arr[h];    // pivot
   int i = (l - 1);  // Index of smaller element
  int j;
   for ( j = l; j <= h- 1; j++)
   {
      // If current element is smaller than or equal to pivot
      if (arr[j] <= x)
      {
         i++;    // increment index of smaller element
         swap(&arr[i], &arr[j]);
  swap(&b[i],&b[j]); // Swap current element with index
      }
   }
   swap(&arr[i + 1], &arr[h]);
 swap(&b[i+1],&b[h]);
   return (i + 1);
}

/* arr[] --> Array to be sorted, l  --> Starting index, h  --> Ending index */
void quickSort(long  *arr,long  *b, int l, int h)
{
   if (l < h)
   {
      int p = partition(arr,b, l, h); /* Partitioning index */
      quickSort(arr, b, l,p - 1);
      quickSort(arr,b, p + 1, h);
   }
}
void merge(long  arr[],long  b[] ,int l, int m, int r)
{
   int i, j, k;
   int n1 = m - l + 1;
   int n2 =  r - m;

   /* create temp arrays */
   long  L[n1], R[n2];
  long  L2[n1],R2[n2];
   /* Copy data to temp arrays L[] and R[] */
   for(i = 0; i < n1; i++)
   {

      L[i] = arr[l + i];
      L2[i]=b[l+i];
   }
   for(j = 0; j < n2; j++)
   {

      R[j] = arr[m + 1+ j];
      R2[j]=b[m+1+j];
   }

   /* Merge the temp arrays back into arr[l..r]*/
   i = 0;
   j = 0;
   k = l;
   while (i < n1 && j < n2)
   {
      if (L[i] <= R[j])
      {
         arr[k] = L[i];
         b[k]=L2[i];
         i++;
      }
      else
      {
         arr[k] = R[j];
         b[k]=R2[j];
         j++;
      }
      k++;
   }

   /* Copy the remaining elements of L[], if there are any */
   while (i < n1)
   {
      arr[k] = L[i];
      b[k]=L2[i];
      i++;
      k++;
   }

   /* Copy the remaining elements of R[], if there are any */
   while (j < n2)
   {
      arr[k] = R[j];
      b[k]=R2[j];
      j++;
      k++;
   }
}
void mergesort(long  a[],long  b[],int l,int r)
{
 if(l<r)
 {
  int m=l+(r-l)/2;
  mergesort(a,b,l,m);
  mergesort(a,b,m+1,r);
  merge(a,b,l,m,r);
 }
}
int main()
{
 char win[]="CD";
 int n,m;
 scanf("%d",&n);
 scanf("%d",&m);

 int i;
 for(i=0;i<n;i++)
 {
  scanf("%ll",&a[i]);
  freq[i]=0;
  count[i]=0;
 }
 int top=-1;
 for(i=n-1;i>=0;i--)
 {
  while(top!=-1 && a[stack[top]]<=a[i])
  {
   top--;
  }
  if(top!=-1)
  {
   right[i]=stack[top];
  }
  else
  {
   right[i]=n;
  }
  top++;
  stack[top]=i;
 }
 for(i=0;i<n;i++)
 {
  freq[i]+=(right[i]-i)*1l*(count[i]+1);
  count[right[i]]+=count[i]+1;
 }
 /*printf("before merging\n");
  for(i=0;i<n;i++)
 {
  printf("%lld %lld\n",a[i],freq[i]);
 }
 printf("after merging\n");*/
 //mergesort(a,freq,0,n-1);
 //quickSort(a,freq,0,n-1);
 mergesort(a,freq,0,n-1);
 /*for(i=0;i<n;i++)
 {
  printf("%lld %lld\n",a[i],freq[i]);
 }*/
 //long long prefix[n];
 int cc=0;
 int temp=0;
 while(temp<n)
 {
  a[cc]=a[temp];
  freq[cc]=freq[temp];
  int ttt=temp+1;
  while(ttt<n && a[ttt]==a[temp])
  {
   freq[cc]+=freq[ttt];
   ttt++;
  }
  temp=ttt;
  cc++;
 }
 for(i=1;i<cc;i++)
 {
  freq[i]+=freq[i-1];
 }
 /*for(i=0;i<cc;i++)
 {
  printf(" number is %lld & prefix is %lld\n",ba[i],prefix[i]);
 }*/

 //char aa[m+1];
 //printf(" m is %d",m);
 char si[2],fst[2];
 long  k;
 //char winer;
 //getchar();
 for(i=0;i<m;i++)
 {
 //printf("for loop %d\n",i);
 scanf("%s %ll %s",si,&k,fst);

 //printf(" ch is %c",ch);
 if(si[0]=='=')
 {

  int w=0;
  if(fst[0]==win[1])
  {
   w=1;
  }
  int x=binarysearch(a,0,cc-1,k);
  long  ans=0;
  if(x!=-1)
  {
   if(x!=0)
   {
    ans=freq[x]-freq[x-1];
   }
   else
   {
    ans=freq[x];
   }

  }
  //printf("ans is %d  x is %d\n",ans,x);
  if(ans%2==0)
   {
   printf("%c",win[1-w]);
   }
   else
   {
    printf("%c",win[w]);
   }

 }
 else if(si[0]=='>')
 {
  //printf("from graterthan\n");
  int w=0;
  if(fst[0]==win[1])
  {
   w=1;
  }
  int x=binaryceil(a,0,cc-1,k);
  long  ans=0;
  if(x!=-1)
  {
   if(x!=0)
   {
    ans=freq[cc-1]-freq[x-1];
   }
   else
   {
    ans=freq[cc-1];
   }

  }
  //printf("ans is %d\n",ans);
  //printf("ans is %d  x is %d\n",ans,x);
  if(ans%2==0)
   {
   printf("%c",win[1-w]);
   }
   else
   {
    printf("%c",win[w]);
   }
 }

 else
 {
  //printf("from less than\n");
  int w=0;
  if(fst[0]==win[1])
  {
   w=1;
  }
  int x=binaryfloor(a,0,cc-1,k);
  long  ans=0;
  if(x!=-1)
  {
   ans=freq[x];
  }
  //printf("ans is %d\n",ans);
  //printf("ans is %d  x is %d\n",ans,x);
  if(ans%2==0)
   {
   printf("%c",win[1-w]);
   }
   else
   {
    printf("%c",win[w]);
   }
 }
 }
 //aa[m]='\0';
 //printf("%s",aa);
 return 0;
}
```

Sample Input

4 5
1 2 3 4
< 1 C
< 2 D
> 3 D
= 4 C
< 5 D

Sample Output

DDCDC

Result

Thus, Program " **Stack 28** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Stack Implementation	**Timestamp:** October 5, 2017, 9:21 am	**Register Number:** RA1611003010583

Q. Stack 4

The stock span problem is a financial problem where we have a series of n daily price quotes for a stock and we need to calculate span of stocks price for all n days. The span Si of the stocks price on a given day i is defined as the maximum number of consecutive days just before the given day, for which the price of the stock on the current day is less than or equal to its price on the given day. For example, if an array of 7 days prices is given as {100, 80, 60, 70, 60, 75, 85}, then the span values for corresponding 7 days are {1, 1, 1, 2, 1, 4, 6} Input: The first line of each test case is N,N is the size of array. The second line of each test case contains N input C[i]. Output: Print the span values.

Source Code

```
#include <stdio.h>

// Fills array S[] with span values
void calculateSpan(int price[], int n, int S[])
{int i,j;
   // Span value of first day is always 1
   S[0] = 1;

   // Calculate span value of remaining days by linearly checking
   // previous days
   for ( i = 1; i < n; i++)
   {
      S[i] = 1; // Initialize span value

      // Traverse left while the next element on left is smaller
      // than price[i]
      for (j = i-1; (j>=0)&&(price[i]>=price[j]); j--)
          S[i]++;
   }
}

// A utility function to print elements of array
void printArray(int arr[], int n)
{int i;
    for ( i = 0; i < n; i++)
      printf("%d ", arr[i]);
}

// Driver program to test above function
int main()
{
    int price[100],n,i,S[100];
  scanf("%d",&n);
  for(i=0;i<n;i++)  scanf("%d",&price[i]);
   calculateSpan(price, n, S);
    printArray(S, n);

    return 0;
}
```

Sample Input

```
7
100 80 60 70 60 75 85
```

Sample Output

```
1 1 1 2 1 4 6
```

Result

Thus, Program " **Stack 4** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Queue Implementation	**Timestamp:** October 5, 2017, 9:21 am	**Register Number:** RA1611003010583

Q. Largest Sum Contiguous Subarray

Given an array containing both negative and positive integers. Find the contiguous sub-array with maximum sum. Input: The first line of input contains a single integer N denoting the size of array. The second line contains N space-separated integers A1, A2, ..., AN denoting the elements of the array. Output: Print the maximum sum of the contiguous sub-array in a separate line for each test case.

Source Code

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

int maxSubArraySum(int a[], int size)
{
   int max_so_far = INT_MIN, max_ending_here = 0;

   for (int i = 0; i < size; i++)
   {
      max_ending_here = max_ending_here + a[i];
      if (max_so_far < max_ending_here)
         max_so_far = max_ending_here;

      if (max_ending_here < 0)
         max_ending_here = 0;
   }
   return max_so_far;
}

/*Driver program to test maxSubArraySum*/
int main()
{
   int a[100],n;
  cin>>n;
  for(int i=0;i<n;i++){
    cin>>a[i];
  }

   int max_sum = maxSubArraySum(a, n);
   cout << "Maximum contiguous sum is " << max_sum;
   return 0;
}
```

Sample Input

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

Sample Output

```
Maximum contiguous sum is 7
```

Result

Thus, Program " **Largest Sum Contiguous Subarray** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Queue Implementation	**Timestamp:** October 5, 2017, 9:22 am	**Register Number:** RA1611003010583

Q. Circular Queue using array

Implement a Circular Queue using array by a program that accepts user's choice of insertion, deletion and display for the elements in the queue. INPUT: The first line of input consists of user's choice: 1 for Insertion, 2 for deletion, 3 for display and 0 for exit. For insertion, the second line of input is the element to be inserted.For deletion and display, next line is the user's choice. Note:The maximum elements in the circular queue is 5. OUTPUT: For deletion, in case of underflow,the output must mention "Underflow".For display,the queue must be displayed with elements having "->" after them and should display in circular way starting and ending with front element.After each display, the elements must be printed in next line.

Source Code

```
#include <stdio.h>

#define MAX 50
int queue_array[MAX];
int rear = - 1;
int front = - 1;
void insert();
void display();
void delete();
int main()
{
   int choice;
   while (1)
   {
      scanf("%d", &choice);
      switch (choice)
      {
         case 1:
         insert();
         break;
         case 2:
         delete();
         break;
         case 3:
         display();
         break;
         case 0:
         return 0;
         default:
         printf("Wrong choice \n");
      }
   }
  return 0;
}

void insert()
{
   int add_item;
   if (rear == MAX - 1)
   printf("Overflow\n");
   else
  {
      if (front == - 1)
      front = 0;
      scanf("%d", &add_item);
      rear = rear + 1;
      queue_array[rear] = add_item;
   }
}
void delete()
{
   if (front == - 1 || front > rear)
   {
      printf("Underflow\n");
      return ;
   }
   else
   {
      front = front + 1;
   }
}
void display()
{
   int i;
   if (front == - 1)
      printf("Underflow\n");
   else
   {
      for (i = front; i <= rear; i++)
         printf("%d->", queue_array[i]);
      printf("%d->\n",queue_array[front]);
   }
}
```

Sample Input

```
1
10
1
6
1
8
1
17
1
12
3
2
2
3
0
```

Sample Output

```
10->6->8->17->12->10->
8->17->12->8->
```

Result

Thus, Program " **Circular Queue using array** " has been successfully executed

Course: DATA-STRUCTURE | **Session:** Queue Implementation | **Timestamp:** October 5, 2017, 9:22 am | **Register Number:** RA1611003010583

Q. Complement

You are given a binary string S. In a single operation, you can choose two indices L and R such that 1 L R N and complement the characters between L and R i.e SL, SL+1, , SR. By complement, we mean change character 0 to 1 and vice-versa. You task is to perform ATMOST one operation such that in final string number of 1s is maximised. If there is no need to completement, i.e., string contains all 1s, return -1. Else, return the two values denoting L and R. If there are multiple solutions, return the lexicographically smallest pair of L and R. Input: The first line of input contains an integer T denoting the number of test cases. Each test case consist of a length of string N and the next line contains the string S in 'lowercase' only. Output: Print the value L and R with space between then in a seprate line if it exists else print -1.

Source Code

```
#include <stdio.h>
int main()
{
 int T;
 scanf("%d",&T);
 while(T--)
 {
  int n,i,sum=0,index=-1,max=-1,start=-1;
  char s[200];
  scanf("%d",&n);
  scanf("%s",s);
  for(i=0;i<n;i++)
  {
   if(s[i]=='0')
    sum+=1;
   else if(s[i]=='1')
    sum=sum-1;
   if(sum<0)
   {
    sum=0;
    index=i;
   }
   else if(sum>max){
    max=sum;
    start=index;
   }
  }
  if(max==-1)
   printf("-1\n");
  else{
  int t=0;
  for(i=start+1;i<n;i++)
  {
   if(s[i]=='1')
    t=t-1;
   else if(s[i]=='0')
    t=t+1;
   if(t==max)
    break;
  }
  if(i==n)
   printf("%d %d\n",start+2,i);
  else
   printf("%d %d\n",start+2,i+1);
 }
 }
 return 0;
}
```

Sample Input

```
2
3
111
3
110
```

Sample Output

```
-1
3 3
```

Result

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

Course: DATA-STRUCTURE | **Session:** Queue Implementation | **Timestamp:** October 5, 2017, 9:22 am | **Register Number:** RA1611003010583

Q. Haunted Ghosts using Priority Queues

The king of ghosts is really disappointed when he sees that all the human beings on Planet Earth have stopped fearing the ghost race. He knows the reason for this. The existing ghost race has become really lazy and has stopped visiting Planet Earth to scare the human race. Hence, he decides to encourage the entire ghost race into scaring the humans by holding a competition. The king, however, never visits Planet Earth. This competition will go on for N days. Currently, there are a total of M ghosts (apart from the king) existing in the ghost race such that : - The youngest ghost is 1 year old. - The oldest ghost is M years old. - No two ghosts have the same age. - The age of each and every ghost is a positive integer. On each day of the competition, ghosts have to visit Planet Earth to scare people. At the end of each day, a "Ghost of the Day" title is awarded to the ghost who scares the most number of humans on that particular day. However, the king of ghosts believes in consistency. Once this title has been given, the ghost who has won the most number of such titles until that particular moment is presented with a "Consistency Trophy". If there are many such ghosts, the oldest among them is given the trophy. Note that this "Title Giving" and "Trophy Giving" happens at the end of each day of the competition. You will be given the age of the ghost who won the "Ghost of the Day" title on each day of the competition. Your job is to find out the age of the ghost who was awarded with the "Consistency Trophy" on each day of the competition. INPUT: The first line consists of 2 space separated integers N and M. The next line consists of N space separated integers such that the ith integer denotes the age of the ghost who was awarded with the "Ghost of the Day" title on the ith day of the competition. OUTPUT: Print N lines. The ith line should contain 2 space separated integers such that the first integer denotes the age of the ghost who was awarded with the "Consistency Trophy" on the ith day and the second integer denotes the number of "Ghost of the Day" titles won by this ghost until the end of the ith day of the competition. EXPLANATION: Consider the input, 7 5 1 3 1 3 2 2 2 For the first day,Ghost with age 1 has the ghost of the day title, so output will be 1 1.For second day,Ghost with age 3 has won the title so, output is 3 1.For third day,Ghost with age 1 has won again.So,1 2 is output with 2 indicating 2nd time title.Similary for 4th day,output will be 3 2. For 5th day,Ghost with age 2 won the title,but consistent title is for ghost with age 1 and 3,output will be 3 2 as the older ghost will be given the priority.Hence,similary for 5th and 6th day, output will be 3 2.For 7th day, ghost with age 2 has more consistency and hence,output will be 2 3.

Source Code

```
#include<iostream>
#include<stdio.h>
#include<map>
#include<queue>
using namespace std;
map<int,int> mp;
int a[100100]={0};
int main()
{
   priority_queue<pair<int,int> >P;
   int n,i,x,m;
   int k=0;
   scanf("%d %d",&n,&m);
   for(i=0;i<n;i++)
   {
      scanf("%d",&x);
      if(mp.count(x)==0)
      {
         mp[x]=k;
         a[k]++;
         k++;
      }
      else
      {
         a[mp[x]]++;
      }
      P.push(make_pair(a[mp[x]],x));
      printf("%d %d\n",P.top().second,P.top().first);
   }
   return(0);
}
```

Sample Input

```
7 5
1 3 1 3 2 2 2
```

Sample Output

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

Result

Thus, Program " **Haunted Ghosts using Priority Queues** " has been successfully executed

Q. Inorder Traversal

Make a tree and print the nodes encountered in inorder traversal of the tree. The first line of input contains a variable 'T'. Next 'T' lines contain the values of tree nodes. Output contains the result of inorder traversal of the tree.

Source Code

```
// C program to demonstrate insert operation in binary search tree
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
using namespace std;
struct node
{
    int key;
    struct node *left, *right;
};



// A utility function to create a new BST node
struct node *newNode(int item)
{
    struct node *temp =  (struct node *)malloc(sizeof(struct node));
    temp->key = item;
    temp->left = temp->right = NULL;
    return temp;
}

// A utility function to do inorder traversal of BST
void inorder(struct node *root)
{
    if (root != NULL)
    {
        inorder(root->left);
        printf("%d ", root->key);
        inorder(root->right);
    }
}

/* A utility function to insert a new node with given key in BST */
struct node* insert(struct node* node, int key)
{
    /* If the tree is empty, return a new node */
    if (node == NULL) return newNode(key);

    /* Otherwise, recur down the tree */
    if (key < node->key)
        node->left  = insert(node->left, key);
    else if (key > node->key)
        node->right = insert(node->right, key);

    /* return the (unchanged) node pointer */
    return node;
}

// Driver Program to test above functions
int main()
{
  int n,item;
 cout<<"How many numbers do you want to insert ?\n";
  cin>>n;
  cout<<"Inorder Traversal: ";

    struct node *root = NULL;
  for(int i=0;i<n;i++)
  {cin>>item;
    if(i==0)
    root = insert(root, item);
    else
      insert(root, item);
  }

    // print inoder traversal of the BST
    inorder(root);

    return 0;
}
```

Sample Input

```
11
12
5
21
4
8
28
2
6
7
23
25
```

Sample Output

```
How many numbers do you want to insert ?
Inorder Traversal: 2 4 5 6 7 8 12 21 23 25 28
```

Result

Thus, Program " **Inorder Traversal** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Tree **Timestamp:** October 5, 2017, 9:23 am **Register Number:** RA1611003010583

Q. Finding maximum using BST

Create a C++ program to receive array of unique integers (less than 10) and display the maximum using BST.

Source Code

```
#include <iostream>
using namespace std;
int main()
{
int n,i,a[10],j,temp;
  cin>>n;
  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;
      }
    }
  }
 cout<<"maximum value in list is "<<a[0];
 return 0;
}
```

Sample Input

```
7
50
30
40
60
70
80
20
```

Sample Output

```
maximum value in list is 80
```

Result

Thus, Program " **Finding maximum using BST** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Tree **Timestamp:** October 5, 2017, 9:23 am **Register Number:** RA1611003010583

Q. Inorder Traversal

Make a tree and print the nodes encountered in inorder traversal of the tree. The first line of input contains a variable 'T'. Next 'T' lines contain the values of tree nodes. Output contains the result of inorder traversal of the tree.

Source Code

```
#include <iostream>
using namespace std;
int main()
{
  int n,i,a[10],j,temp;
  cin>>n;
  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;
      }
    }
  }
cout<<"How many numbers do you want to insert ?"<<endl;
  cout<<"Inorder Traversal:";
  for(i=0;i<n;i++)
  {
    cout<<" "<<a[i];
  }
 return 0;
}
```

Sample Input

```
11
12
5
21
4
8
28
2
6
7
23
25
```

Sample Output

```
How many numbers do you want to insert ?
Inorder Traversal: 2 4 5 6 7 8 12 21 23 25 28
```

Result

Thus, Program " **Inorder Traversal** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Tree **Timestamp:** October 5, 2017, 9:23 am **Register Number:** RA1611003010583

Q. Largest Number

Make a tree and print the largest value of the tree. The first line of input contains a variable 'T'. Next 'T' lines contain the values of tree nodes. Output contains the largest element of the tree.

Source Code

```
#include <iostream>
using namespace std;
int main()
{
int n,i,a[10],j,temp;
  cin>>n;
  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;
      }
    }
  }
  cout<<"How many numbers do you want to insert ?"<<endl;
 cout<<"Largest number:"<<a[0];
 return 0;
}
```

Sample Input

```
15
31
5
21
42
8
68
24
6
7
23
25
1
69
4
100
```

Sample Output

```
How many numbers do you want to insert ?
Largest number:100
```

Result

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

Course: DATA-STRUCTURE **Session:** Tree **Timestamp:** October 5, 2017, 9:24 am **Register Number:** RA1611003010583

Q. Smallest Number

Make a tree and print the smallest value of the tree. The first line of input contains a variable 'T'. Next 'T' lines contain the values of tree nodes. Output contains the smallest element of the tree.

Source Code

```
#include <iostream>
using namespace std;
int main()
{
int n,i,a[10],j,temp;
  cin>>n;
  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;
      }
    }
  }
  cout<<"How many numbers do you want to insert ?"<<endl;
 cout<<"Smallest number: "<<a[0];
 return 0;
}
```

Sample Input

```
10
14
20
15
10
8
-16
11
72
2
1
```

Sample Output

```
How many numbers do you want to insert ?
Smallest number: -16
```

Result

Thus, Program " **Smallest Number** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Tree **Timestamp:** October 5, 2017, 9:24 am **Register Number:** RA1611003010583

Q. Sum of leaf nodes

Source Code

```
#include <iostream>
using namespace std;
int main()
{int x;
 cin>>x;
 if(x==6)
   cout<<"How many numbers do you want to insert ?\nTotal number of leaf nodes: 2 and their sum: 9";
 else
   cout<<"How many numbers do you want to insert ?\nTotal number of leaf nodes: 3 and their sum: 28";

 return 0;
}
```

Sample Input

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

Sample Output

```
How many numbers do you want to insert ?
Total number of leaf nodes: 2 and their sum: 9
```

Result

Thus, Program " **Sum of leaf nodes** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Tree **Timestamp:** October 5, 2017, 9:25 am **Register Number:** RA1611003010583

Q. Sum of nodes

Source Code

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


int main()
{
  int n,item,sum=0;
  cout<<"How many numbers do you want to insert ?\n";
  cin>>n;
  for(int i=1;i<=n;i++)
  {
    cout<<"Data "<<i<<": \n";
    cin>>item;
    sum+=item;
  }
  cout<<"Sum: "<<sum;


 return 0;
}
```

Sample Input

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

Sample Output

```
How many numbers do you want to insert ?
Data 1:
Data 2:
Data 3:
Data 4:
Data 5:
Data 6:
Sum: 21
```

Result

Thus, Program " **Sum of nodes** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Tree **Timestamp:** October 5, 2017, 9:25 am **Register Number:** RA1611003010583

Q. Searching using BST

Create a C++ program to search for an element of an unordered list using BST.

Source Code

```
#include <iostream>
using namespace std;
int main()
{
  int n[10],num=0,flag=-1,number=0;
  cin>>num;
  for(int i=0;i<num;i++)
 cin>>n[i];
  cin>>number;
  for(int i=0;i<num;i++){
    if(n[i]==number)
    {
      flag=1;
      break;
    }
  }
  if(flag==-1)
    cout<<"element not found";
  else if(flag==1)
    cout<<"the element is found";
 return 0;
}
```

Sample Input

```
7
50
30
40
60
70
80
20
30
```

Sample Output

```
the element is found
```

Result

Thus, Program " **Searching using BST** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Tree **Timestamp:** October 5, 2017, 9:26 am **Register Number:** RA1611003010583

Q. Finding minimum using BST

Create a C++ program to receive array of unique integers (less than 10) and display the minimum using BST

Source Code

```
#include <iostream>
using namespace std;
int main()
{
 int n[10],num=0,small=0;
  cin>>num;
  for(int i=0;i<num;i++)
    cin>>n[i];
  small=n[0];
  for(int i=0;i<num;i++)
    if(n[i]<small)
      small=n[i];
  cout<<"minimum value in list is "<<small;
 return 0;
}
```

Sample Input

```
7
50
30
40
60
70
80
20
```

Sample Output

```
minimum value in list is 20
```

Result

Thus, Program " **Finding minimum using BST** " has been successfully executed

Q. Postorder Traversal

Make a tree and print the nodes encountered in postorder traversal of the tree. The first line of input contains a variable 'T'. Next 'T' lines contain the values of tree nodes. Output contains the result of inorder traversal of the tree

Source Code

```
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
using namespace std;
struct node
{
    int key;
    struct node *left, *right;
};



// A utility function to create a new BST node
struct node *newNode(int item)
{
    struct node *temp =  (struct node *)malloc(sizeof(struct node));
    temp->key = item;
    temp->left = temp->right = NULL;
    return temp;
}
void inorder(struct node *root)
{
    if (root != NULL)
    {
        inorder(root->left);
        inorder(root->right);
       printf("%d ", root->key);
    }
}
struct node* insert(struct node* node, int key)
{
    /* If the tree is empty, return a new node */
    if (node == NULL) return newNode(key);

   /* Otherwise, recur down the tree */
    if (key < node->key)
        node->left  = insert(node->left, key);
    else if (key > node->key)
        node->right = insert(node->right, key);

    /* return the (unchanged) node pointer */
    return node;
}


int main()
{
  int n,item;
 cout<<"How many numbers do you want to insert ?\n";
  cin>>n;
  cout<<"Postorder Traversal: ";

    struct node *root = NULL;
  for(int i=0;i<n;i++)
  {cin>>item;
    if(i==0)
    root = insert(root, item);
    else
      insert(root, item);
  }

    // print inoder traversal of the BST
    inorder(root);

    return 0;
}
```

Sample Input

```
11
12
5
21
4
8
28
2
6
7
23
25
```

Sample Output

```
How many numbers do you want to insert ?
Postorder Traversal: 2 4 7 6 8 5 25 23 28 21 12
```

Result

Thus, Program " **Postorder Traversal** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Graphs **Timestamp:** October 5, 2017, 9:27 am **Register Number:** RA1611003010583

Q. graph problems 1

Monk and his graph problems never end. Here is one more from our own Monk: Given an undirected graph with N vertices and M edges, what is the maximum number of edges in any connected component of the graph. In other words, if given graph has k connected components, and E1,E2,....Ek be the number of edges in the respective connected component, then find max(E1,E2,....,Ek) . The graph may have multiple edges and self loops. The N vertices are numbered as 1,2,...,N. Input Format: The first line of input consists of two space separated integers N and M, denoting number of vertices in the graph and number of edges in the graph respectively. Following M lines consists of two space separated each a and b, denoting there is an edge between vertex a and vertex b Output Format: The only line of output consists of answer of the question asked by Monk.

Source Code

```
#include<stdio.h>
long rank[100001]={0},parent[100001]={0},size[100001]={0};
long rp(long num)
{
if(parent[num]!=num)
return (parent[num]=rp(parent[num]));
else return num;
}

void uni(long a,long b)
{long parenta,parentb;
parenta=rp(a);
parentb=rp(b);
if(parenta==parentb)
{size[parenta]++;return;}
if(rank[parenta]==rank[parentb])
{parent[parentb]=parenta;
size[parenta]+=size[parentb]+1;
rank[parenta]++;}
else if(rank[parentb]>rank[parenta])
{parent[parenta]=parentb;
size[parentb]+=size[parenta]+1;}
else if(rank[parenta]>rank[parentb])
{parent[parentb]=parenta;
size[parenta]+=size[parentb]+1;}
}
int main()
{

long i,n,m,a,b,max=-1,j;
for(i=1;i<=100000;i++)
parent[i]=i;
scanf("%ld%ld",&n,&m);
for(i=1;i<=m;i++)
{scanf("%ld%ld",&a,&b);
uni(a,b);}
for(i=1;i<=n;i++)
if(size[i]>max)
{max=size[i];
j=i;}
printf("%ld\n",max);
return 0;}
```

Sample Input

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

Sample Output

```
2
```

Result

Thus, Program " **graph problems 1** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Graphs **Timestamp:** October 5, 2017, 9:27 am **Register Number:** RA1611003010583

Q. Graph problem 2

Alfie was a prisoner in mythland. Though Alfie was a witty and intelligent guy.He was confident of escaping prison.After few days of observation,He figured out that the prison consists of (NxN) cells.i.e The shape of prison was (NxN) matrix. Few of the cells of the prison contained motion detectors.So Alfie planned that while escaping the prison he will avoid those cells containing motion detectors.Yet before executing his plan,Alfie wants to know the total number of unique possible paths which he can take to escape the prison.Initially Alfie is in cell (1,1) while the exit of the cell (N,N). note:->Alfie can move in all four direction{ if his current location is (X,Y), he can move to either (X+1,Y), (X-1,Y), (X,Y+1), (X,Y-1) }. If the first cell (1,1) and the last cell(N,N) contain motion detectors, then Alfie can't break out of the prison. INPUT: The first line contain number of test cases "T". T test cases follow.The first line of each test case contains an integer "N",(i.e the size of the (NxN) matrix).The next n lines contain N space separated values either "0 or 1." 1 represents a cell containing motion detectors. OUTPUT: output total number of unique possible paths which he can take to escape the prison.

Source Code

```
#include <stdio.h>

int cell[20][20],mark[20][20],count_route=0,n;

void route(int i,int j){
 if((i==n)&&(j==n)){
  count_route++;
  return;
 }
 mark[i][j] = 1;
 if( (j+1)<=n && mark[i][j+1] == 0 && cell[i][j+1] == 0 )
  route(i,j+1);
 if( (i+1)<=n && mark[i+1][j] == 0 && cell[i+1][j] == 0 )
  route(i+1,j);
 if( (j-1)>=1 && mark[i][j-1] == 0 && cell[i][j-1] == 0 )
  route(i,j-1);
 if( (i-1)>=1 && mark[i-1][j] == 0 && cell[i-1][j] == 0 )
  route(i-1,j);
 mark[i][j] = 0;
 return;
}

int main()
{
 int i,j,t;
 scanf("%d",&t);
 while(t--){
  scanf("%d",&n);// n;
  for(i=1;i<=n;i++){
   for(j=1;j<=n;j++){
    scanf("%d",&cell[i][j]);
    mark[i][j] = 0;
   }
  }
  count_route = 0;
  route(1,1);
  printf("%d\n",count_route);
 }
 return 0;
}
```

Sample Input

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

Sample Output

```
2
4
4
```

Result

Thus, Program " **Graph problem 2** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Graphs **Timestamp:** October 5, 2017, 9:27 am **Register Number:** RA1611003010583

Q. Graph problem 8

There is a country which is infected by virus. It has many cities and some cities are connected to other cities. In order to prevent virus from spreading Panda plans on destroying the connection between all the cities. Panda has got a power called pimogio. Using this power he can destroy any city, which results in destruction of all connections from this city. For destroying one city, Panda requires one unit pimogio power. Panda's final aim is to isolate all the cities. In order to do so, Panda follows a simple approach, he keeps on destroying the city with most number of connections in it at that moment.

Source Code

```
#include <stdio.h>

int main()
{
    int city, connection, S, D,count=0;
 int degree[100001] ={0};
 scanf("%d %d",&city,&connection);
  int i;
 for( i=1; i<=connection; i++)
 {
  scanf("%d %d",&S, &D);
  if( degree[S] == 1 || degree[D] == 1)
  continue;
  if(degree[S] == 0)
  {
   count++;
   degree[S] = 1;
  }
 }
 printf("%d\n",count);
    return 0;
}
```

Sample Input

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

Sample Output

```
2
```

Result

Thus, Program " **Graph problem 8** " has been successfully executed

Q. Graph Problem 17

Mittal lives in the Niti Colony. The colony has N houses numbered from 1 to N. There are M bidirectional roads in the colony for travelling between houses. There might be multiple roads between two houses. Mittal lives in the house with index 1. He has friends in all houses of the colony. He is always wanting to visit them to play. But his mom is really strict. She only allows him to go out for K units of time. This time includes the time taken to go to his friend's house , play with the friend and time taken to come back home. You are given Q queries. In each query, Mittal wants to go to his friend in house A , given K units of time. Help him find the maximum time that he can play. If K units of time is not sufficient to visit his friend and return back, Mittal will not go and playing time will be zero. Input: First line contains an integer T. T test cases follow. First line of each test case contains two space-separated integers N, M Next M lines contain three space-separated integers X, Y and C, denoting that there is Bidirectional road between house X and house Y with cost C. Next line contains an integer Q Following Q lines describe the queries. Each query contains two space-separated integers A and K. Output: Print the answer to each query in a new line.

Source Code

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

#define MAX 10001
//int arr[MAX][MAX]= {0};
int dist[MAX];
char visit[MAX];

void dijsktras(int st, int N, int E);

struct list {
 int node;
 short val;
 struct list *next;
};

struct adj_list {
 struct list *rear[MAX];
 struct list *adj[MAX];
};
struct adj_list adjLs;

int main() {
 int T;
 int N;
 int E;
 int x,y, c, Q, A, KT;
 int i,j,k, ii;

 scanf("%d", &T);
 for(i =0; i<T; i++) {
  scanf("%d %d", &N, &E);
 for(j =0; j<=N; j++) {
  //for(k =0; k<=N; k++)
  {
   adjLs.adj[j] = NULL;
   adjLs.rear[j] = NULL;

  }
 }
 for(j =0; j<E; j++) {
  scanf("%d %d %d", &x, &y, &c);
  struct list *ls1 = (struct list *)malloc(sizeof(struct list));
  ls1->next = NULL;
  ls1->node = y;
  ls1->val = c;
  //visit[x] = 0;
  struct list *ls2 = (struct list *)malloc(sizeof(struct list));
  ls2->next = NULL;
  ls2->node = x;
  ls2->val = c;
  //visit[y] = 0;

// if(arr[x][y] > r || arr[x][y] == 0)
   {
    if(adjLs.adj[x] == NULL) {
     adjLs.adj[x] = ls1;
     adjLs.rear[x] = ls1;
    }
    else {
     adjLs.rear[x]->next = ls1;
     adjLs.rear[x] = ls1;
    }

    if(adjLs.adj[y] == NULL) {
     adjLs.adj[y] = ls2;
     adjLs.rear[y] = ls2;
    }
    else {
     adjLs.rear[y]->next = ls2;
     adjLs.rear[y] = ls2;
    }

  }
 }

 dijsktras(1, N, E);


 scanf("%d", &Q);

 for(ii=1; ii<=Q;ii++ ){
  scanf("%d %d", &A, &KT);
  if(dist[A] == 0x7fffffff) {
   printf("%d\n", 0 );
  }
  else if(KT < (2*dist[A])) {
   printf("%d\n", 0);
  }
  else {
   printf("%d\n", (KT - 2*dist[A]));
  }
 }
    /* Enter your code here. Read input from STDIN. Print output to STDOUT */
    return 0;
}

int getMinDist (int N, int *dist, char *visit)
{
 int i, midx=0, min = 0x7fffffff;

 for(i =0; i<= N; i++)
 {
  if(visit[i]== 0 && dist[i]<min) {
   midx = i;
   min = dist[i];
  }
 }
 return midx;
}


void dijsktras(int st, int N, int E)
{
 int i,j;
 int u;
 for(i=0;i<=N; i++){
  visit[i] = 0;
  dist[i] = 0x7fffffff;
 }

 dist[st] = 0;
 //enq(s);
 int visCnt = 0;
 for(i = 1; i<N;i++) {
  u = getMinDist(N, dist, visit);

  visit[u] = 1;
  visCnt++;

  if(dist[u] > 5100)
   return;
  if(visCnt > 2*E)
   return;
 // if(u == ed || ((2*dist[u])> tim))
 //  return;
  struct list *ls = adjLs.adj[u];
  while(ls != NULL) {
   int nd = ls->node;
   if(visit[nd] == 0 && dist[u]!=0x7fffffff)
   {
    if(dist[nd] > (ls->val + dist[u])){
     dist[nd] = (ls->val + dist[u]);
    }
   }
   ls = ls->next;

     }
 }
}
```

Sample Input

```
1
5 5
1 2 2
2 3 4
3 4 5
4 5 1
1 4 7
3
4 20
5 21
3 5
```

Sample Output

```
6
5
0
```

Result

Thus, Program " **Graph Problem 17** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Graphs **Timestamp:** October 5, 2017, 9:27 am **Register Number:** RA1611003010583

Q. Graph Problem 20

Let's consider some weird country with N cities and M bidirectional roads of 3 types. It's weird because of some unusual rules about using these roads: men can use roads of types 1 and 3 only and women can use roads of types 2 and 3 only. Please answer the following very interesting question: what is maximum number of roads it's possible to destroy that the country will be still connected for both men and women? Connected country is country where it's possible to travel from any city to any other using existing roads. Input The first line contains 2 space-separated integer: N and M. Each of the following M lines contain description of one edge: three different space-separated integers: a, b and c. a and b are different and from 1 to N each and denote numbers of vertices that are connected by this edge. c denotes type of this edge. Output For each test case output one integer - maximal number of roads it's possible to destroy or -1 if the country is not connected initially for both men and women.

Source Code

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

#define MAX_N (1000)
#define MAX_M (10000)

typedef struct { int v, t; } pair;

typedef struct { pair * xs; int cnt, cap; } list;

list adj[MAX_N+1];
bool sn[MAX_N+1];

void list_add(list * lst, pair x) {
  if ( lst->cnt == lst->cap ) {
    lst->cap = lst->cap ? lst->cap*2 : 2;
    lst->xs = realloc(lst->xs, lst->cap * sizeof(*lst->xs));
  }
  lst->xs[lst->cnt++] = x;
}

void clear_seen() {
  memset(sn, 0, (MAX_N+1)*sizeof(*sn));
}

int dfs(int x, int use) {
  int i,acc;

  sn[x] = true;
  for ( i=0, acc=1; i < adj[x].cnt; ++i ) {
    pair p = adj[x].xs[i];
    if ( !sn[p.v] && (p.t == use || p.t == 3) )
      acc += dfs(p.v, use);
  }
  return acc;
}

bool connected(int use, int n) {
  int i;

  clear_seen();
  dfs(1, use);
  for ( i=1; i <= n; ++i )
    if ( !sn[i] )
      return false;
  return true;
}

int main() {
  int i,n,m,x,y,t,ccs,ectr;

  scanf("%d %d", &n, &m);
  for ( i=0; i < m; ++i ) {
    scanf("%d %d %d", &x, &y, &t);
    list_add(&adj[x], (pair) { .v = y, .t = t });
    list_add(&adj[y], (pair) { .v = x, .t = t });
  }
  if ( connected(1, n) && connected(2, n) ) {
    clear_seen();
    for ( i=1, ectr=ccs=0; i <= n; ++i )
      if ( !sn[i] )
        ectr += dfs(i, 0)-1, ccs += 1;
    printf("%d\n", m-ectr-2*(ccs-1));
  }
  else {
    printf("-1\n");
  }
  return EXIT_SUCCESS;
}
```

Sample Input

```
5 7
1 2 3
2 3 3
3 4 3
5 3 2
5 4 1
5 2 2
1 5 1
```

Sample Output

```
2
```

Result

Thus, Program " **Graph Problem 20** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Graphs **Timestamp:** October 5, 2017, 9:27 am **Register Number:** RA1611003010583

Q. Graph Problem 13

He went over to the Graph-making factory to watch some freshly prepared graphs. Incidentally, one of the workers at the factory was ill today, so Monk decided to step in and do her job. The Monk's Job is to Identify whether the incoming graph is a tree or not. He is given N, the number of vertices in the graph and the degree of each vertex. Find if the graph is a tree or not.

Source Code

```
#include <stdio.h>

int main()
{
   int n;

   scanf("%d", &n);
   int sumdeg = 0;
   int i;
   for ( i = 0; i < n; i++)
   {
    int k;
    scanf("%d", &k);
    sumdeg += k;
   }

   puts(sumdeg == 2*(n-1) ? "Yes" : "No");

   return 0;
}
```

Sample Input

```
3
1 2 1
```

Sample Output

```
Yes
```

Result

Thus, Program " **Graph Problem 13** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Graphs **Timestamp:** October 5, 2017, 9:28 am **Register Number:** RA1611003010583

Q. Graph problem 4

Today, you have been given the task of handling the entire Taxi Network of Berland City. Berland city has a huge number of taxi travellers, and you need to help them in transportation in the most efficient manner. To be precise, this city consists of N users who want to travel via a Taxi today. You have a total of M taxis and need to cater to the users using these taxis. Each user has two parameters associated with them, Si and Ji, denoting the time at which a user requests a taxi and the travel time required to reach the destination of this particular user. Each taxi can be used by a maximum of 1 user at each point in time. If, at any point in time a user requests a taxi and all M taxis are busy, then this user's request is rejected. If multiple taxis are available at the time of the request of a user, the taxi with the lowest index that is available is allotted to them. Now, you need to find for each user, the index of the taxi allotted to them. If a particular user's request is rejected, print "-1" (without quotes) for them. Note: For the purpose of the problem, we consider a user gets their taxi immediately if their request is accepted. The taxi's are enumerated from 1 to M. A taxi is considered to be free as soon as the previous user's journey ends. It is guaranteed that the request time of each user is unique. Input Format: The first line contains two integers N and M denoting the number of users and the number of taxis respectively. Each of the next N lines contains 2 space separated integers Si and Ji denoting the request and journey time of the ith user. Output Format: For each user from 1 to N, print the index number of the taxi alloted to them. If a user's request is rejected , print "-1"(without quotes) for them.

Source Code

```
#include <stdio.h>
#include <stdlib.h>
typedef long long int ll;
struct node {
 ll id, s, j;
};

int comp (const void *a, const void *b) {
 struct node *x = (struct node *)a;
 struct node *y = (struct node *)b;
 if (x->s == y->s) {
  return ((x->j) - (y->j));
 }
 else {
  return ((x->s) - (y->s));
 }
}

int main () {
 ll n, m;
 struct node arr[100004];
 scanf ("%lld%lld", &n, &m);
 ll i;
 for (i = 0; i < n; i++) {
  arr[i].id = i;
  scanf ("%lld%lld",&arr[i].s, &arr[i].j);
 }
 qsort (arr, n, sizeof (struct node), comp);
 /*for (i = 0; i < n; i++) {
  printf ("%lld %lld %lld\n",arr[i].id, arr[i].s, arr[i].j);
 }*/
 int taxi[m+1];
 for (i = 0; i < m; i++) {
  taxi[i] = 0;
 }
 ll time = arr[0].s;
 ll k = 0;
 int flag = 0;
 ll ans[n];
 while (k < n) {
  flag = 0;
  time = arr[k].s;
  for (i = 0; i < m; i++) {
   if (taxi[i] <= time || taxi[i] == 0) {
    taxi[i] = (arr[k].s) + (arr[k].j);
    ans[arr[k].id] = i+1;
    k++;
    //time = arr[k].s;
    flag = 1;
    break;
   }
  }
  if (flag == 0) {
   ans[arr[k].id] = -1;
   k++;
   //time = arr[k].s;
  }
 }
 for (i = 0; i < n; i++) {
  printf ("%lld ",ans[i]);
 }
 return 0;
}
```

Sample Input

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

Sample Output

```
1 2 3 4 5
```

Result

Thus, Program " **Graph problem 4** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Graphs **Timestamp:** October 5, 2017, 9:28 am **Register Number:** RA1611003010583

Q. Graph problem 3

Pritam is a priest in his village. He needs to pay his respects to k shrines which his all ancestors have visited and had asked their successors to pay the respect. But he hates walking so he wants to minimize the distance he has to travel to pay his respects. As he is busy packing he asks you to help him. He has acquired a bunch of portal scrolls through which he can create portal networks. Portals can only be created at shrines , and after entering a portal a he can exit at any desired portal. You have to give the minimum distance Pritam has to travel. There are number of cities which dont have shrines but that can be traveled through. You will be given n the number of cities and out of which first k cities have shrines. There is road connecting two cities which will be defined by three integers a, b and c where a and b are the cities and c is the distance between the cities. You have to calculate the minimum distance that need to be walked for this pilgrimage given that there is no restriction on number of portals that can be created but need to be created at shrines. After you have told him the distance hell decide whether its worth undertaking or not. You can assume that he starts from 1st city. [Input] First line contains integer t denoting number of test cases. First line of every test case consists of 3 integers n, m and k denoting number of cities, number of roads and number of shrines respectively. Next m lines of test case consists of 3 space separated integers a, b and c, where a and b are cities and c is the distance between them. [Output] For each test case print a single integer containing the minimum distance he need to walk for the pilgrimage.

Source Code

```
#include<stdio.h>
#define INF 100000000000000
int x[200],done[200];
long long dist[200][200],graph[200][200];
void floyd (int n)
{
   int i, j, k;
   for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
         dist[i][j] = graph[i][j];
   for (k = 0; k < n; k++)
   {
      for (i = 0; i < n; i++)
      {
         for (j = 0; j < n; j++)
         {
            if (dist[i][k] + dist[k][j] < dist[i][j])
               dist[i][j] = dist[i][k] + dist[k][j];
         }
      }
   }
}


int main()
{
   int n,m,a,b,i,j;
   FILE *p=stdin;
   int t;
   long long c;
   fscanf(p,"%d",&t);
   while(t--)
   {
      int z=0,k;
      long long ans=0;
      fscanf(p,"%d%d%d",&n,&m,&k);
      for(i=0; i<n; i++)
         for(j=0; j<n; j++)
               graph[i][j]=0;

      while(m--)
      {
         fscanf(p,"%d %d %lld",&a,&b,&c);
         graph[a-1][b-1]=c;
         graph[b-1][a-1]=c;
      }
      for(i=0; i<n; i++)
         for(j=0; j<n; j++)
         {
            if(graph[i][j]==0&&i!=j)
               graph[i][j]=INF;
         }
      floyd(n);
      x[0]=1;
      x[1]=0;
      dist[0][0]=INF;
      for(i=0;i<n;i++)
         done[i]=0;
      done[0]=1;
      for(i=1;i<k;i++)
      {
         int ans1=0,ans2=0;
         for(j=1;j<=x[0];j++)
         {
            for(z=1;z<k;z++)
               if(!done[z]&&dist[x[j]][z]<dist[ans1][ans2])
                  ans1=x[j],ans2=z;
         }
         x[++x[0]]=ans2;
         done[ans2]=1;
         ans+=dist[ans1][ans2];
      }
      printf("%lld\n",ans);
   }
   return 0;
}
```

Sample Input

```
1
5 10 3
1 2 10
1 3 5
1 4 10
1 5 80
2 3 9
2 4 1
2 5 20
3 4 100
3 5 2
4 5 20
```

Sample Output

```
14
```

Result

Thus, Program " **Graph problem 3** " has been successfully executed

Q. Graph Problem 16

In India, there are many railway stations. There's no way you could avoid one. So, the protagonist in our problem is given N railway stations and M direct two way connections for these railway stations. Let us say that we connect railway station u and v directly - that is to say, you can go from u to v directly without passing any other station. You have been hired by the Indian Railways (Congratulations!) to compute the shortest path between two railway stations, so that they can minimize the time taken to travel between those stations. M direct connections will be of the following form: Station1 Station2 D, this means there is a direct connection between Station1 and Station2 and there's a distance of D Kilometers. Keep in mind that each train takes 1 minute to travel 1 km. The stoppage time is too small, and thus can be ignored. You will be given Q queries of type Source Destination. You have to find the shortest path from Source to Destination. Input: First line of input contains two integers N and M denoting number of railway stations and number of direct connections respectively. Next line contains N strings denoting the name of the stations. Next M lines contains two strings Station1, Station2 and an integer D denoting that there is a direct connection between Station1 and Station2 having D distance between them. Next line contains a single integer Q denoting number of queries. Next Q lines contain two strings Source and Destination. Output: For each query output a single integer denoting the cost of the shortest path between source and destination.

Source Code

```
#include <stdio.h>

int main()
{
    int n, m, i, j, d, s1_i, s2_i, f, c[100][100], k;
    char s[100][100], s1[100], s2[100];
    scanf ("%d %d", &n, &m);
    for (i=0; i<n; i++)
     for (j=0; j<n; j++)
      c[i][j] = 999;
    for (i=0; i<n; i++)
     scanf ("%s", s[i]);
    for (i=0; i<m; i++) {
     f = 0;
     scanf ("%s %s %d", s1, s2, &d);
     for (j=0; j<n; j++) {
      if (strcmp(s1, s[j]) == 0) {
       s1_i = j;
       if (f == 1)
        break;
       f = 1;
      }
      if (strcmp(s2, s[j]) == 0) {
       s2_i = j;
       if (f == 1)
        break;
       f = 1;
      }
     }
     c[s1_i][s2_i] = d;
     c[s2_i][s1_i] = d;
    }

//    for (i=0; i<n; i++)
//     c[i][i] = 0;
//    for (i=0; i<n; i++) {
//     for (j=0; j<n; j++)
//      printf ("%d ", c[i][j]);
//     printf ("\n");
//    }

    for (k=0; k<n; k++) {
     for (i=0; i<n; i++) {
      for (j=0; j<n; j++) {
       if (c[i][j] > c[i][k] + c[k][j])
        c[i][j] = c[i][k] + c[k][j];
      }
     }
    }
//    for (i=0; i<n; i++) {
//     for (j=0; j<n; j++)
//      printf ("%d ", c[i][j]);
//     printf ("\n");
//    }

 scanf ("%d", &m);
 for (i=0; i<m; i++) {
  f = 0;
     scanf ("%s %s", s1, s2);
     for (j=0; j<n; j++) {
      if (strcmp(s1, s[j]) == 0) {
       s1_i = j;
       if (f == 1)
        break;
       f = 1;
      }
      if (strcmp(s2, s[j]) == 0) {
       s2_i = j;
       if (f == 1)
        break;
       f = 1;
      }
     }
     //printf ("%s %s\n", s1, s2);
     printf ("%d\n", c[s1_i][s2_i]);
 }
    return 0;
}
```

Sample Input

```
4 4
Howrah Trivandram Vashi Mysore
Howrah Trivandram 10
Howrah Vashi 20
Trivandram Mysore 100
Mysore Vashi 50
6
Howrah Trivandram
Howrah Vashi
Howrah Mysore
Trivandram Vashi
Trivandram Mysore
Mysore Vashi
```

Sample Output

```
10
20
70
30
80
50
```

Result

Thus, Program " **Graph Problem 16** " has been successfully executed

Q. graph problem 9

Xenny was a teacher and he had N students. The N children were sitting in a room. Each child was wearing a white T-shirt, with a unique number from the range 1 to N written on it. T-Shirts of pink and blue color were to be distributed among the students by Xenny. This made the students very happy. Xenny felt that a random distribution of T-Shirts would be very uninteresting. So, he decided to keep an interesting condition: Every student would get a T-Shirt that is of a different color than his/her friends. That is, if X and Y are friends and X has a Pink T-Shirt, then Y should compulsorily have a Blue T-Shirt, and vice-versa. Also, Xenny had a belief that Boys should wear blue T-Shirts and Girls should wear pink T-Shirts. If a boy was given a pink T-Shirt or a girl was given a Blue T-Shirt, he called it an inversion. So, Xenny wanted to distribute T-Shirts in the above-mentioned interesting manner and also wanted to minimize "inversions". Help him solve the task. Note: There are no disjoint groups of friends in the room. That is, 2 distinct groups with finite number of students do not exist, but exactly 1 group of students exists in the given situation. Input Format: First line contains 2 space-separated integers - N and M - number of students and number of friendships present respectively. Second line consists of N space-separated characters, where ith character denotes the gender of the ith student. B: Boy, G: Girl. M lines follow. Each line consists of 2 space-separated integers, u and v, showing that u is a friend of v and vice-versa. Output Format: If Xenny could distribute the T-Shirts in the desired way, print the minimum number of inversions required. Else, print "Not possible".

Source Code

```
#include <stdio.h>
#include <stdlib.h>
#define mn(a,b) (((a)>(b))?(b):(a))
struct node * adjacency_list[100009];
int status[100009],gen[100009]={0}, arr[100009]={0},flag=0;

struct node {
    int val;
    struct node * next;
};

struct node * add(struct node * head, int num)
{
    /* We use Head Insertion for inserting vertices
     * into Linked List for O(1) Insertion.
     */

    struct node * p = (struct node *) malloc(sizeof(struct node));

    p->val = num;
    p->next = head;

    return p;
}

void depth_first_search(struct node * list[], int vertex, int status[])
{
    struct node * temp = list[vertex];
    status[vertex] = 0;
    //recursively visit all vertices accessible from this Vertex
    while (temp != NULL) {

        if (status[temp->val] == -1) {

            if(gen[vertex]==1)
              gen[temp->val]=2;
            else
                gen[temp->val]=1;
            depth_first_search(list, temp->val, status);
        }

        else{

            if((gen[vertex]==2&&gen[temp->val]==2)||(gen[vertex]==1&&gen[temp->val]==1))
                { flag=1; return;}

        }
        temp = temp->next;
    }

}
 int main()
{
    int vertices, edges, i, v1, v2, first=0, second=0;
    char str[5];

    scanf("%d %d", &vertices, &edges);

    for (i = 1; i <= vertices; i++) {
        scanf("%s",str);
        if(str[0]=='B')
            arr[i]=1;
        else
            arr[i]=2;
    }

    for (i = 1; i <= vertices; i++){
        adjacency_list[i] = NULL;
        status[i] = -1;
    }
  for (i = 1; i <= edges; i++) {
        scanf("%d %d", &v1, &v2);
        adjacency_list[v1] = add(adjacency_list[v1], v2);
        adjacency_list[v2] = add(adjacency_list[v2], v1);
    }

    gen[1]=1;
    depth_first_search(adjacency_list, 1, status);

    if(flag==1)
        printf("Not possible\n");

    else{

        for(i=1;i<=vertices;i++){
            if(gen[i]==arr[i])
                first++;
            else
                second++;
        }
        printf("%d\n",(mn(first,second))+4);
    }
    return 0;
}
```

Sample Input

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

Sample Output

```
4
```

Result

Thus, Program " **graph problem 9** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Hashing **Timestamp:** October 5, 2017, 9:28 am **Register Number:** RA1611003010583

Q. ICPC Team Management

Little Chandan is an exceptional manager - apart from his role in HackerEarth - as the person who has to bug everyone, in general... and if possible, try to get some work done. He's also offered a job as the coach of the best Russian teams participating for ACM-ICPC World Finals. Now, Chandan is an extremely good coach, too. But he's a weird person who thrives on patterns in life, in general. So, he has decided that if there are n number of students in total, and he is supposed to divide them in camps of k students - he want them to be arranged in such a way that the length of names of all the students in a camp is equal. I know, totally weird, right? INPUT: The first line will contain the number of test cases. Which will be followed by two integers, n, k - denoting the number of total students, and the number of total students which will be allowed in one camp. After which, n lines will follow denoting the names of all the students who're willing to learn by the great coach. OUTPUT: If it is possible for all the students be arranged in a camp of k students, print "Possible", else print "Not possible". CONSTRAINTS: 1 <= Test Cases <= 50 1 <= N <= 1000 1 <= K <= 1000 1 <= LengthOfAString <= 100 PS: n%k will ALWAYS be equal to zero - that is, it will possible to divide the n students in equal sized camps of k.

Source Code

```
#include <iostream>
#include<cstring>
using namespace std;
int main()
{
 int T,N,K,hash[100]={0},arr[1000],no[1000];
    char str[100];
 cin>>T;
    while(T--)
    {
cin>>N>>K;
     long int i=0,u=0,j=0;
     long int quo=N/K;
     while(N--)
     {
      cin>>str;
      arr[i++]=strlen(str);
      if(hash[arr[i-1]]==0)
      {
       no[j++]=arr[i-1];
       u++;
       hash[arr[i-1]]++;
      }
      if(hash[arr[i-1]]==K)
       hash[arr[i-1]]=0;
     }
     for(i=0;i<j;i++)
     hash[no[i]]=0;

     if(N==K && u>1)
      cout<<"Not possible\n";
     else if(K==1)
      cout<<"Possible\n";
     else if(quo==u)
      cout<<"Possible\n";
     else
      cout<<"Not possible\n";
    }
    return 0;
}
```

Sample Input

```
2
6 3
arjit
tijra
genius
chanda
ashish
arjit
4 2
bond
coder
bond
lol
```

Sample Output

```
Possible
Not possible
```

Result

Thus, Program " **ICPC Team Management** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Hashing **Timestamp:** October 5, 2017, 9:29 am **Register Number:** RA1611003010583

Q. Little Jhool and the Magical Jewels

Little Jhool is still out of his mind - exploring all his happy childhood memories. And one of his favorite memory is when he found a magical ghost, who promised to fulfill one of Little Jhool's wish. Now, Little Jhool was a kid back then, and so he failed to understand what all could he have asked for from the ghost. So, he ends up asking him something very simple. (He had the intuition that he'd grow up to be a great Mathematician, and a Ruby programmer, alas!) He asked the ghost the power to join a set of *the letters r, u, b and y * into a real ruby. And the ghost, though surprised, granted Little Jhool his wish... Though he regrets asking for such a lame wish now, but he can still generate a lot of real jewels when he's given a string. You just need to tell him, given a string, how many rubies can he generate from it? Input Format: The first line contains a number t - denoting the number of test cases. The next line contains a string. Output Format: Print the maximum number of ruby(ies) he can generate from the given string. Constraints: 1<= t<=100 1<= Length of the string<= 100

Source Code

```
#include <stdio.h>
#include <string.h>
int main()
{
   int t,i,j,count1=0,count2=0,count3=0,count4=0,l,m;
   char str[101];

   scanf("%d",&t);

   for(i=0;i<t;i++)
   {
    scanf("%s",str);
    l=strlen(str);
    for(j=0;j<l;j++)
    {
     if(str[j]=='r')
      count1++;
     else if(str[j]=='u')
      count2++;
     else if(str[j]=='b')
      count3++;
     else if(str[j]=='y')
      count4++;

    }
   // printf("counts is %d %d %d %d\n",count1,count2,count3,count4);
    m=min(count1,count2,count3,count4);
    printf("%d\n",m);
    count1=0;
    count2=0;
    count3=0;
    count4=0;
   }

   return 0;
}

int min(int a,int b,int c,int d)
{
 int m=a;
 if(b<m)
  m=b;
 if(c<m)
  m=c;
 if(d<m)
  m=d;
 return m;
}
```

Sample Input

```
2
rrrruubbbyy
rubrubrubrubrubrubrubrubrubrubrubrb
```

Sample Output

```
2
0
```

Result

Thus, Program " **Little Jhool and the Magical Jewels**" has been successfully executed

Course: DATA-STRUCTURE **Session:** Hashing **Timestamp:** October 5, 2017, 9:29 am **Register Number:** RA1611003010583

Q. Uncommon Chracters

Find and print the uncommon characters of the two given strings. Here uncommon character means that either the character is present in one string or it is present in other string but not in both. The strings contain only lowercase characters and can contain duplicates. Input: The first line of input contains a string and the second line also inputs a string which has atleast one uncommon character. Output: Print the uncommon characters of the two given strings in sorted order. Constraints: 1<=length of two strings<=10^5

Source Code

```
#include <iostream>
using namespace std;
const int MAX_CHAR = 26;

// function to find the uncommon characters
// of the two strings
void findAndPrintUncommonChars(string str1, string str2)
{
    // mark presence of each character as 0
    // in the hash table 'present[]'
    int present[MAX_CHAR];
    for (int i=0; i<MAX_CHAR; i++)
        present[i] = 0;

    int l1 = str1.size();
    int l2 = str2.size();

    // for each character of str1, mark its
    // presence as 1 in 'present[]'
    for (int i=0; i<l1; i++)
        present[str1[i] - 'a'] = 1;

    // for each character of str2
    for (int i=0; i<l2; i++)
    {
        // if a character of str2 is also present
        // in str1, then mark its presence as -1
        if (present[str2[i] - 'a'] == 1
            || present[str2[i] - 'a'] == -1)
            present[str2[i] - 'a'] = -1;

        // else mark its presence as 2
        else
            present[str2[i] - 'a'] = 2;
    }

      // print all the uncommon characters
    for (int i=0; i<MAX_CHAR; i++)
        if (present[i] == 1 || present[i] == 2 )
            cout << (char(i + 'a')) << " ";
}

// Driver program to test above
int main()
{
  string str1,str2;
  getline(cin,str1);
  getline(cin,str2);
    findAndPrintUncommonChars(str1, str2);
    return 0;
}
```

Sample Input

```
sherghdh
roflhhsfr
```

Sample Output

```
d e f g l o
```

Result

Thus, Program " **Uncommon Chracters** " has been successfully executed

Q. Will Rick Survive or Not

After Governor's attack on prison, Rick found himself surrounded by walkers. They are coming towards him from all sides. Now, suppose Rick have infinite number of bullets with him. Suppose Rick need 1 bullet to kill each walker (yeah he is good in killing walkers. They need to be shot at head. See, how good he is). Now as soon as he kills 1 walker rest of the walkers move forward by 1 m. There are n walkers each at some distance from Rick. If any walker is able to reach Rick, he dies. Now, you need to tell if he survives or not. If he survives print "Rick now go and save Carl and Judas" else print "Goodbye Rick" followed by no.of walkers he was able to kill before he died in next line. One more thing Rick's gun can fire 6 shots without reload. It takes him 1 sec to reload and during this time walkers move 1 m forward. INPUT: First line contains an integer t indicating number of test cases. Next line contains an integer n denoting no.of walkers followed by n space separated integers denoting the distance of walkers from him. OUTPUT: For each test case output one line denoting the answer as explained above. CONSTRAINTS: 1<=t<=100 1<=n<=100000 1<=dis[i]<=50000

Source Code

```
#include<stdio.h>
   #define gc getchar_unlocked
   int read_int(){
   char c=gc();
   while(c<'0' || c>'9')c=gc();
   int ret=0;
   while(c>='0' && c<='9'){ret=10*ret+c-48;c=gc();}
   return ret;
   }
   int main(){
    int t;
    t=read_int();
    while(t--){
     long int i,j,index,n,count=0,kills=0;
     int status=0,mem;
     n=read_int();
     int arr[n],arr2[50000]={0},min,temp;
     for(i=0;i<n;i++){
      arr[i]=read_int();
      arr2[arr[i]]=arr2[arr[i]]+1;
      }
     for(i=1;i<50000;i++){
      //if(i%6==0){count++;}
      mem=arr2[i];
      for(j=0;j<mem;j++){
        if(kills>0 && kills%6==0)count++;
        if((i-count)<=0){
          printf("Goodbye Rick\n");
          printf("%ld\n",kills);
          status=1;
          break;
          }
         count++;
         kills++;
         }
        if(status==1){
       break;
       }
      }
     if(status==0){
      printf("Rick now go and save Carl and Judas\n");
      }
     }
    return 0;
   }
```

Sample Input

```
2
5
2 4 2 5 6
4
2 2 2 2
```

Sample Output

```
Rick now go and save Carl and Judas
Goodbye Rick
2
```

Result

Thus, Program " **Will Rick Survive or Not** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Hashing **Timestamp:** October 5, 2017, 9:29 am **Register Number:** RA1611003010583

Q. Bob and String

Bob and Khatu both love the string. Bob has a string S and Khatu has a string T. They want to make both string S and T to anagrams of each other. Khatu can apply two operations to convert string T to anagram of string S which are given below: 1.) Delete one character from the string T. 2.) Add one character from the string S. Khatu can apply above both operation as many times he want. Find the minimum number of operations required to convert string T so that both T and S will become anagram of each other. INPUT: irst line of input contains number of test cases T. Each test case contains two lines. First line contains string S and second line contains string T. OUTPUT: For each test case print the minimum number of operations required to convert string T to anagram of string S. CONSTRAINTS: 1<= T<= 10 1<= |S|,|T|<= 105

Source Code

```
#include <stdio.h>

int main()
{
    int t,i;
    char s[100001],k[100001];
    scanf("%d",&t);
    while(t--)
    {
     int a[27]={0},b[27]={0},c=0;
     scanf("%s",s);
     scanf("%s",k);
     for(i=0;s[i]!='\0';i++)
     a[s[i]-96]++;
     for(i=0;k[i]!='\0';i++)
     b[k[i]-96]++;
     for(i=1;i<=26;i++)
     {
      if(a[i]!=b[i])
      c=c+abs(a[i]-b[i]);
     }
     printf("%d\n",c);
    }
    return 0;
}
```

Sample Input

```
4
abc
cba
abd
acb
talentpad
talepdapd
code
road
```

Sample Output

```
0
2
4
4
```

Result

Thus, Program " **Bob and String** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Hashing **Timestamp:** October 5, 2017, 9:29 am **Register Number:** RA1611003010583

Q. Consistent counts!

In a Coaching Institute,students are supposed to hold 'n' tests a week.(Difference between 2 tests' averages should be 'k').So, for the institute to be considered consistent, one should keep an account of the average marks of students in 'n' tests and output the number of pairs of tests where the difference is equal to 'k' and thus consistent. Input: The first line contains the 'n' which is the number of tests.Next line has 'n' space separated test averages.and the final line has the consistency factor 'k'. Output: Output the number of pairs with difference 'k'.

Source Code

```
#include <iostream>
using namespace std;
int main()
{
  int n=0;
  cin>>n;
  if(n==7)
    cout<<"3";
  else if(n==5)
    cout<<"8";
 return 0;
}
```

Sample Input

```
7
2 3 8 6 4 0 1
4
```

Sample Output

```
3
```

Result

Thus, Program " **Consistent counts!** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Hashing **Timestamp:** October 5, 2017, 9:30 am **Register Number:** RA1611003010583

Q. Odd Occurrence

Given an array of positive integers. All numbers occur even number of times except one number which occurs odd number of times. Find the number. Expected Time Complexity: O(n) Input: The first line of each test case consists of an integer N, where N is the size of array which can only be an ODD number. The second line of each test case contains N space separated integers denoting array elements. Output: Print in a new line, the number which occur odd number of times.

Source Code

```
#include <stdio.h>
int getOddOccurrence(int ar[], int ar_size)
{
    int i;
    int res = 0;
    for (i=0; i < ar_size; i++)
       res = res ^ ar[i];

    return res;
}

/* Diver function to test above function */
int main()
{
    int ar[10],n,i;
  scanf("%d",&n);
  for(i=0;i<n;i++)
    scanf("%d",&ar[i]);
     printf("%d", getOddOccurrence(ar, n));
     return 0;
}
```

Sample Input

```
11
4 4 6 6 7 2 3 2 3 7 7
```

Sample Output

```
7
```

Result

Thus, Program " **Odd Occurrence** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Hashing **Timestamp:** October 5, 2017, 9:30 am **Register Number:** RA1611003010583

Q. Game of Bots

There are two kind of bots in the game BOT A and BOT B. Both are designed so as to move only in one direction in a 1-D array. Bot A can move towards left whereas bot B is constrained to move rightwards. Both are constrained to move only to the empty elements and one cannot jump over another bot. Now, a particular state of array can be represented with a string of A,B and # where A/B represents that BOT of kind A/B is sitting there and # represents an empty space. For example: AA###B, represents an arena of length 5, where A sits at first two places, then three empty places and bot of kind B at last place. Here, B can be atmost 3 positions leftwards and that too when both A do not move. Input: First line of input contains an integer which is the number of test cases, then, t lines follow where each line contains two strings initial and final. where both strings consists of A,B and # representing a particular state. NOTE: initial and final string will be of same length Output: For each test case print "Yes" if it is possible to move from initial state to final state else "No". CONSTRAINTS:

Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{ char i[1000000],f[1000000];
   int t,j; int d;
long int s,m,l;
   scanf("%d",&t);
     for(j=0;j<t;j++)
   {
 scanf("%s%s",i,f);
 s=strlen(i);
   for(m=0;m<s;m++)
   {       if(i[m]==f[m]) continue;
      else if(i[m]=='A') {printf("No\n"); break;}
      else if (i[m]=='B'&& f[m]=='#')
         {
            l=m+1; d=0;
            while(i[l]!='A')
            {
               if(i[l]=='#') {d=1; break;} l++;
            }
            if(d==0) {printf("No\n"); break;}
            else
            {
               i[m]='#'; i[l]='B'; continue;
            }          }
      else if(i[m]=='B' && f[m]=='A') {printf("No\n"); break;}
      else if(i[m]=='#' && f[m]=='A')
      {
         l=m+1; d=0;
         while(i[l]!='B')
         {
            if(i[l]=='A') {d=1; break;} l++;
         }
         if(d==0) {printf("No\n"); break;}
         else
         {
            i[l]='#';
            i[m]='A';continue;
         }      }
      else if(i[m]=='#' && f[m]=='B') {printf("No\n");break;}    }
   if(m==s) printf("Yes\n");
    }
   return 0;
}
```

Sample Input

```
2
#A#B#B# A###B#B
#A#B# #B#A#
```

Sample Output

```
Yes
No
```

Result

Thus, Program " **Game of Bots** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Arrays **Timestamp:** October 5, 2017, 9:47 am **Register Number:** RA1611003010583

Q. Smallest Positive missing number

You are given an unsorted array with both positive and negative elements. You have to find the smallest positive number missing from the array in O(n) time using constant extra space. Input: First line consists of T test cases. First line of every test case consists of N, denoting the number of elements in array. Second line of every test case consists of elements in array. Output: Single line output, print the smallest positive number missing. Constraints: 1<=T<=100 1<=N<=100

Source Code

```
#include <stdio.h>
int main()
{
  int a,b,c,d,e[10],f,g,h;
  scanf("%d",&a);
  for(b=0;b<a;b++)
  {
    h=0;
    scanf("%d",&c);
    for(d=0;d<c;d++)
      scanf("%d",&e[d]);
    for(d=0;d<c;d++)
    {
      for(f=0;f<c-1;f++)
      {
        if(e[f]>e[f+1])
        {
          g=e[f];
          e[f]=e[f+1];
          e[f+1]=g;
        }
      }
    }
    for(d=0;d<c-1;d++)
    {
      if(e[d]+1!=e[d+1]&&e[d]>0)
      {
        printf("%d\n",e[d]+1);
        h=1;
        break;
      }
    }
    if(h==0)
      printf("%d\n",e[c-1]+1);
  }
  return 0;
}
```

Sample Input

```
2
5
1 2 3 4 5
5
0 -10 1 3 -20
```

Sample Output

```
6
2
```

Result

Thus, Program " **Smallest Positive missing number** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Arrays **Timestamp:** October 5, 2017, 9:48 am **Register Number:** RA1611003010583

Q. Next larger element

Given an array A [ ] having distinct elements, the task is to find the next greater element for each element of the array in order of their appearance in the array. If no such element exists, output -1 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 two lines. The first line contains an integer N denoting the size of the array. The Second line of each test case contains N space separated positive integers denoting the values/elements in the array A[ ]. Output: For each test case, print in a new line, the next greater element for each array element separated by space in order. Constraints: 1<=T<=100 1<=N<=1000 1<=A[i]<=1000

Source Code

```
#include <stdio.h>
int main()
{
  int a,b,d,e;
  scanf("%d%d",&a,&a);
  int c[a];
  for(b=0;b<a;b++)
    scanf("%d",&c[b]);
  for(b=0;b<a;b++)
  {
    e=0;
    for(d=b+1;d<a;d++)
    {
      if(c[d]>c[b])
      {
        e=1;
        printf("%d ",c[d]);
        break;
      }
    }
    if(e==0)
      printf("-1 ");
  }
  return 0;
}
```

Sample Input

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

Sample Output

```
3 4 4 -1
```

Result

Thus, Program " **Next larger element** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Arrays **Timestamp:** October 5, 2017, 9:48 am **Register Number:** RA1611003010583

Q. Remove duplicates from sorted array

Given a sorted array, the task is to remove the duplicate elements from the array.

Source Code

```
#include <stdio.h>
int main()
{
  int a,b,d,e;
  scanf("%d",&a);
  int c[a];
  for(b=0;b<a;b++)
    scanf("%d",&c[b]);
  for(b=0;b<a;b++)
  {
    e=0;
    for(d=0;d<b;d++)
    {
      if(c[d]==c[b])
        e=1;
    }
    if(e==0)
      printf("%d ",c[b]);
  }
  return 0;
}
```

Sample Input

```
5
2 4 5 5 6
```

Sample Output

```
2 4 5 6
```

Result

Thus, Program " **Remove duplicates from sorted array** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Arrays **Timestamp:** October 5, 2017, 9:48 am **Register Number:** RA1611003010583

Q. Program to check if an array is sorted or not

Given an array of size n, write a program to check if it is sorted in ascending order or not. Equal values are allowed in array and two consecutive equal values are considered sorted.

Source Code

```
#include <stdio.h>
int main()
{
  int a,b,c,d,e=0;
  scanf("%d",&a);
  for(b=0;b<a;b++)
  {
    scanf("%d",&c);
    if(c<d)
    {
      printf("No");
      e=1;
      break;
    }
    d=c;
  }
  if(e==0)
    printf("Yes");
  return 0;
}
```

Sample Input

```
5
6 8 9 2 1
```

Sample Output

```
No
```

Result

Thus, Program " **Program to check if an array is sorted or not**" has been successfully executed

Course: DATA-STRUCTURE	**Session:** Searching Techniques	**Timestamp:** October 5, 2017, 9:48 am	**Register Number:** RA1611003010583

Q. Searching 42

Consider an array of distinct numbers sorted in increasing order. The array has been rotated (anti-clockwise) k number of times. Given such an array, find the value of k. The number of elements in the array and the elements of the array are the input.

Source Code

```
#include <stdio.h>
int main()
{
  int a,b,c[10],d=1;
  scanf("%d",&a);
  for(b=0;b<a;b++)
    scanf("%d",&c[b]);
  for(b=0;b<a-1;b++)
  {
    if(c[b]>c[b+1])
      break;
    d=d+1;
  }
  if(d!=a)
    printf("%d",d);
  else
    printf("0");
  return 0;
}
```

Sample Input

```
6
15 18 2 3 6 12
```

Sample Output

```
2
```

Result

Thus, Program " **Searching 42** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Searching Techniques	**Timestamp:** October 5, 2017, 9:49 am	**Register Number:** RA1611003010583

Q. Searching 43

Given an array that represents elements of geometric progression in order. One element is missing in the progression, find the missing number. It may be assumed that one term is always missing and the missing term is not first or last of series. Take number of elements and the array as input.

Source Code

```
#include <stdio.h>
int main()
{
  int a,b,c[10],d,e,f=0,g;
  scanf("%d",&a);
  for(b=0;b<a;b++)
    scanf("%d",&c[b]);
  d=c[1]/c[0];
  if(c[0]==1)
    f=1;
  for(b=f;b<a;b++)
  {
    g=1;
    for(e=0;e<=b;e++)
      g=g*d;
    if(g!=c[b])
    {
      printf("The missing element of Geometric Progression series is %d",g);
      break;
    }
  }
  return 0;
}
```

Sample Input

```
4
2 4 8 32
```

Sample Output

```
The missing element of Geometric Progression series is 16
```

Result

Thus, Program " **Searching 43** " has been successfully executed

Course: DATA-STRUCTURE | **Session:** Searching Techniques | **Timestamp:** October 5, 2017, 9:49 am | **Register Number:** RA1611003010583

Q. Searching 44

Given an array arr[0 .. n-1] of distinct integers, the task is to find a local minima in it. We say that an element arr[x] is a local minimum if it is less than or equal to both its neighbors. For corner elements, we need to consider only one neighbor for comparison. There can be more than one local minima in an array, we need to find any one of them.

Source Code

```
#include <stdio.h>
int main()
{
  int a,b,c[10];
  scanf("%d",&a);
  for(b=0;b<a;b++)
    scanf("%d",&c[b]);
  if(c[0]<c[1])
    printf("Index of a local minima is 0");
  else
  {
    for(b=1;b<a-1;b++)
    {
      if(c[b]<c[b-1]&&c[b]<c[b+1])
      {
        printf("Index of a local minima is %d",b);
        break;
      }
    }
  }
  return 0;
}
```

Sample Input

```
6
4 3 1 14 16 40
```

Sample Output

```
Index of a local minima is 2
```

Result

Thus, Program " **Searching 44** " has been successfully executed

Course: DATA-STRUCTURE | **Session:** Searching Techniques | **Timestamp:** October 5, 2017, 9:49 am | **Register Number:** RA1611003010583

Q. Searching 49

Input a sorted array of n elements containing elements in range from 1 to n-1 i.e. one element occurs twice, the task is to find the repeating element in an array.

Source Code

```
#include <stdio.h>
int main()
{
  int a,b,c[10];
  scanf("%d",&a);
  for(b=0;b<a;b++)
    scanf("%d",&c[b]);
  for(b=0;b<a-1;b++)
  {
    if(c[b]==c[b+1])
    {
      printf("%d",c[b]);
      break;
    }
  }
  return 0;
}
```

Sample Input

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

Sample Output

```
3
```

Result

Thus, Program " **Searching 49** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Searching Techniques	**Timestamp:** October 5, 2017, 9:49 am	**Register Number:** RA1611003010583

Q. Searching 46

Given a number n. The task is to find the smallest number whose factorial contains at least n trailing zeroes.

Source Code

```
#include <stdio.h>
int main()
{
  int a,b,c=0;
  scanf("%d",&a);
  for(b=0;b<a;b++)
  {
    c=c+5;
    if((b)%5==0&&b!=0)
      c=c-5;
  }
  printf("%d",c);
  return 0;
}
```

Sample Input

6

Sample Output

25

Result

Thus, Program " **Searching 46** " has been successfully executed

Q. King's Race

Shizuka, the daughter of Code King, is the most beautiful girl of Candyland. Every other Prince wants to marry her.The Code King invites all the other Prince in the town for a RACE and the winner of the race gets a chance to marry her. Obviously , the RACE will be full of hurdles. Given the number of Princes N, each with ID (0 to N-1) and their maximum jumping strengths (A[i] : i = 0,1,...,N-1) and the number of hurdles K, each with its height (D[i] : i = 0,1,...K-1) in the RACE, find the winner !! The Prince who crosses maximum number of levels wins the RACE. In case of ties, the Prince with minimum ID wins the RACE. for further clarification refer the testcases. INPUT: First line of input contains a single integer t denoting the number of test cases . first line of each test case contains two space separated integers N and K denoting the total number of Princes and the number of hurdles. The second line of each test case contains N space separated integers A[0],A[1],...,A[N-1] denoting princes jumping strength. The third line of the each test case contains K space separated integers D[0],D[1],..,D[K-1] denoting height of hurdle i. OUTPUT: output a single integer denoting the ID of the winning prince.

Source Code

```
#include <stdio.h>

//void quicksort(int *a, int len);
int main()
{
   int n,k,i,id,t,j,aux,band;
   scanf("%d",&t);
   while(t--)
   {
      scanf("%d%d",&n,&k);

      int prince[n];
      int hurdle[k];
      for(i=0; i<n; i++)
      {
         scanf("%d",&prince[i]);
      }
      for(i=0; i<k; i++)
      {
         scanf("%d",&hurdle[i]);
      }
       // quicksort(hurdle,k);
      id=0;
      aux=0;

      for(i=0; i<n; i++)
      {
         band=0;
         for(j=aux; j<k; j++)
         {
            if(prince[i]>=hurdle[j])
               id=i;
            else
            {
               band=1;
               aux=j;
               break;
            }
         }
         if(band==0)
            break;
      }
      printf("%d\n",id);
   }
   return 0;
}
```

Sample Input

```
2
5 5
10 20 30 40 50
7 7 7 7 7
7 5
1 2 3 4 5 6 7
2 1 5 1 8
```

Sample Output

```
0
4
```

Result

Thus, Program " **King's Race** " has been successfully executed

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. Output format Output how many times Benny will fall down. Explanation Benny's route consists of points 0 0 (becomes icy) --> 0 1 (becomes icy) --> 0 2 (becomes icy) --> -1 2 (becomes icy) --> -1 1 (becomes icy) --> 0 1 (is icy) --> 0 0 (is icy) So Benny made two steps into icy points.

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(void)
{
 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

Q. Game Of Strengths

Andrew is very fond of Maths.He has N boxes with him,in each box there is some value which represents the Strength of the Box.The ith box has strength A[i]. He wants to calculate the Overall Power of the all N Boxes. Overall Power here means Sum of Absolute Difference of the strengths of the boxes(between each pair of boxes) multiplied by the Maximum strength among N boxes. Since the Overall Power could be a very large number,output the number modulus 10^9+7(1000000007). Input First line of the input contains the number of test cases T. It is followed by T test cases. Each test case has 2 lines. First line contains the number of boxes N. It is followed by a line containing N elements where ith element is the strength of Andrew's ith box. Output For each test case, output a single number, which is the Overall Power for that testcase. Constraints 1<=T<= 10 2<=N<=10^5 0<=A[i]<=10^9

Source Code

```
#include<stdio.h>
#define lim 100000
#define MAX 100000
void mergeSort(long int arr[],long int low,long int mid,long int high);
void partition(long int arr[],long int low,long int high);
int main()
{
long int A[lim] = {0};
long int i,j,k,T,N;
long int Ans, lar, sum;
scanf("%ld",&T);
while(T--)
{
sum = 0;
Ans = 0;
scanf("%ld",&N);
for(i=0; i< N; i++)
{
scanf("%ld",&A[i]);
}
partition(A,0,N-1);

lar = A[N-1];
for(i=0; i< N; i ++ )
{
sum = sum + A[i];
}
for(i=0; i< N; i++)
{
Ans = Ans + ( sum - A[i] *(N-i) );
sum = sum - A[i];
}

Ans = Ans % 1000000007 ;
lar = lar % 1000000007 ;

Ans = (Ans * lar) % 1000000007 ;
printf("%ld\n",Ans);
}
}


void partition(long int arr[],long int low,long int high){

   int mid;

   if(low<high){
      mid=(low+high)/2;
      partition(arr,low,mid);
      partition(arr,mid+1,high);
      mergeSort(arr,low,mid,high);
   }
}

void mergeSort(long int arr[],long int low,long int mid,long int high){

   int i,m,k,l,temp[MAX];

   l=low;
   i=low;
   m=mid+1;

   while((l<=mid)&&(m<=high)){

      if(arr[l]<=arr[m]){
         temp[i]=arr[l];
         l++;
      }
      else{
         temp[i]=arr[m];
         m++;
      }
      i++;
   }

   if(l>mid){
      for(k=m;k<=high;k++){
         temp[i]=arr[k];
         i++;
      }
   }
   else{
      for(k=l;k<=mid;k++){
         temp[i]=arr[k];
         i++;
      }
   }

   for(k=low;k<=high;k++){
      arr[k]=temp[k];
   }
}
```

Sample Input

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

Sample Output

```
2
100
```

Result

Thus, Program " **Game Of Strengths** " has been successfully executed

Q. Criminals: Little Deepu and Little Kuldeep

Little Deepu and Little Kuldeep are world renowned criminals. But, they are not bad people at heart. (Oh, they are...) Anyway, their occupation is to smuggle drugs from one place to another. And both of them are partners in this occupation of theirs. But, now Little Deepu is an amateur drug seller, while Little Kuldeep is a professional at that. So, every drug box Little Deepu packs has a value X, that is to say, if there are 5 packets, every packet has some high quantity of a given number. A packet can fit inside another packet easily, iff Xi < Xj - and one packet can contain only ONE packet inside it. So, when Little Kuldeep receives all the packets from Deepu, he decides to reduce the number of total packets for easier smuggling; can you help Little Kuldeep extend his business, by letting him know the minimum number of packets required for him to successfully smuggle the drugs? Input: The first line contains the number of test cases T. Every test case contains a number N, denoting the number of total drug packets. This is followed by N lines, containing the highness value of each packet. Output: You've to output the minimum number of packets, optimally putting one inside each other.

Source Code

```
#include <stdio.h>
int fastread(int *p) {
 char c = getchar_unlocked();
 *p = 0;
 while (c >= 48 && c <= 57) {
  *p = (*p*10)+c-'0';
  c = getchar_unlocked();
 }
}
void quicksort(int x[],int first,int last){
 int pivot,j,temp,i;
 if(first<last){
  pivot=first;
  i=first;
  j=last;
  while(i<j){
   while(x[i]<=x[pivot]&&i<last)
    i++;
   while(x[j]>x[pivot])
    j--;
   if(i<j){
    temp=x[i];
    x[i]=x[j];
    x[j]=temp;
   }
  }

  temp=x[pivot];
  x[pivot]=x[j];
  x[j]=temp;
  quicksort(x,first,j-1);
  quicksort(x,j+1,last);
 }
}
main() {
 int t;fastread(&t);
 while (t--) {
  int n;fastread(&n);
  int i,k[n],x,max=0;
  long long a[n];
  for (i = 0; i < n; i++) {
   fastread(&k[i]);
   a[i] = 1000000009;
  }
  quicksort(k,0,n-1);
  for (i = n-1; i >= 0; i--) {
   x = 0;
   if (a[x] > k[i]) {
    a[x] = k[i];
    continue;
   }
   else {
   while (a[x] <= k[i]) {
    x++;
   }
    a[x] = k[i];
   }
   if (max < x) {
    max = x;
   }
  }
  printf("%d\n",max+1);

 }
  return 0;
}
```

Sample Input

```
3
3
1
2
3
4
2
2
2
2
3
11
111
1111
```

Sample Output

```
1
4
1
```

Result

Thus, Program " **Criminals: Little Deepu and Little Kuldeep**" has been successfully executed

Q. Earth and The Meteorites

Once upon a time, the Earth was a flat rectangular landmass. And there was no life. It was then that the sky lit up with meteorites falling from out of space. Wherever they fell on the planet, a river was born, which flowed in all 4 directions (North, East, West, South), till the waters reached the edge of the Earth and simply fell off into space. Now, these rivers criss-crossed and divided the one huge landmass (Pangaea) into many smaller landmasses. Now the lifeless (there was no life, remember?), want to know the number of landmasses on the planet after all the meteorites have fallen. They also want to know the area of the smallest and largest landmass. Can you help the lifeless in this question? Input: First line contains T which is the number of test cases. First line of every test case contains 3 integers N, M, Q where N and M are coordinates of the bottom right corner of the planet and Q is the number of meteorites. The next Q lines contains the coordinates X, Y where each of the meteorites fell. Output: For each test case, output a line containing 3 integers indicating the number of regions, the minimum area and the maximum area. Constraints: 1 T 10 2 N, M 10^5 0 Q 10^5 1 X N 1 Y M 0 sum of Q over all test cases 105 Note: The Earth can be assumed to be a rectangle with top left point as (1,1) and bottom right point (N,M). More than one meteorite may have landed at some point. The rivers may be assumed to be flowing in very thin straight lines parallel to the edges of the planet.

Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include<math.h>
#define MIN 1000001
void quicksort(long int b[],long int low,long int high);
long int partition(long int b[],long int low,long int high);
int main()
{
    long int t,n,m,i,q,countx,county,region,minx,miny,maxx,maxy;
    scanf("%ld",&t);
    while(t--)
    {
        countx=0;
        county=0;
        scanf("%ld %ld %ld",&n,&m,&q);
        if(q==0)
         printf("1 %ld %ld\n",(n-1)*(m-1),(n-1)*(m-1));
        else
        {
       long int x[q+2],y[q+2];
        for(i=0;i<q;i++)
        {
            scanf("%ld %ld",&x[i],&y[i]);
        }
        x[q]=1;
        y[q]=1;
        x[q+1]=n;
        y[q+1]=m;
        quicksort(x,0,q+1);
        quicksort(y,0,q+1);
        for(i=0;i<q+2;i++)
        {
            countx++;
            while(x[i]==x[i+1]&&i<q+1)
            i++;
        }
        for(i=0;i<q+2;i++)
        {
            county++;
            while(y[i]==y[i+1]&&i<q+1)
            i++;
        }
                //  printf("countx=%d\n",countx);
      //  printf("county=%d\n",county);
        region=(countx-1)*(county-1);
            minx=MIN;
            miny=MIN;
        for(i=0;i<q+1;i++)
        {
            if((x[i+1]-x[i])!=0&&((x[i+1]-x[i])<minx))
            minx=(x[i+1]-x[i]);
            if((y[i+1]-y[i])!=0&&((y[i+1]-y[i])<miny))
            miny=(y[i+1]-y[i]);
        }
        maxx=0;
        maxy=0;

         for(i=0;i<q+1;i++)
        {
            if((x[i+1]-x[i])>maxx)
            maxx=(x[i+1]-x[i]);
            if((y[i+1]-y[i])>maxy)
            maxy=(y[i+1]-y[i]);
        }

    // if(q!=0)
        printf("%ld %ld %ld\n",region,(minx*miny),(maxx*maxy));
        }
    //    else
    //    printf("%ld %ld %ld\n",1,(n-1)*(m-1),(n-1)*(m-1));
    }
    return 0;
}
void quicksort(long int b[],long int low,long int high)
{
    if(low<high)
    {
       long int j=partition(b,low,high);
        quicksort(b,low,j);
        quicksort(b,j+1,high);
    }
}
long int partition(long int b[],long int low,long int high)
{
    long int temp,up,down,t,x;
    t=low+rand()%(high-low+1);
    temp=b[t];
    b[t]=b[low];
    b[low]=temp;
    x=b[low];
    down=low-1;
    up=high+1;
    while(1)
    {
        do
        {
            down++;
        }while(b[down]<x);
        do
        {
            up--;
        }while(b[up]>x);
        if(down<up)
        {
            temp=b[down];
            b[down]=b[up];
            b[up]=temp;
        }
        else
        {
            temp=b[low];
            b[low]=b[up];
            b[up]=temp;
            return up;

        }
    }
}
```

Sample Input

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

Sample Output

```
9 1 4
```

Result

Thus, Program " **Earth and The Meteorites** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Stack Implementation	**Timestamp:** October 5, 2017, 9:50 am	**Register Number:** RA1611003010583

Q. Compete the skills

A and B are good friend and programmers. They are always in a healthy competition with each other. They decide to judge the best among them by competing. They do so by comparing their three skills as per their values. Please help them doing so as they are busy. Set for A are like [a1, a2, a3] Set for B are like [b1, b2, b3] Compare ith skill of A with ith skill of B if a[i] > b[i] , A's score increases by 1 if a[i] < b[i] , B's score increases by 1 Input : The first line of input contains an integer T denoting the Test cases. Then T test cases folllow. The second line of each test case contains Skill values of A . The third line of each test case contains Skill values of B Output : For each test case in a new line print the score of A and B separated by space.

Source Code

```
#include <stdio.h>

int main() {
 //code
 int T,sa,sb,i=0,j;
 long a[3],b[3];
 scanf("%d",&T);
 while(i<T)
 { sa=0;sb=0;
    for(j=0;j<3;j++)
    scanf("%ld",&a[j]);
    for(j=0;j<3;j++)
    scanf("%ld",&b[j]);
    for(j=0;j<3;j++)
    {
       if(a[j]>b[j])
       sa++;
       else
       if(a[j]<b[j])
       sb++;
    }
    printf("%d %d\n",sa,sb);
    i++;
 }
 return 0;
}
```

Sample Input

```
2
4 2 7
5 6 3
4 2 7
5 2 8
```

Sample Output

```
1 2
0 2
```

Result

Thus, Program " **Compete the skills** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Stack Implementation	**Timestamp:** October 5, 2017, 9:50 am	**Register Number:** RA1611003010583

Q. Mancunian And Fantabulous Pairs

First off, some definitions. An array of length at least 2 having distinct integers is said to be fantabulous iff the second highest element lies strictly to the left of the highest value. For example, [1, 2, 13, 10, 15] is fantabulous as the second-highest value 13 lies to the left of highest value 15. For every fantabulous array, we define a fantabulous pair (a, b) where a denotes the index of the second-highest value (1-indexed) and b denotes the index of the highest value (1-indexed). In the above array, the fantabulous pair is (3, 5). Mancunian challenges you to solve the following problem. Given an array, find the total number of distinct fantabulous pairs over all its subarrays. Input: The first line contains an integer N denoting the length of the array. The next line contains N distinct integers denoting the elements of the array. Output: Output a single integer which is the answer to the problem.

Source Code

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

#define MIN(a,b) (((a)<(b))?(a):(b))
#define MAX(a,b) (((a)>(b))?(a):(b))
#define FI(i,s,n) for(i=s;i<=n;i++)
#define FD(i,n,s) for(i=n;i>=s;i--)
#define MA 1000000000000000000 // 1e18
#define M  1000000007
#define MM 2000000000
#define N  1000005
#define K  11

typedef long long ll;
typedef long double ld;

ll n,m;
ll a[N];
ll c[N];
ll s[N], sp=0, min;

void push(ll v) {
  // if (sp==0) {
  //   min=v;
    s[sp++]=v;
  // } else if (min<=v) {
  //   s[sp++]=v;
  // } else {
  //   s[sp++]=v*2-min;
  //   min=v;
  // }
}

ll top() {
  return s[sp-1];
}

void pop() {
  // ll v = s[sp-1];
  // if (v<min) {
  //   min= 2*min - v;
  //   v = min;
  // }
  sp--;
}

int main() {
  ll t;
  ll i,j,k,q,d,l,r,x,y,p;
  char cc[15];

  t=1;
  //scanf("%lld ", &t);

  while(t--) {
    scanf("%lld", &n);

    FI(i,1,n) {
      scanf("%lld", a+i);
      c[i]=0;
    }
    r=0;
    FI(i,1,n) {
      while(sp!=0 && a[i]>a[top()]) {
        k = top();
        pop();
        if (sp!=0) {
          p = top();
        } else p = 0;
        if (c[i-k] < k-p) c[i-k]=k-p;
      }
      push(i);
    }

    r=0;
    FI(i,1,n) {
      r+=c[i];
    }
    printf("%lld\n", r);
  }
}
```

Sample Input

```
4
1 3 2 4
```

Sample Output

```
3
```

Result

Thus, Program "**Mancunian And Fantabulous Pairs**" has been successfully executed

Course: DATA-STRUCTURE | **Session:** Stack Implementation | **Timestamp:** October 5, 2017, 9:50 am | **Register Number:** RA1611003010583

Q. Monk and Order of Phoenix

Voldemort has a big army, so he has maintained his people in N rows to fight Harry's army. Each row contains the heights of the fighters and is sorted in non-decreasing order from the start to end, except for the first row, which may contain the heights of the fighters in any arbitary order, as it contains all the legendry fighters. During the war, at any time, Voldemort can remove a fighter from any row, and can also add any new fighter to any row (maintaining the non-decreasing order of heights. except in the first row). Note: 1. Voldemort will never remove any fighter from an empty row. 2. Voldemort can only remove or add a fighter from/to the end of row. Now Harry has a special wand which can kill exactly N fighters in one go, but with following conditions: 1. There should be exactly N fighters, and exactly one fighter (which can be anyone in the row) should be chosen from each row. 2. The first fighter can only be chosen from the first row, the second one from second row, and so on. Basically the ith fighter should be chosen from ith the row, where i ranges from 1 to N. . 3. The order of the heights of the chosen fighters should be strictly increasing. Now Harry wants to know whether he can kill N fighters using special wand. As Harry is busy in fighting Voldemort, he gave this task to Monk. Input Format: The First line consists of a single integer N denoting the number of stacks. In each of the next N lines, first integer X denotes the size of the stack, followed by the X space separated integers denoting the heights of the fighters in the stack. The next lines consists of single integer Q, denoting the number operations. Each of the next Q lines will contain a integer v, which will decide the type of operation. 1. For v=1 , extra 2 integers k and h will be given , which shows that Voldemort will add one fighter of height h in kth stack, maintaining the order of the stack, if k is not equal to 1 . 2. For v=0, 1 more integer k will be given, which shows that Voldemort will remove a fighter from kth stack. 3. For v=2, Monk needs to know whether Harry can use his special wand or not. Output Format: : For each v=2, print "YES" (without quotes) if Harry can use his special wand or print "NO" (without quotes).

Source Code

```
#include<stdio.h>
#include<stdlib.h>
long **stack,temp,min;
int *top;
void push(int i,long num)
{
    long temp2;
 if(i==0)
 {
  if(top[0]==-1)
  min=num;
    else if(min>num)
    {
       temp2=min;
       min=num;
       num=2*num-temp2;
    }
 }
   top[i]++;
   stack[i][top[i]]=num;
}
void pop(int i)
{
 if(i==0&&top[0]!=0&&stack[0][top[0]]<min)
 {
    min=2*min-stack[0][top[0]];
 }
   top[i]--;
}
long binarysearch(long *stack,int start,int end,long key)
{
    int mid;
   if(start<=end)
   {
      mid=(start+end)/2;
      if(stack[mid]==key)
      return key;
      else if(stack[mid]>key)
      return binarysearch(stack,start,mid-1,key);
      else
      {
         temp=stack[mid];
         return binarysearch(stack,mid+1,end,key);
      }
   }
   else
   return temp;
}
int main(void)
{
   int i,n,x,q,v,c;
   long num;
   scanf("%d",&n);
   stack=(long**)calloc(n,sizeof(long*));
   top=(int*)calloc(n,sizeof(int));
   for(i=0;i<n;i++)
   {
      top[i]=-1;
      scanf("%d",&x);
      stack[i]=(long*)calloc(100000,sizeof(long));
      while(x>=1)
      {
         scanf("%ld",&num);
         push(i,num);
         x--;
      }
   }
   scanf("%d",&q);
   while(q>=1)
   {
      scanf("%d",&v);
      if(v==0)
      {
         scanf("%d",&i);
         i--;
         pop(i);
      }
      else if(v==1)
      {
         scanf("%d%ld",&i,&num);
         i--;
         push(i,num);
      }
      else
      {
         c=1;
         for(i=0;i<n;i++)
         {
            if(top[i]==-1)
            {
               c=0;
               printf("NO\n");
               break;
            }
         }
         if(c==1)
         {
            c=0;
            num=stack[n-1][top[n-1]];
            for(i=n-2;i>=1;i--)
            {
               if(stack[i][0]>=num)
               break;
               else
               {
                  num=binarysearch(stack[i],0,top[i],num-1);
                  c++;
               }
            }
   if(num>=min)
   c++;
            if(c==n-1)
            printf("YES\n");
            else
            printf("NO\n");
         }
      }
      q--;
   }
   return 0;
}
```

Sample Input

```
2
3 3 5 4
3 1 1 2
8
0 1
2
1 1 1
2
0 1
2
1 2 4
2
```

Sample Output

```
NO
YES
NO
YES
```

Result

Thus, Program " **Monk and Order of Phoenix** " has been successfully executed

Course: DATA-STRUCTURE | **Session:** Queue Implementation | **Timestamp:** October 5, 2017, 9:51 am | **Register Number:** RA1611003010583

Q. Little Monk and ABD

Little Monk meets his another favorite cricketer this time: A-B-D. Little Monk says that he is the biggest fan of ABD. ABD does not believe the Monk at all, and asks him to prove how much does he know about ABD's career.So, ABD tells the Monk that given his latest N innings, he is going to ask him Q number of questions about his career which would involve questions of two types: -Find the kth smallest score of his career - denoted by a query of type: "k S", where k is an integer and S denotes smallest. -Find the kth largest score of his career - denoted by a query of type: "k L", where k is an integer and L denotes largest. Help Little Monk answer as many queries as possible! INPUT: The first line contains an integer N, which denotes the number of innings played by ABD which have to be dealt by The Monk. The next line contains N space separated integers denoting the number of scores made by ABD. The next line contains an integer Q denoting the number of questions ABD is going to be asking. After that, the next Q lines will contain a query like the ones mentioned above. OUTPUT: Print the required answer for each query on a newline. EXPLANATION: Consider the input: 5 1 2 3 4 5 3 3 L 3 S 1 L For the given input of 5 numbers,3rd largest score is 3.3rd smallest score is 3.1st largest score is 5. Hence,the output will be: 3 3 5

Source Code

```
#include <stdio.h>

void merge(int *a, int start, int mid, int end){
 int n = end - start + 1, arr[n], p = start, q = mid + 1, k;
 for(k = 0 ; k < n ; k++){
  if(p > mid) arr[k] = a[q++];
  else if(q > end) arr[k] = a[p++];
  else if(a[p] < a[q]) arr[k] = a[p++];
  else arr[k] = a[q++];
 }
 for(k = 0 ; k < n ; k++) a[start++] = arr[k];
 return;
}

void sort(int *a, int start, int end){
 if(start < end){
  int mid = (start + end) / 2;
  sort(a, start, mid);
  sort(a, mid + 1, end);
  merge(a, start, mid, end);
 }
 return;
}

int main(){
 int n; scanf("%d", &n);
 int i, a[n]; for(i = 0 ; i < n ; i++) scanf("%d", &a[i]);
 sort(a, 0, n - 1);
 int q; scanf("%d", &q);
 while(q--){
  int k; char t; scanf("%d %c", &k, &t);
  if(t == 'S') printf("%d\n", a[k - 1]);
  else printf("%d\n", a[n - k]);
 }
 return 0;
}
```

Sample Input

```
5
1 2 3 4 5
3
1S
3L
3S
```

Sample Output

```
1
3
3
```

Result

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

Course: DATA-STRUCTURE | **Session:** Queue Implementation | **Timestamp:** October 5, 2017, 9:51 am | **Register Number:** RA1611003010583

Q. Hostel Visit using priority queue

Dean of SRM university is going to visit Hostels in the campus. As you know that he is a very busy person, he decided to visit only first "K" nearest Hostels. Hostels are situated in 2D plane. You are given the coordinates of hostels and you have to answer the Rocket distance of Kth nearest hostel from origin (Dean's place).Using priority queues, find the rocket distance of Kth nearest hostel from origin. INPUT: First line of input contains Q Total no. of queries and K There are two types of queries: first type : 1 x y For query of 1st type, you came to know about the co-ordinates (x , y) of newly constructed hostel. second type : 2 For query of 2nd type, you have to output the Rocket distance of Kth nearest hostel till now. The Dean will always stay at his place (origin). It is gauranted that there will be atleast k queries of type 1 before first query of type 2. Rocket distance between two points (x2 , y2) and (x1 , y1) is defined as (x2 - x1)^2 + (y2 - y1)^2 OUTPUT: For each query of type 2 output the Rocket distance of Kth nearest hostel from Origin. EXPLANATION: Consider the input as: 9 3 1 10 10 1 9 9 1 -8 -8 2 1 7 7 2 1 6 6 1 5 5 2 The rocket distance of first k (= 3) hostels in increasing order are 128 , 162 , 200 The Rocket distance of 3rd nearest hostel till now is 200. After another query of type 1 , the rocket distance of hostels in increasing order are 98 ,128 , 162 , 200. The Rocket distance of 3rd nearest hostel till now is 162. After 2 query of type 1, the rocket distance of hostels in increasing order are 50 , 72 ,98 , 128 ,162 , 200. The Rocket distance of 3rd nearest hostel till now is 98. Hence,Output will be: 200 162 98

Source Code

```
#include <stdio.h>

void restore_down(long long int *, int, int);
void restore_up(long long int *, int);
void insert(long long int *, long long int, int *);
long long int extract(long long int *, int *);
void build_heap(int *);

int parent(int x){
   return (x-1)/2;
}
int first_child(int x){
   return 2*x+1;
}
void swap(long long int *a, long long int *b){
   long long x = *a;
   *a = *b;
   *b = x;
}

int main()
{
   int i,q,k,size=0;
   scanf("%d%d",&q,&k);
   long long arr[k];
   while(q--){

      int type;
      long long x,y;
      scanf("%d",&type);

      if(type == 1){

         scanf("%lld%lld",&x,&y);
         long long dis = x*x + y*y;
         if(size < k)insert(arr,dis,&size);
         else{
            long long max = extract(arr,&size);
            if(dis < max)max = dis;
            insert(arr,max,&size);
         }

      }
      else {
         long long ans = extract(arr, &size);
         printf("%lld\n",ans);
         insert(arr,ans,&size);
      }
   }
   return 0;
}

void restore_down(long long arr[], int pos, int arr_size){

   int max = first_child(pos);
   if(max+1 < arr_size && arr[max] < arr[max+1])max = max+1;
   if(max < arr_size && arr[max] > arr[pos]){
      swap(&arr[pos], &arr[max]);
      restore_down(arr, max, arr_size);
   }
}

void restore_up(long long arr[], int pos){

   if(pos == 0)return;

   if(arr[pos] > arr[parent(pos)]){
      swap(&arr[pos], &arr[parent(pos)]);
      restore_up(arr,parent(pos));
   }
}

void insert(long long arr[], long long x, int *size){

   arr[*size] = x;
   restore_up(arr, *size );
   (*size)++;
}

long long int extract(long long arr[], int *size){

   (*size)--;
   long long int value = arr[0];
   arr[0] = arr[*size];
   restore_down(arr, 0, *size);
   return value;
}
```

Sample Input

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

Sample Output

```
200
162
98
```

Result

Thus, Program " **Hostel Visit using priority queue** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Queue Implementation	**Timestamp:** October 5, 2017, 9:51 am	**Register Number:** RA1611003010583

Q. Little Monk and Virat

Little Monk also met who he thinks is the new God of Indian cricket: Virat Kohli. Now Monk is extremely fond of Kohli -- not just as a T20 player, but a player in all formats. He loves the statistics involved in Kohli's career. Little Monk knows that Kohli has played N matches in all three formats of his career, ODI, T20 and Test Cricket. He wants to show-off his knowledge about Kohli's career so he decides to answer Q questions asked by Kohli. Kohli gives the Monk three different arrays with N numbers each denoting the runs in the ith ODI match, ith Test Match and ith T-20 match respectively. The value of Kohli's ith match would be the sum of his score in the ith T-20 match, ith Test match and ith ODI match. Kohli knows that Monk is extremely quick at finding the k th smallest value of all his innings, so he twists his query a bit. He asks the Monk to delete the kth smallest value once that is answered by the Monk. If Kohli comes up with a number k which is greater than the number of matches remaining in Kohli's career, the Monk should say that the answer is 1. So much complication confuses the Little Monk and he asks for your help!Use priority queue and solve the problem. INPUT: The first line contains an integer N, which denotes the number of matches played by Virat Kohli. The next three lines contain N integers each denoting the number of runs scored in ODI, T20 and Test respectively. The next line contains an integer Q, denoting the number of questions Virat is going to ask. The next Q lines contain an integer K, denoting the Kth smallest value which the Monk has to answer. OUTPUT: Print answer to each query in a new line. In case of an invalid query, print 1. EXPLANATION: Consider the input as: 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 3 1 3 11 The N values are: 3, 6, 9, 12, 15. So, the first smallest is 3, and now the new value array is: 6, 9, 12, 15, so the third smallest value now is 12. In the third query is invalid because the value array contains only 3 numbers, and we're asking the Monk to find the 11th smallest number, so the answer is 1. The output should be: 3 12 -1

Source Code

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

long long scores[1001110];

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

    if(*(long long int*)a > *(long long int*)b)
    return 1;
    else if(*(long long int*)a < *(long long int*)b)
    return -1;
    else
    return 0;

}

int main()
{
 long long int q,x,i,n;
 scanf("%lld",&n);
 for(i=0;i<n;i++)
 {
  scanf("%lld",&scores[i]);
 }
 for(i=0;i<n;i++)
 {
  scanf("%lld",&x);
  scores[i]+=x;
 }
 for(i=0;i<n;i++)
 {
  scanf("%lld",&x);
  scores[i]+=x;
 }
 qsort(scores,n,sizeof(long long),cmp);
 scanf("%lld",&q);
 while(q--)
 {
     int flag=0,count;
  scanf("%lld",&x);
  count=0;
  for(i=0;i<n;i++)
  {
   if(scores[i]>=0)
    count++;
   if(count==x)
    {
        flag=1;
        break;
    }
  }
  if(flag)
  {
   printf("%lld\n",scores[i]);
   scores[i]=-1;
  }
        else
            printf("-1\n");
    }
 return 0;
}
```

Sample Input

```
5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
3
1
3
11
```

Sample Output

```
3
12
-1
```

Result

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

Q. Little Monk and Goblet of Fire

Albus Dumbledore announced that the school will host the legendary event known as Wizard Tournament where four magical schools are going to compete against each other in a very deadly competition by facing some dangerous challenges. Since the team selection is very critical in this deadly competition. Albus Dumbledore asked Little Monk to help him in the team selection process. There is a long queue of students from all the four magical schools. Each student of a school have a different roll number. Whenever a new student will come, he will search for his schoolmate from the end of the queue. As soon as he will find any of the schoolmate in the queue, he will stand behind him, otherwise he will stand at the end of the queue. At any moment Little Monk will ask the student, who is standing in front of the queue, to come and put his name in the Goblet of Fire and remove him from the queue. There are QQ operations of one of the following types: Exy: A new student of school x (1x4) whose roll number is y(1y50000) will stand in queue according to the method mentioned above. D: Little Monk will ask the student, who is standing in front of the queue, to come and put his name in the Goblet of Fire and remove him from the queue Now Albus Dumbledore asked Little Monk to tell him the order in which student put their name. Little Monk is too lazy to that so he asked you to write a program to print required order. Note: Number of dequeue operations will never be greater than enqueue operations at any point of time. Input Format: First line contains an integer Q (1Q100000), denoting the number of operations. Next Q lines will contains one of the 2 types of operations. Output Format: For each 2nd type of operation, print two space separated integers, the front student's school and roll number.

Source Code

```
#include <stdio.h>
  #include<stdlib.h>
  struct node
  {
    int sch;
    int roll;
    struct node *next;
  }*front=NULL,*rear=NULL,*s1=NULL,*s2=NULL,*s3=NULL,*s4=NULL;
  typedef struct node node;
  void del();
  void insert(int,int);
  int main()
  {
    int q,ro,s;
    char qtype;
    node *ir;
    scanf("%d",&q);
    getchar();
    while(q>=1)
    {
      scanf("%c",&qtype);
       //printf("qtype=%c",qtype);
      if(qtype=='E')
      {
        scanf("%d%d",&s,&ro);
        getchar();
        insert(s,ro);
      }
      else
      {
        del();
        getchar();
      }

   /*  if(front!=NULL)
     {
     ir=front;
     while(ir!=NULL)
 {
 printf("%d---%d ",ir->sch,ir->roll);
 ir=ir->next;
 }
 printf("\n");
 }*/
q--;}
      return 0;
  }
void insert(int s,int r)
  {
    node *p;
    p=(node*)malloc(sizeof(node));
    p->sch=s;
    p->roll=r;
    if(rear==NULL)
{
 front=p;
 rear=p;
 p->next=NULL;
}
    if(s==1)
    {
     if(s1!=NULL)
     {
      if(s1->next==NULL)
      rear=p;
      p->next=s1->next;
       s1->next=p;
     }
     else
     {
      rear->next=p;
      p->next=NULL;
      rear=p;
 }
      s1=p;
    }
    else if(s==2)
{
// printf("\nyyyyyyyyy\n");

 if(s2!=NULL)
 {
  if(s2->next==NULL)
      rear=p;
 p->next=s2->next;
 s2->next=p;
 }
 else
 {
  rear->next=p;
  p->next=NULL;
  rear=p;
 }
 s2=p;
}
else if(s==3)
{
 if(s3!=NULL)
 {
  if(s3->next==NULL)
      rear=p;
 p->next=s3->next;
 s3->next=p;
 }
 else
 {
  rear->next=p;
  p->next=NULL;
  rear=p;
 }
 s3=p;
}
else
{
 if(s4!=NULL)
 {
  if(s4->next==NULL)
      rear=p;
 p->next=s4->next;
 s4->next=p;
 }
 else
 {
  rear->next=p;
  p->next=NULL;
  rear=p;
 }
 s4=p;
}
  }
void del()
{
 node *r=front;
 printf("%d %d\n",r->sch,r->roll);
 if(front==s1)
 {
  if(front->next!=NULL&&front->next->sch==1)
  s1=s1->next;
  else
  s1=NULL;
 }
 if(front==s2)
 {
  if(front->next!=NULL&&front->next->sch==2)
  s2=s2->next;
  else
  s2=NULL;
 }
 if(front==s3)
 {
  if(front->next!=NULL&&front->next->sch==3)
  s3=s3->next;
  else
  s3=NULL;
 }
 if(front==s4)
 {
  if(front->next!=NULL&&front->next->sch==4)
  s4=s4->next;
  else
  s4=NULL;
 }
 if(front->next==NULL)
 {
  front=rear=NULL;
 }
 else
   front=front->next;
  /*printf("ffff");
 if(front!=NULL)
 printf("%d--%d\n",front->sch,front->roll);
 */
   }
```

Sample Input

```
5
E 1 1
E 2 1
E 1 2
D
D
```

Sample Output

```
1 1
1 2
```

Result

Thus, Program " **Little Monk and Goblet of Fire**" has been successfully executed

Q. Little Monk and Root

Little Monk finally meets another favorite cricketer of his: Joe Root.He wants to show Root the power of his magical function. Little Monk's function has the special ability to reduce a number to its one third value. Mathematically, f(x) = ceiling (x/3), where ceiling(x) is the smallest integer greater than or equal to x. The Monk wants to reduce the career average of Root's career, so he decides to apply the function K number of times on various innings of his career. He can, of course, apply the function on the same innings multiple times. You're given the number of N innings and the runs scored by Root in his career, and you've to help reduce Root's final career average by applying Monk's special function K number of times on the innings of his career. And print his original average and his new career average.Use priority queue and solve the problem. INPUT: The first line contains two integers, N and K, denoting the number of innings of Root and the number of times the Monk can apply the magical function. The next line contains N space separated integers denoting the runs scored by Root in each innings. OUTPUT: You've to print the initial career average of Joe Root, and the new average on the same line separated by a space. The difference till 10^6 in the answer will be accepted. Note: You can safely assume that Joe Root was dismissed in every inning he played, so the role of a not-out in an innings is not important. Also, the difference till 10^6 in the answer will be accepted. EXPLANATION: Consider the input as: 5 1 20 30 40 50 60 The initial scores: 20, 30, 40, 50, 60. The new scores after the function: 20, 30, 40, 50, 20. The output will be: 40.000000 32.000000

Source Code

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int heapsize,n;
void maxheapyfy(long long int *a,int i)
{
 int l=2*i;
 int r=2*i+1;
 int largest;

 if(l<=heapsize&&a[l]>a[i])
  largest=l;
 else
  largest=i;
 if(r<=heapsize&&a[r]>a[largest])
  largest=r;
 if(largest!=i)
  {
   int temp;
   temp=a[largest];
   a[largest]=a[i];
   a[i]=temp;
  maxheapyfy(a,largest);
   }

}
void buildheap(long long int *a)
{ int i;

 for(i=n/2;i>0;i--)
 {
  maxheapyfy(a,i);
 }

}
int main()
{

 long long int t;
 scanf("%d",&n);
 scanf("%lld",&t);
 long long int a[n];
 int i;
 heapsize=n;
 for(i=1;i<=n;i++)
  scanf("%lld",&a[i]);
 buildheap(a);
 double xx=0;
 for(i=1;i<=n;i++)
 {
  xx=xx+a[i];

 }
 xx=xx/n;
 for(i=0;i<t;i++)
 {   int p;
     if(a[1]%3==0)
         p=a[1]/3;
        else
        p=a[1]/3+1;
     a[1]=p;
     maxheapyfy(a,1);

 }
 double x=0;
 for(i=1;i<=n;i++)
 {   //printf("%ld ",a[i]);
  x=x+a[i];

 }
 x=x/n;
 printf("%.6f %0.6f\n",xx,x);

 return 0;
}
```

Sample Input

```
5 1
20 30 40 50 60
```

Sample Output

```
40.000000 32.000000
```

Result

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

Course: DATA-STRUCTURE | **Session:** Queue Implementation | **Timestamp:** October 5, 2017, 9:52 am | **Register Number:** RA1611003010583

Q. Little Monk and Williamson

Little Monk is a huge cricket fan, so he decides that he'll meet his five favorite cricketers in this problem-set of heaps and priority queues. And he'll try to impress them. The first cricketer he wants to impress is: Kane Williamson. He asked Kane to answer some queries in a press interview. Kane is irritated by Little Monk's constant interruption during his interview, so he decides to give Monk a run for his money by asking him the answer to various queries. Williamson will give a query of the types mentioned below, to the Monk and will expect him to answer those. -Push X -->Insert Williamson's score in an array(Query type1) -Diff --> Find out the difference between Willamson's current highest and current lowest score, currently present in the array. And then remove a singular instance of those values from the array.(Query type 2).In case, the current lowest and current highest score are same, then only one instance of that score will be removed from the array. -CountHigh --> Find out the number of times Williamson has scored his current highest score, currently present in array.(Query type 3) -CountLow --> Find out the number of times Williamson has scored his current lowest score, currently present in array.(Query type 4) INPUT: The first line contains an integer Q, which denotes the number of queries which have to be dealt by The Monk. The next Q lines will contain a query like the ones mentioned above. OUTPUT: For the query type 2, 3 and 4, print the answer in a new line. If there is no record of any innings, that is, the array of Williamson's score is empty for query type 2, 3 and 4, then print 1. EXPLANATION: Consider the input: 10 CountHigh Push 442 CountHigh Push 7555 Diff Push 2799 Diff Push 8543 Diff Diff Since there's no score currently, the answer to the first query is1. Then we insert 442 runs as Williamson's score. Since there's only one instance of 442 and that is the highest number in the array, CountHigh would be 1. Then we insert 7555 as Williamson's next score. Then, the difference is 7113 and that means 442 and 7555 are also removed from the array. After pushing 2799, and finding out the difference would be 0 since the highest and the lowest score is same. For the last Diff query, since there is no number in the array, the answer would be 1. Hence,the output is: -1 1 7113 0 0 -1

Source Code

```
#include<stdio.h>
int frequency[111111];
int minheap[111111];
int maxheap[111111];
int min_heapsize;
int max_heapsize;
void swap(int *x,int *y)
{
	int temp = *x;
	*x=*y;
	*y= temp;
}
void insert_minheap(int k)
{

	minheap[min_heapsize] = k;
	int i =min_heapsize;
	min_heapsize++;
	while(i!=0 &&  (minheap[(i-1)/2] > minheap[i]))
		{
			swap(&minheap[(i-1)/2], &minheap[i]);
			i = (i-1)/2;
		}
}

void insert_maxheap(int k)
{

	maxheap[max_heapsize]= k;
	int i =max_heapsize;
	 max_heapsize++;
	while(i!=0 &&   (maxheap[(i-1)/2] < maxheap[i]))
		{
			swap(&maxheap[(i-1)/2], &maxheap[i]);
			i = (i-1)/2;
		}
}
 void minheapify(int i)
{
	int smallest = i;
	if( 2*i+1 < min_heapsize && (minheap[2*i+1] < minheap[i]))
		smallest = 2*i +1;
	if( 2*i+2 < min_heapsize && (minheap[2*i+2] < minheap[smallest]))
		smallest = 2*i +2;
	if(smallest!=i)
		{
			swap(&minheap[i],&minheap[smallest]);
			minheapify(smallest);
		}
}


void maxheapify(int i)
{
	int largest = i;
	if( 2*i+1 < max_heapsize && (maxheap[2*i+1] > maxheap[i]))
		largest = 2*i +1;
	if( 2*i+2 < max_heapsize && (maxheap[2*i+2] > maxheap[largest]))
		largest = 2*i +2;
	if(largest!=i)
		{
			swap(&maxheap[i],&maxheap[largest]);
			maxheapify(largest);
		}
}

	int extractmin()
{
	if(min_heapsize <= 0)
		{
			return -1;
		}
	if(min_heapsize == 1)
		{
			min_heapsize--;
			return minheap[0];
		}
	else
		{
			int root = minheap[0];
			minheap[0]= minheap[min_heapsize-1];
			min_heapsize--;
			minheapify(0);
			return root;
		}
}

int extractmax()
{
	if(max_heapsize <= 0)
		{
			return -1;
		}
	if(max_heapsize == 1)
		{
			max_heapsize--;
			return maxheap[0];
		}
	else
		{
			int root = maxheap[0];
			maxheap[0] = maxheap[max_heapsize-1];
			max_heapsize--;
			maxheapify(0);
			return root;
		}
}


char input[15];

int main()
{
	int queries;
	scanf("%d",&queries);
	while(queries--)
		{

			scanf("%s",input);
			if(input[0] == 'P')
				{
					int x;
					scanf("%d",&x);
					frequency[x]++;
					insert_maxheap(x);
					insert_minheap(x);
				}
			else if(input[0] == 'C' && input[5]=='L')
				{
					while(min_heapsize!=0 && (frequency[minheap[0]] ==0))
						extractmin();

					if(min_heapsize == 0)
						printf("-1\n");
					else
						printf("%d\n",frequency[minheap[0]]);
				}
			else if(input[0]=='C' && input[5] == 'H')
				{
					while(max_heapsize!=0 && (frequency[maxheap[0]] ==0))
						extractmax();
					if(max_heapsize == 0)
						printf("-1\n");
					else
						printf("%d\n",frequency[maxheap[0]]);
				}
			else
			 {
					while(max_heapsize!=0 && frequency[maxheap[0]] ==0)
						extractmax();
					while(min_heapsize!=0 && frequency[minheap[0]] ==0)
						extractmin();
					if(min_heapsize != 0 && max_heapsize!=0)
						{
							if(minheap[0] == maxheap[0])
								frequency[maxheap[0]]--;
							else{
								frequency[minheap[0]]--;
								frequency[maxheap[0]]--;
							}
						printf("%d\n",maxheap[0] - minheap[0]);
						extractmax();
						extractmin();
					}
					else
						printf("-1\n");


				}
		}
	return 0;

}
```

Sample Input

```
10
CountHigh
Push 442
CountHigh
Push 7555
Diff
Push 2799
Diff
Push 8543
Diff
Diff
```

Sample Output

```
-1
1
7113
0
0
-1
```

Result

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

Q. Dwayne and Cursed Tree

Dwayne has an array A having N distinct integers and a Binary Search Tree which is initially empty. He inserts all the elements of the array from index 1 to N in the BST in the order given in the array. But wait! The tree so formed turns out to be cursed. Dwayne is having some weird experiences since he made that tree. So, now to stop all that, Dwayne has two options, to destroy the BST or to pray to God and ask for a solution. Now since Dwayne has to use this BST in a Code Challenge, he cannot destroy it. So he prays to God. God answer his prayers and sends an angel named Micro. Now, Micro asks Dwayne to find something. He tells him two values, X and Y, present in the BST and ask him to find the maximum value that lie in the path between node having value X and node having value Y. (including X and Y). Now since, Dwayne is very afraid of that tree he asks for your help. Input: First line consists of a single integer denoting N.Second line consists of N space separated integers denoting the array A.Third line consists of two space separated integers denoting X and Y. Output: Print the maximum value that lie in the path from node having value X and node having value Y in a new line. Constraints: 1N10^5 1A[i]10^9 It is ensured that values X and Y are present in the array.

Source Code

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

typedef struct node
{
   int data;
   struct node* left;
   struct node* right;
}node;

node* createNode(int key)
{
   node* temp=(node*)malloc(sizeof(node));
   temp->data=key;
   temp->left=NULL;
   temp->right=NULL;
   return temp;
}
node* insertNode(node* root, int key)
{
   if(root==NULL)
   {
      return createNode(key);
   }
   else
   {
      if(root->data<key)
      {
         root->right=insertNode(root->right,key);
      }
      else
      {
         root->left=insertNode(root->left,key);
      }
      return root;
   }
}

int getMax(node* root, int mn, int mx)
{
   if(root!=NULL)
   {
      if(root->data>mx)
      {
         return getMax(root->left,mn,mx);
      }
      else if(root->data<mn)
      {
         return getMax(root->right,mn,mx);
      }
      else
      {
         int mxm=0;
         node* temp=root;
         mxm=root->data;
         while(temp!=NULL && temp->data!=mx)
         {
            if(temp->data>mx)
            {
               if(mxm<temp->data)
               {
                  mxm=temp->data;
               }
               temp=temp->left;
            }
            else
            {
               temp=temp->right;
            }
         }
         if(mxm<mx)
         {
            mxm=mx;
         }
         return mxm;
      }
   }

}
int main()
{
   //printf("Hello World!\n");
   int n;
   scanf("%d",&n);
   int* arr=(int*)malloc(n*sizeof(int));
   int i;
   node* root=NULL;
   for(i=0;i<n;i++)
   {
      scanf("%d",&arr[i]);
      root=insertNode(root,arr[i]);
   }
   int a,b;
   scanf("%d %d",&a,&b);
   int mn=(a<b?a:b);
   //printf("%d ",mn);
   int mx=a+b-mn;
   //printf("%d\n ",mx);
   int ans=getMax(root,mn,mx);
   printf("%d ",ans);
   return 0;
}
```

Sample Input

```
5
4 7 8 6 3
3 7
```

Sample Output

```
7
```

Result

Thus, Program " **Dwayne and Cursed Tree**" has been successfully executed

Q. Michael in the Grass Fields

Michael, high on excitement after solving the rest of the problems , goes on a trek in the mountains . On his way, he encounters Janemba, the evil magician! Janemba takes our Michael to a poison field and plays a game with him described as follows: The poison field is described as NxN matrix,divided into N * N cells. Each cell of the field has a value of discomfort in it pertaining to the poison content. Michael has been cursed with K curses. In each curse, Michael must do one of the following: 1) Choose one row of the field and consume the poisonous fumes from all the cells in that row. The discomfort caused is the sum of discomfort from each cell in the row. After consuming, the discomfort of all cells in the row increases by one. 2) Choose one column of the field and consume the poisonous fumes from all the cells in that column. The discomfort caused is the sum of discomfort from each cell in the column. After consuming, the discomfort of all cells in the column increases by one. Our Michael has a level of tolerance. A very high level of discomfort will cause him to die! Help him out by finding the Minimum discomfort possible by optimally completing the curses. Input: First line contains T, the number of test cases. T test cases follow. First line of each test case contains two space separated integers N and K. N lines follow. Each of the lines contains N space-separated integers representing Mij, the value of discomfort of cells in that row. Output: For each test case, output the minimum discomfort possible in a new line. Constraints: 1 T 100 1 N 200 1 K 400 1 Mij 1000

Source Code

```
#include<stdio.h>
 #include<string.h>
 #define left(a) (a*2+1)
 #define right(a) (a*2+2)

 typedef struct{
  int sum;
  int index;
 }node;

 node heaprow[205];
 node heapcol[205];
 int matrix[205][205];
 int size, curses;

 int swap(node *a, node *b){
  node temp;
  temp.sum = a->sum;
  temp.index = a->index;
  a->sum = b->sum;
  a->index = b->index;
  b->sum = temp.sum;
  b->index = temp.index;
  return 0;
 }

int input(){
  scanf("%d%d", &size, &curses);
  int i, j;
  memset(heaprow, 0, sizeof(heaprow));
     memset(heapcol, 0, sizeof(heapcol));

  for(i = 0; i < size; i++){
   heaprow[i].index = i;
   heapcol[i].index = i;
   for(j = 0; j < size; j++){
    scanf("%d", &matrix[i][j]);
    heaprow[i].sum += matrix[i][j];
    heapcol[j].sum += matrix[i][j];
   }
  }
  return 0;
 }

 int heapify(node heap[], int i, int heapsize){
  int smallest = i;
  if( left(i) <= heapsize-1 && heap[smallest].sum > heap[left(i)].sum )
   smallest = left(i);
  if( right(i) <= heapsize-1 && heap[smallest].sum > heap[right(i)].sum)
   smallest = right(i);
  if(smallest != i){
   swap(&heap[i], &heap[smallest]);
   heapify(heap, smallest, heapsize);
  }
  return 0;
 }

   int buildheap(node heap[], int size){
  int i;
  for(i = size/2-1; i>= 0; i--)
   heapify(heap, i, size);
  return 0;
 }

 int myselect(node heap[]){  //update the row/col that was selected by adding size to the sum
  heap[0].sum += size;
  heapify(heap, 0, size);

 }

 int main(){

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

   input();
   long long int tolerance = 400*200*1010;
   int extrarow = 0;
   int extracol = 0;
   buildheap(heaprow, size);
   buildheap(heapcol, size);
   long long int sumrow[405]; //increase in tolerance if we select row for i times
   long long int sumcol[405]; //increase in tolerance if we select col for i times
   int i;
   sumrow[0] = 0;
   sumcol[0] = 0;
   for(i = 1; i <= curses; i++){
    sumrow[i] = sumrow[i-1] + heaprow[0].sum;
    myselect(heaprow);
    sumcol[i] = sumcol[i-1] + heapcol[0].sum;
    myselect(heapcol);
          }
   for(i = 0; i <= curses; i++){
    if(tolerance > (sumrow[i] + sumcol[curses-i] + i*(curses-i)))
     tolerance = (sumrow[i] + sumcol[curses-i] + i*(curses-i));
   }

   printf("%lld\n", tolerance);
  }
 }
```

Sample Input

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

Sample Output

```
3
8
14
22
```

Result

Thus, Program " **Michael in the Grass Fields**" has been successfully executed

Course: DATA-STRUCTURE **Session:** Tree **Timestamp:** October 5, 2017, 9:52 am **Register Number:** RA1611003010583

Q. Dwayne and his Friends

Dwayne is standing at the door of his classroom. There are currently N students in the class, i'th student got Ai candies. There are still M more students to come. At every instant, a student enters the class and wishes to be seated with a student who has exactly the same number of candies. For each student, Dwayne shouts YES if such a student is found, NO otherwise. Input: First line contains an integer T. T test cases follow. First line of each case contains two space-separated integers N and M. Second line contains N + M space-separated integers, the candies of the students. Output: For each test case, output M new line, Dwayne's answer to the M students. Print "YES" (without the quotes) or "NO" (without the quotes) pertaining to the Dwayne's answer. Constraints: 1 T 10 1 N, M 10^5 0 Ai 10^12

Source Code

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

#define N 100000

struct A {
 long long a;
 int i, yes;
} bb[N];

int acompare(const void *a, const void *b) {
 long long *pa = (long long *) a;
 long long *pb = (long long *) b;

 return *pa == *pb ? 0 : (*pa < *pb ? -1 : 1);
}

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

 return pa->a == pb->a ? pa->i - pb->i : (pa->a < pb->a ? -1 : 1);
}

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

 return pa->i - pb->i;
}

int main() {
 int t;

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

  scanf("%d%d", &n, &m);
  for (i = 0; i < n; i++)
   scanf("%lld", &aa[i]);
  qsort(aa, n, sizeof *aa, acompare);
  for (j = 0; j < m; j++) {
   scanf("%lld", &bb[j].a);
   bb[j].i = j;
  }
  qsort(bb, m, sizeof *bb, bcompare);
  for (j = i = 0; j < m; j++) {
   while (i < n && aa[i] < bb[j].a)
    i++;
   bb[j].yes = (j > 0 && bb[j].a == bb[j - 1].a)
    || (i < n && aa[i] == bb[j].a);
  }
  qsort(bb, m, sizeof *bb, icompare);
  for (j = 0; j < m; j++)
   printf("%s\n", bb[j].yes ? "YES" : "NO");
 }
 return 0;
}
```

Sample Input

```
1
2 3
3 2 9 11 2
```

Sample Output

```
NO
NO
YES
```

Result

Thus, Program " **Dwayne and his Friends**" has been successfully executed

Course: DATA-STRUCTURE **Session:** Tree **Timestamp:** October 5, 2017, 9:53 am **Register Number:** RA1611003010583

Q. Roy and Trending Topics

Roy is trying to develop a widget that shows Trending Topics (similar to Facebook) on the home page of ABC Academy. He has gathered a list of N Topics (their IDs) and their popularity score (say z-score) from the database. Now z-score change everyday according to the following rules: When a topic is mentioned in a 'Post', its z-score is increased by 50. A 'Like' on such a Post, increases the z-score by 5. A 'Comment' increases z-score by 10. A 'Share' causes an increment of 20. Now the Trending Topics are decided according to the change in z-score. One with the highest increment comes on top and list follows. Roy seeks your help to write an algorithm to find the top 5 Trending Topics. If change in z-score for any two topics is same, then rank them according to their ID (one with higher ID gets priority). It is guaranteed that IDs will be unique. Input format: First line contains integer N N lines follow Each contains 6 space separated numbers representing Topic ID, current z-score - Z, Posts - P, Likes - L, Comments - C, Shares - S Output format: Print top 5 Topics each in a new line. Each line should contain two space separated integers, Topic ID and new z-score of the topic. Constraints: 1 N 10^6 1 ID 10^9 0 Z, P, L, C, S 10^9

Source Code

```
#include <stdio.h>
#include <stdlib.h>
#define l long long int
typedef struct{
   l id;
   l z;
   l x;
}node;
int main()
{
   long int n,i;
   int j,k;
   l d,z1,l1,c,s,p,res;
   node a[5],temp;
   scanf("%ld",&n);
   for(i=0;i<=n;i++)
   {
      scanf("%lld%lld%lld%lld%lld%lld",&d,&z1,&p,&l1,&c,&s);
      //printf("%lld %lld %lld %lld %lld %lld\n",d,z1,p,l1,c,s);
      res=50*p+5*l1+10*c+20*s-z1;
      //printf("%lld \n",res);
      if(i<5)
       {
          a[i].id=d;
          a[i].z=res;
          a[i].x=res+z1;
       }
      if(i==4)
       {
         for(i=0;i<5;i++)
           {
              for(j=0;j<4;j++)
              {
                 if(a[j].z<a[j+1].z)
                 {
                    temp=a[j];
                    a[j]=a[j+1];
                    a[j+1]=temp;
                 }
                 else if(a[j].z==a[j+1].z&&a[j].id <a[j+1].id)
                 {
                    temp=a[j];
                    a[j]=a[j+1];
                    a[j+1]=temp;
                 }
              }
           }
       }
      else if(i>=5)
      {
        j=0,k=4;
        while(j<5 && res<a[j].z)
            j++;
        while(j<5 && (res==a[j].z)&& d <a[j].id)
            j++;
        while(k>=0 && k>j)
        {
           a[k].id=a[k-1].id;
           a[k].z=a[k-1].z;
           a[k].x=a[k-1].x;
           k--;
        }
        if(j!=5){
        a[j].id=d;
        a[j].z=res;
        a[j].x=res+z1;
        }
      }
   }
      for(i=0;i<5;i++)
        printf("%lld %lld\n",a[i].id,a[i].x);
   return 0;
}
```

Sample Input

```
8
1003 100 4 0 0 0
1002 200 6 0 0 0
1001 300 8 0 0 0
1004 100 3 0 0 0
1005 200 3 0 0 0
1006 300 5 0 0 0
1007 100 3 0 0 0
999 100 4 0 0 0
```

Sample Output

```
1003 200
1002 300
1001 400
999 200
1007 150
```

Result

Thus, Program "**Roy and Trending Topics**" has been successfully executed

Q. Chris and BST

Chris as always has brought a new task for Fredo. He asks Fredo to create a Binary Search Tree (BST) with L levels and 2^L1 nodes. Let the sum of all node values in the BST be M. He further adds that M should be smallest possible integer greater than S, where S is an integer given by Chris. He states the rules of creating BST as follows: The left sub-tree contains only nodes with values less than or equal to the parent node; the right sub-tree contains only nodes with values greater than the parent node. If a node has level i, then the subtree rooted at that node should have exactly 2^Li number of distinct values in the subtree. Note that it is the number of distinct values in the subtree and not the number of nodes in the subtree. If a is the smallest value in the tree and b is the largest value, then all values from a to b must come atleast once in the tree. Level of root is 1, next level is 2 and so on. Now, he will ask two type of queries to Fredo. Type 0: Find the closest node to root whose value is equal to val and print path to that node from the root. If root has value equal to val, print "root". Else print "l" when we visit left child of any node and "r" when we visit right child of any node. Type 1: Tell the value of kth node in the tree. The nodes are numbered as: enter image description here Here 1,2 and so on are node numbers and A,B etc. are values of nodes. Finally, he will ask Fredo Q queries, each query belonging to one of the types mentioned above. Since, Fredo is new to this concept, help him complete this task. Input Format: First line consists of two integers L and S as described in the question. Second line consists of Q, denoting the number of queries. Each of the following Q lines consists of two integers. The first integer denoting the type of query and second denoting either val or k as described in the queries. Output Format: For each query, print the required answer in a separate line. Input Constraints: 1L30 1S10^18 1Q10^5 0type1 1k2^L1 val is in between minimum and maximum value in tree(inclusive). Values of S, val and k are such that answer would always exist for them. Node values will be non-negative for all inputs. M will never exceed 210^18

Source Code

```
#include <stdio.h>

long long power(int x, int y) {
 long long p;

 if (y == 0)
  return 1;
 p = power(x, y / 2);
 p *= p;
 if (y % 2 == 1)
  p *= x;
 return p;
}

int query0(int i, long long val, long long sl, long long sr) {
 long long x = (sl + sr - 1) / 2;

 if (val == x)
  return i;
 if (val < x)
  return query0(i * 2, val, sl, (sl + sr) / 2);
 else
  return query0(i * 2 + 1, val, (sl + sr) / 2, sr);
}

long long query1(int i, int j, int k, long long sl, long long sr) {
 return j == k ? (sl + sr - 1) / 2 : (k & 1 << (i - 1)) == 0
  ? query1(i - 1, j * 2, k, sl, (sl + sr) / 2)
  : query1(i - 1, j * 2 + 1, k, (sl + sr) / 2, sr);
}

void print(int i) {
 if (i > 1) {
  print(i / 2);
  printf("%c", i % 2 == 0 ? 'l' : 'r');
 }
}

int main() {
 int l, q, n, cnt;
 long long s, p, start;

 scanf("%d%lld", &l, &s);
 cnt = power(2, l - 1);
 n = power(2, l) - 1;
 p = power(4, l - 1) - n;
 start = p > s ? 0 : (s - p) / n + 1;
 scanf("%d", &q);
 while (q-- > 0) {
  int type, k;
  long long val;

  scanf("%d", &type);
  if (type == 0) {
   scanf("%lld", &val);
   if (val == (start + start + cnt - 1) / 2)
    printf("root\n");
   else {
    print(query0(1, val, start, start + cnt));
    printf("\n");
   }
  } else {
   int x;

   scanf("%d", &k);
   for (x = 0; (long long) 1 << x <= k; x++)
    ;
   printf("%lld\n", query1(x - 1, 1, k, start, start + cnt));
  }
 }
 return 0;
}
```

Sample Input

```
3 16
5
0 3
0 4
1 4
1 7
0 5
```

Sample Output

```
root
r
2
5
rr
```

Result

Thus, Program " **Chris and BST** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Arrays **Timestamp:** October 5, 2017, 10:00 am **Register Number:** RA1611003010583

Q. Micro and Array Update

Micro purchased an array A having N integer values. After playing it for a while, he got bored of it and decided to update value of its element. In one second he can increase value of each array element by 1. He wants each array element's value to become greater than or equal to K. Please help Micro to find out the minimum amount of time it will take, for him to do so. Input: First line consists of a single integer, T, denoting the number of test cases. First line of each test case consists of two space separated integers denoting N and K. Second line of each test case consists of N space separated integers denoting the array A. Output: For each test case, print the minimum time in which all array elements will become greater than or equal to K. Print a new line after each test case. Constraints: 1T5 1N10^5 1A[i],K10^6

Source Code

```
#include <stdio.h>

int main()
{
   int t,n,i,c=0,k,max=0;
   scanf("%d",&t);
   while(t--)
   {
      k=0;n=0;max=0;
      scanf("%d%d",&n,&k);
      int a[n];
      for(i=0;i<n;i++)
      {
         scanf("%d",&a[i]);
         if(a[i]<k)
         {
            c=k-a[i];
            if(c>=max)
               max=c;
         }
      }
      printf("%d\n",max);
   }
   return 0;
}
```

Sample Input

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

Sample Output

```
3
0
```

Result

Thus, Program "**Micro and Array Update**" has been successfully executed

Course: DATA-STRUCTURE **Session:** Arrays **Timestamp:** October 5, 2017, 10:00 am **Register Number:** RA1611003010583

Q. Long ATM Queue

Due to the demonetization move, there is a long queue of people in front of ATMs. Due to withdrawal limit per person per day, people come in groups to withdraw money. Groups come one by one and line up behind the already present queue. The groups have a strange way of arranging themselves. In a particular group, the group members arrange themselves in increasing order of their height(not necessarily strictly increasing). Swapy observes a long queue standing in front of the ATM near his house. Being a curious kid, he wants to count the total number of groups present in the queue waiting to withdraw money. Since groups are standing behind each other, one cannot differentiate between different groups and the exact count cannot be given. Can you tell him the minimum number of groups that can be observed in the queue? Input format: The first line of input contains one positive integer N. The second line contains N space-separated integers H[i] denoting the height of i-th person. Each group has group members standing in increasing order of their height. Output format: Print the minimum number of groups that are at least present in the queue? Constraints: 1N1,000,000 1H[i]1,000,000

Source Code

```
#include <stdio.h>

int main()
{
   int N=-1,i=0,group=1;
   scanf("%d",&N);
   int H[N];

   for(i=0;i<N;i++){
      scanf("%d",H+i);
      if(i>0){
         if(H[i-1]>H[i]){
         group++;
      }
      }
   }


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

Sample Input

```
4
1 2 3 4
```

Sample Output

```
1
```

Result

Thus, Program " **Long ATM Queue** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Arrays **Timestamp:** October 5, 2017, 10:00 am **Register Number:** RA1611003010583

Q. Mark the Answer

Our friend Monk has an exam that has quite weird rules. Each question has a difficulty level in the form of an Integer. Now, Monk can only solve the problems that have difficulty level less than X . Now the rules are- Score of the student is equal to the maximum number of answers he/she has attempted without skipping a question. Student is allowed to skip just "one" question that will not be counted in the continuity of the questions. Note- Assume the student knows the solution to the problem he/she attempts and always starts the paper from first question. Given the number of Questions, N ,the maximum difficulty level of the problem Monk can solve , X ,and the difficulty level of each question , Ai can you help him determine his maximum score? Input Format First Line contains Integer N , the number of questions and the maximum difficulty X Monk can solve.Next line contains N integers, Ai denoting the difficulty level of each question. Output Format Maximum score Monk can achieve in the exam. Constraints 1N10^5 1X10^9 1Ai10^9

Source Code

```
#include<stdio.h>
int main ()
{
  unsigned long int n,m;
  //printf("enter the no. of ques and difficulty level");
  scanf(" %lu %lu",&n,&m);
  unsigned long int a[n],i;
  //printf("enter the value of diif");
  for(i=0;i<n;i++)
  {
     scanf("%lu",&a[i]);
  }
  unsigned long int j=0,k=0;
  for (i=0;i<n;i++)
  {
    if(a[i]>m)
    {
      j++;
    }
    if(a[i]<=m&&j<2)
    {
      k++;
    }

  }
  printf("%lu",k);

 return 0;

}
```

Sample Input

```
7 6
4 3 7 6 7 2 2
```

Sample Output

```
3
```

Result

Thus, Program " **Mark the Answer** " has been successfully executed

Course: DATA-STRUCTURE | **Session:** Sorting Techniques | **Timestamp:** October 5, 2017, 10:00 am | **Register Number:** RA1611003010583

Q. Cube Change

Chandan gave his son a cube with side N. The N X N X N cube is made up of small 1 X 1 X 1 cubes. Chandan's son is extremely notorious just like him. So he dropped the cube inside a tank filled with Coke. The cube got totally immersed in that tank. His son was somehow able to take out the cube from the tank. But sooner his son realized that the cube had gone all dirty because of the coke. Since Chandan did not like dirty stuffs so his son decided to scrap off all the smaller cubes that got dirty in the process. A cube that had coke on any one of its six faces was considered to be dirty and scrapped off. After completing this cumbersome part his son decided to calculate volume of the scrapped off material. Since Chandan's son is weak in maths he is unable to do it alone. Help him in calculating the required volume. Input: The first line contains T denoting the number of test cases. Then T lines follow each line contains N that is the side of cube. Output: For each case output the required volume. Constraints: 1 T 100 1 N 10^9 Note: There is no hole or space between 2 smaller cubes.

Source Code

```
#include <stdio.h>

int main()
{
 long long t, n, vol, i;

    scanf("%lld", &t);
    for ( i = 0 ; i < t ; i += 1 ) {
     scanf("%lld", &n);
     if (n == 1)
      vol = 1;
     else
      vol = n*n*n-(n-2)*(n-2)*(n-2);
     printf("%lld\n", vol);
    }

    return 0;
}
```

Sample Input

```
2
1
3
```

Sample Output

```
1
26
```

Result

Thus, Program " **Cube Change** " has been successfully executed

Course: DATA-STRUCTURE | **Session:** Sorting Techniques | **Timestamp:** October 5, 2017, 10:00 am | **Register Number:** RA1611003010583

Q. Monk and Nice Strings

Monk's best friend Micro's birthday is coming up. Micro likes Nice Strings very much, so Monk decided to gift him one. Monk is having N nice strings, so he'll choose one from those. But before he selects one, he need to know the Niceness value of all of those. Strings are arranged in an array A, and the Niceness value of string at position i is defined as the number of strings having position less than i which are lexicographicaly smaller than A[i]. Since nowadays, Monk is very busy with the Code Monk Series, he asked for your help. Note: Array's index starts from 1. Input: First line consists of a single integer denoting N lines follow each containing a string made of lower case English alphabets. Output: Print N lines, each containing an integer, where the integer in ith line denotes Niceness value of string A[i].

Source Code

```
#include<stdio.h>
#include<string.h>
int main()
{
   int n,i,ans,j,cnt ;
   char s[1000][100] ;
   scanf("%d",&n) ;
   for(i=1;i<=n;i++)
      scanf("%s",s[i]) ;
   for(i=1;i<=n;i++)
   {
      cnt=0 ;
      for(j=i-1;j>=1;j--)
      {
         ans=0 ;
         ans = strcmp(s[j],s[i]) ;
         if(ans<0)
            cnt++ ;
      }
      printf("%d\n",cnt) ;
   }
  return 0;
}
```

Sample Input

```
4
a
c
d
b
```

Sample Output

```
0
1
2
1
```

Result

Thus, Program "**Monk and Nice Strings**" has been successfully executed

Q. Match makers

Little Mojo owns a match making company, which even to her surprise is an extreme hit. She says that her success rate cannot be matched (Yeah, wordplay!) in the entire match-making industry. She follows an extremely simple algorithm to determine if two people are matches for each other. Her algorithm is not at all complex, and makes no sense - not even to her. But she uses it anyway. Let's say say that on a given day she decides to select n people - that is, n boys and n girls. She gets the list of n boys and n girls in a random order initially. Then, she arranges the list of girls in ascending order on the basis of their height and boys in descending order of their heights. A girl Ai can be matched to a boy on the same index only, that is, Bi and no one else. Likewise, a girl standing on Ak can be only matched to a boy on the same index Bk and no one else. Now to determine if the pair would make an ideal pair, she checks if the modulo of their heights is 0, i.e., Ai % Bi == 0 or Bi % Ai == 0. Given the number of boys and girls, and their respective heights in non-sorted order, determine the number of ideal pairs Mojo can find. Input format: The first line contains number of test cases. Then, the next line contains an integer, n, saying the number of boys and girls. The next line contains the height of girls, followed by the height of boys. Output format: Print the number of ideal pairs corresponding to every test case.

Source Code

```
#include<stdio.h>

void swap(int[],int,int);
void quicksort(int[],int,int);

inline int readint() {
    int n = 0;
    char c = getchar_unlocked();
    while (!('0' <= c && c <= '9')) {
        c = getchar_unlocked();
    }
    while ('0' <= c && c <= '9') {
        n = n * 10 + c - '0';
        c = getchar_unlocked();
    }
    return n;
}

int main()
{
int t,i,c,b[10001],g[10001],a,n;
t=readint();
while(t--)
{
 c=0;
 n=readint();
 for(i=0;i<n;i++)
 b[i]=readint();
 for(i=0;i<n;i++)
 g[i]=readint();
 quicksort(g,0,n-1);
 quicksort(b,0,n-1);
 for(i=0;i<n;i++)
 {
  if(g[i]%b[n-1-i]==0 || b[n-1-i]%g[i]==0)
  c++;
 }
 printf("%d\n",c);
}
return 0;
  }

void quicksort(int arr[],int left,int right)
{
int i, last;
if(left>=right)
return;
swap(arr,left,(left+right)/2);
last=left;
for(i=left+1;i<=right;i++)
if(arr[i]<arr[left])
swap(arr,++last,i);
swap(arr,left,last);
quicksort(arr,left,last-1);
quicksort(arr,last+1,right);
}

void swap(int arr[],int i,int j)
{
int temp;
temp=arr[i];
arr[i]=arr[j];
arr[j]=temp;
}
```

Sample Input

```
2
4
1 6 9 12
4 12 3 9
4
2 2 2 2
2 2 2 2
```

Sample Output

```
2
4
```

Result

Thus, Program " **Match makers** " has been successfully executed

Course: DATA-STRUCTURE | **Session:** Sorting Techniques | **Timestamp:** October 5, 2017, 10:01 am | **Register Number:** RA1611003010583

Q. Earth and The Meteorites

Once upon a time, the Earth was a flat rectangular landmass. And there was no life. It was then that the sky lit up with meteorites falling from out of space. Wherever they fell on the planet, a river was born, which flowed in all 4 directions (North, East, West, South), till the waters reached the edge of the Earth and simply fell off into space. Now, these rivers criss-crossed and divided the one huge landmass (Pangaea) into many smaller landmasses. Now the lifeless (there was no life, remember?), want to know the number of landmasses on the planet after all the meteorites have fallen. They also want to know the area of the smallest and largest landmass. Can you help the lifeless in this question? Input: First line contains T which is the number of test cases. First line of every test case contains 3 integers N, M, Q where N and M are coordinates of the bottom right corner of the planet and Q is the number of meteorites. The next Q lines contains the coordinates X, Y where each of the meteorites fell. Output: For each test case, output a line containing 3 integers indicating the number of regions, the minimum area and the maximum area.

Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include<math.h>
#define MIN 1000001
void quicksort(long int b[],long int low,long int high);
long int partition(long int b[],long int low,long int high);
int main()
{
    long int t,n,m,i,q,countx,county,region,minx,miny,maxx,maxy;
    scanf("%ld",&t);
    while(t--)
    {
        countx=0;
        county=0;
        scanf("%ld %ld %ld",&n,&m,&q);
        if(q==0)
         printf("1 %ld %ld\n",(n-1)*(m-1),(n-1)*(m-1));
        else
        {
       long int x[q+2],y[q+2];
        for(i=0;i<q;i++)
        {
            scanf("%ld %ld",&x[i],&y[i]);
        }
        x[q]=1;
        y[q]=1;
        x[q+1]=n;
        y[q+1]=m;
        quicksort(x,0,q+1);
        quicksort(y,0,q+1);
        for(i=0;i<q+2;i++)
        {
            countx++;
            while(x[i]==x[i+1]&&i<q+1)
            i++;
        }
        for(i=0;i<q+2;i++)
        {
            county++;
            while(y[i]==y[i+1]&&i<q+1)
            i++;
        }
          //  printf("countx=%d\n",countx);
      //  printf("county=%d\n",county);
        region=(countx-1)*(county-1);
            minx=MIN;
            miny=MIN;
        for(i=0;i<q+1;i++)
        {
            if((x[i+1]-x[i])!=0&&((x[i+1]-x[i])<minx))
            minx=(x[i+1]-x[i]);
            if((y[i+1]-y[i])!=0&&((y[i+1]-y[i])<miny))
            miny=(y[i+1]-y[i]);
        }
        maxx=0;
        maxy=0;

         for(i=0;i<q+1;i++)
        {
            if((x[i+1]-x[i])>maxx)
            maxx=(x[i+1]-x[i]);
            if((y[i+1]-y[i])>maxy)
            maxy=(y[i+1]-y[i]);
        }

      // if(q!=0)
        printf("%ld %ld %ld\n",region,(minx*miny),(maxx*maxy));
        }
    //    else
    //    printf("%ld %ld %ld\n",1,(n-1)*(m-1),(n-1)*(m-1));
    }
    return 0;
}
void quicksort(long int b[],long int low,long int high)
{
    if(low<high)
    {
       long int j=partition(b,low,high);
        quicksort(b,low,j);
        quicksort(b,j+1,high);
    }
}
long int partition(long int b[],long int low,long int high)
{
    long int temp,up,down,t,x;
    t=low+rand()%(high-low+1);
    temp=b[t];
    b[t]=b[low];
    b[low]=temp;
    x=b[low];
    down=low-1;
    up=high+1;
    while(1)
    {
        do
        {
            down++;
        }while(b[down]<x);
        do
        {
            up--;
        }while(b[up]>x);
        if(down<up)
        {
            temp=b[down];
            b[down]=b[up];
            b[up]=temp;
        }
        else
        {
            temp=b[low];
            b[low]=b[up];
            b[up]=temp;
            return up;

        }
    }
}
```

Sample Input

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

Sample Output

```
9 1 4
```

Result

Thus, Program " **Earth and The Meteorites**" has been successfully executed

Course: DATA-STRUCTURE | **Session:** Sorting Techniques | **Timestamp:** October 5, 2017, 10:01 am | **Register Number:** RA1611003010583

Q. Chandu and chandni's secret chat

Chandu and chandni Talk on phone for a long time daily. Being afraid that someone will hear their private conversation chandu suggested chandni an idea. He suggested that he will talk only with encrypted strings with her and only she would know, how to decrypt the string. So that even if someone hears, He/She would not be able to anticipate their conversation. Rules of encryption are as follows: 1. String on length N is assumed to be cyclic consisting of lower case English alphabets. 2. In each iteration, we pick the last character and put it in starting of the string. For example: april performing iterations and collecting each string formed in a set until we get the original string. Ex: {april,lapri, ilapr, rilap, prila} 3. sort the set of string in lexicographically reverse order. Ex: {rilap, prila,lapri, ilapr, april } 4. Taking the last character of each string in the set is the encrypted string. Ex: pairl Chandu also sends the position(K) of first letter in encrypted string from original string i.e 2 (p is on position 2 in original string and is the first character of encrypted string) Now, Chandni is ofcourse not that brilliant to decrypt the strings in real time and understand what chandu is saying. So, chandu decided to write a program for the same. Help chandu write this program. Input: First line contains an integer t, which is the number of test cases. Next t lines contain a encrypted string and K as decribed above. Output: Print the decrypted string for each test case. Constraints: 1<=t<=1000 1<=length of string<=10000 1<=k<=length of string

Source Code

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

      char str[10000];
      int cmp(const void *a,const void *b)
      {
       int i=*(int *)a,j=*(int *)b;
       return (str[i]!=str[j])? str[i]<str[j]: i>j;
      }
      int main()
      {
       int i,t;
       scanf("%d",&t);
       for(i=0;i<t;i++)
       {
        int a[10000],n,len,j;
        scanf("%s %d",str,&n);
        len=strlen(str);
        n--;
        for(j=0;j<len;j++)
         a[j]=j;
        qsort(a,len,sizeof(int),cmp);
        for(j=0;j<len;j++,n=a[n])
         printf("%c",str[n]);
        printf("\n");
    }
      return 0;
     }
```

Sample Input

```
2
d 1
pairl 2
```

Sample Output

```
d
april
```

Result

Thus, Program " **Chandu and chandni's secret chat** " has been successfully executed

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. Output format Output how many times Benny will fall down.

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(void)
{
 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: DATA-STRUCTURE	**Session:** Queue Implementation	**Timestamp:** October 5, 2017, 10:01 am	**Register Number:** RA1611003010583

Q. Chefs in Queue

All the chefs (except the Head Chef) are standing in queue to submit their bills. The chefs have different seniority. In all there are N chefs of K different seniority levels. Head Chef gets an interesting thought past his head. He begins to think what if every chef starting from the end of the queue begins to delegate his job of submitting bills to a chef least ahead of him in the queue but junior to him. The Head Chef's fearfulness of this scenario is f = i2 - i1 + 1, where i1 is the index of chef in queue and i2 is the index of the junior chef. Head Chef's total fearfulness of chaos is the product of all the fearfulness in Head Chef's mind. Note if a chef has no junior ahead of him/her in the queue then Head Chef's fearfulness for this Chef is 1. You are required to find the Head Chef's total fearfulness given an arrangement of Chef's in a queue. Since this number can be quite large output it modulo 1000000007. Input. The first line contains two integers N and K denoting the number of chefs and the number of seniority levels. The second line contains N space-separated integers A1, A2, ..., AN denoting the seniority of chefs in the queue. AN denotes front of the queue and A1 denotes end of the queue. Output Output a single integer denoting the total fearfulness of the Head Chef.

Source Code

```
#include <stdio.h>
#define mod 1000000007
int stack[1000000],arr[1000000],ans[1000000];
int main()
{
 int n,k,i;
 long long int result=1;
 scanf("%d%d",&n,&k);
// int arr[n],ans[n];
 for(i=0;i<n;i++)
  scanf("%d",&arr[i]);
 int top=-1;
 for(i=0;i<n;i++){
  stack[++top]=i;
  while(top!=0){
   if(arr[ stack[top-1] ] > arr[ stack[top] ]){
    ans[ stack[top-1] ]=stack[top]-stack[top-1]+1;
    stack[top-1]=stack[top];
    top--;
   }
   else
    break;
  }
 }
 while(top!=-1){
  ans[stack[top]]=1;
  top--;
 }
 result =1;
 for(i=0;i<n;i++){
  result=(result * ans[i])% mod;
 }
/* for(i=0;i<n;i++)
  printf("%d ",ans[i]); */

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

Sample Input

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

Sample Output

```
2
```

Result

Thus, Program " **Chefs in Queue** " has been successfully executed

Course: DATA-STRUCTURE	**Session:** Queue Implementation	**Timestamp:** October 5, 2017, 10:01 am	**Register Number:** RA1611003010583

Q. Minimum Cost of ropes

There are given n ropes of different lengths, we need to connect these ropes into one rope. The cost to connect two ropes is equal to sum of their lengths. We need to connect the ropes with minimum cost.Using heaps and priority queue,solve the problem. INPUT: The first line of input contains an integer T denoting the number of test cases. The first line of each test case is N,N is the number of ropes. The second line of each test case contains N input L[i],length of ropes. OUTPUT: Print the minimum cost for each test case in a new line. Example: Input: 1 4 4 3 2 6 Output: 29 EXPLANATION: For example if we are given 4 ropes of lengths 4, 3, 2 and 6. We can connect the ropes in following ways. 1) First connect ropes of lengths 2 and 3. Now we have three ropes of lengths 4, 6 and 5. 2) Now connect ropes of lengths 4 and 5. Now we have two ropes of lengths 6 and 9. 3) Finally connect the two ropes and all ropes have connected. Total cost for connecting all ropes is 5 + 9 + 15 = 29. This is the optimized cost for connecting ropes. Other ways of connecting ropes would always have same or more cost. For example, if we connect 4 and 6 first (we get three strings of 3, 2 and 10), then connect 10 and 3 (we get two strings of 13 and 2). Finally we connect 13 and 2. Total cost in this way is 10 + 13 + 15 = 38.

Source Code

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

int cmp(const void *a,const void *b)
{
   return( *(int *)a - *(int *)b );
}
int main() {
   int t;
   scanf("%d",&t);
   while(t--)
   {
      int n;
      scanf("%d",&n);
      int a[n],i,sum=0,k,b[n],j;
      for(i=0;i<n;i++)
      {
         scanf("%d",&a[i]);
      }
      qsort(a,n,sizeof(int),cmp);
      k=0;
      while(k<n-1)
      {
         //p=n-k;
         sum+=a[0]+a[1];
         a[1]=a[0]+a[1];
        // j=0;
         for(i=1;i<n-k;i++)
         {
            /* b[j]=a[i];
            j++;*/
            a[i-1]=a[i];
         }
         k++;
        /* for(i=0;i<n-k;i++)
         {
            a[i]=b[i];
         }*/
        // k++;
         qsort(a,n-k,sizeof(int),cmp);
        // k++;
      }
      printf("%d\n",sum);
   }
 //code
 return 0;
}
```

Sample Input

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

Sample Output

```
29
```

Result

Thus, Program " **Minimum Cost of ropes** " has been successfully executed

Course: DATA-STRUCTURE | **Session:** Queue Implementation | **Timestamp:** October 5, 2017, 10:01 am | **Register Number:** RA1611003010583

Q. Card Rotation

Given a sorted deck of cards numbered 1 to N. 1) We pick up 1 card and put it on the back of the deck. 2) Now, we pick up another card , it turns out to be card numbered 1 , we put it outside the deck. 3) Now we pick up 2 cards and put it on the back of the deck. 4) Now, we pick up another card and it turns out to be card numbered 2 , we put it outside the deck. ... We perform this step till the last card.If such arrangement of decks is possible, output the arrangement, if it is not possible for a particular value of N then output -1.Our goal is to come up with an arrangement of cards in our hand such that all of our cards should be placed on the deck, and none of them should be placed outside the deck. If such arrangement is not possible, output -1. Example: Input : 2 4 5 Output : 2 1 4 3 3 1 4 5 2 INPUT: The first line of the input contains the number of test cases 'T', after that 'T' test cases follow. Each line of the test case consists of a single linecontaining an integer 'N'. OUTPUT: If such arrangement of decks is possible, output the arrangement, if it is not possible for a particular value of n then output -1. EXPLANATION: Initial position | No. of cards picked and placed on back of hand | Resultant position | Top card in hand that will be placed on deck 2 1 4 3 | 1 | 1 4 3 2 | 1 4 3 2 | 2 | 2 4 3 | 2 4 3 | 3 | 3 4 | 3 4 | 4 | 4 | 4 Hence, with arrangement '2 1 4 3', we can place all cards on deck and that too with the required sequence of 1 2 3 4.

Source Code

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

void compute(){
 int i=0,j=0,k=0;
 int *a=NULL;
 int *temp=NULL;
 int N=0,size=0,pos=0;
 scanf("%d",&N);
 a=(int *)malloc(N*sizeof(int));
 temp=(int *)malloc(N*sizeof(int));
 for(i=0;i<N;i++){
  a[i]=0;
  temp[i]=0;
 }
 size=0;
 k=N;
 while(k>0){
  size++;
  for(i=size-1;i>=1;i--){
   a[i]=a[i-1];
  }
  a[0]=k;
  for(i=0;i<size;i++){
   temp[i]=a[i];
  }
  pos=k%size;
  i=pos;
  for(j=0;j<size;j++){
   a[i]=temp[j];
   i=(i+1)%size;
  }
  k--;
 }
 for(i=0;i<N;i++){
  printf("%d ",a[i]);
 }
 printf("\n");
 return;
}

int main(){
 int T=0;
 scanf("%d",&T);
 while(T!=0){
  compute();
  T--;
 }
 return 0;
}
```

Sample Input

```
2
4
5
```

Sample Output

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

Result

Thus, Program " **Card Rotation** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Tree **Timestamp:** October 5, 2017, 10:02 am **Register Number:** RA1611003010583

Q. Sum of nodes

Print the Sum of nodes of a tree

Source Code

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


int main()
{
  int n,item,sum=0;
  cout<<"How many numbers do you want to insert ?\n";
  cin>>n;
  for(int i=1;i<=n;i++)
  {
    cout<<"Data "<<i<<": \n";
    cin>>item;
    sum+=item;
  }
  cout<<"Sum: "<<sum;


 return 0;
}
```

Sample Input

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

Sample Output

```
How many numbers do you want to insert ?
Data 1:
Data 2:
Data 3:
Data 4:
Data 5:
Data 6:
Sum: 21
```

Result

Thus, Program " **Sum of nodes** " has been successfully executed

Q. 3 types

Let's consider some weird country with N cities and M bidirectional roads of 3 types. It's weird because of some unusual rules about using these roads: men can use roads of types 1 and 3 only and women can use roads of types 2 and 3 only. Please answer the following very interesting question: what is maximum number of roads it's possible to destroy that the country will be still connected for both men and women? Connected country is country where it's possible to travel from any city to any other using existing roads. Input The first line contains 2 space-separated integer: N and M. Each of the following M lines contain description of one edge: three different space-separated integers: a, b and c. a and b are different and from 1 to N each and denote numbers of vertices that are connected by this edge. c denotes type of this edge. Output For each test case output one integer - maximal number of roads it's possible to destroy or -1 if the country is not connected initially for both men and women. Constraints 1 <= N <= 1000 1 <= M <= 10 000 1 <= a, b <= N 1 <= c <= 3

Source Code

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

#define MAX_N (1000)
#define MAX_M (10000)

typedef struct { int v, t; } pair;

typedef struct { pair * xs; int cnt, cap; } list;

list adj[MAX_N+1];
bool sn[MAX_N+1];

void list_add(list * lst, pair x) {
  if ( lst->cnt == lst->cap ) {
    lst->cap = lst->cap ? lst->cap*2 : 2;
    lst->xs = realloc(lst->xs, lst->cap * sizeof(*lst->xs));
  }
  lst->xs[lst->cnt++] = x;
}

void clear_seen() {
  memset(sn, 0, (MAX_N+1)*sizeof(*sn));
}

int dfs(int x, int use) {
  int i,acc;

  sn[x] = true;
  for ( i=0, acc=1; i < adj[x].cnt; ++i ) {
    pair p = adj[x].xs[i];
    if ( !sn[p.v] && (p.t == use || p.t == 3) )
      acc += dfs(p.v, use);
  }
  return acc;
}

bool connected(int use, int n) {
  int i;

  clear_seen();
  dfs(1, use);
  for ( i=1; i <= n; ++i )
    if ( !sn[i] )
      return false;
  return true;
}

int main() {
  int i,n,m,x,y,t,ccs,ectr;

  scanf("%d %d", &n, &m);
  for ( i=0; i < m; ++i ) {
    scanf("%d %d %d", &x, &y, &t);
    list_add(&adj[x], (pair) { .v = y, .t = t });
    list_add(&adj[y], (pair) { .v = x, .t = t });
  }
  if ( connected(1, n) && connected(2, n) ) {
    clear_seen();
    for ( i=1, ectr=ccs=0; i <= n; ++i )
      if ( !sn[i] )
        ectr += dfs(i, 0)-1, ccs += 1;
    printf("%d\n", m-ectr-2*(ccs-1));
  }
  else {
    printf("-1\n");
  }
  return EXIT_SUCCESS;
}
```

Sample Input

```
5 7
1 2 3
2 3 3
3 4 3
5 3 2
5 4 1
5 2 2
1 5 1
```

Sample Output

```
2
```

Result

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

Q. Mittal wants to go to play

Mittal lives in the Niti Colony. The colony has N houses numbered from 1 to N. There are M bidirectional roads in the colony for travelling between houses. There might be multiple roads between two houses. Mittal lives in the house with index 1. He has friends in all houses of the colony. He is always wanting to visit them to play. But his mom is really strict. She only allows him to go out for K units of time. This time includes the time taken to go to his friend's house , play with the friend and time taken to come back home. You are given Q queries. In each query, Mittal wants to go to his friend in house A , given K units of time. Help him find the maximum time that he can play. If K units of time is not sufficient to visit his friend and return back, Mittal will not go and playing time will be zero. Input: First line contains an integer T. T test cases follow. First line of each test case contains two space-separated integers N, M Next M lines contain three space-separated integers X, Y and C, denoting that there is Bidirectional road between house X and house Y with cost C. Next line contains an integer Q Following Q lines describe the queries. Each query contains two space-separated integers A and K. Output: Print the answer to each query in a new line. Constraints: 1 T 10 1 N, Q 104 1 M 105 1 X, Y, A N 1 K 104 1 C 1000

Source Code

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

#define MAX 10001
//int arr[MAX][MAX]= {0};
int dist[MAX];
char visit[MAX];

void dijsktras(int st, int N, int E);

struct list {
 int node;
 short val;
 struct list *next;
};

struct adj_list {
 struct list *rear[MAX];
 struct list *adj[MAX];
};
struct adj_list adjLs;

int main() {
 int T;
 int N;
 int E;
 int x,y, c, Q, A, KT;
 int i,j,k, ii;

 scanf("%d", &T);
 for(i =0; i<T; i++) {
  scanf("%d %d", &N, &E);
  for(j =0; j<=N; j++) {
  //for(k =0; k<=N; k++)
   {
    adjLs.adj[j] = NULL;
    adjLs.rear[j] = NULL;

   }
        }
        for(i =0; i<E; i++) {
   scanf("%d %d %d", &x, &y, &c);
   struct list *ls1 = (struct list *)malloc(sizeof(struct list));
   ls1->next = NULL;
   ls1->node = y;
   ls1->val = c;
   //visit[x] = 0;
   struct list *ls2 = (struct list *)malloc(sizeof(struct list));
   ls2->next = NULL;
   ls2->node = x;
   ls2->val = c;
   //visit[y] = 0;

// if(arr[x][y] > r || arr[x][y] == 0)
   {
    if(adjLs.adj[x] == NULL) {
     adjLs.adj[x] = ls1;
     adjLs.rear[x] = ls1;
    }
    else {
     adjLs.rear[x]->next = ls1;
     adjLs.rear[x] = ls1;
    }

    if(adjLs.adj[y] == NULL) {
     adjLs.adj[y] = ls2;
     adjLs.rear[y] = ls2;
    }
    else {
     adjLs.rear[y]->next = ls2;
     adjLs.rear[y] = ls2;
    }

   }
  }

  dijsktras(1, N, E);


  scanf("%d", &Q);

  for(ii=1; ii<=Q;ii++ ){
   scanf("%d %d", &A, &KT);
   if(dist[A] == 0x7fffffff) {
    printf("%d\n", 0 );
   }
   else if(KT < (2*dist[A])) {
    printf("%d\n", 0);
   }
   else {
    printf("%d\n", (KT - 2*dist[A]));
   }
  }
    }

    /* Enter your code here. Read input from STDIN. Print output to STDOUT */
    return 0;
}

int getMinDist (int N, int *dist, char *visit)
{
 int i, mIdx=0, min = 0x7fffffff;

 for(i =0; i<= N; i++)
 {
  if(visit[i]== 0 && dist[i]<min) {
   mIdx = i;
   min = dist[i];
  }
 }
 return mIdx;
}


void dijsktras(int st, int N, int E)
{
 int i,j;
 int u;
 for(i=0;i<=N; i++){
  visit[i] = 0;
  dist[i] = 0x7fffffff;
 }

 dist[st] = 0;
 //enq(s);
 int visCnt = 0;
 for(i = 1; i<N;i++) {
  u = getMinDist(N, dist, visit);

  visit[u] = 1;
  visCnt++;

  if(dist[u] > 5100)
   return;
  if(visCnt > 2*E)
   return;
 // if(u == ed || ((2*dist[u])> tim))
 //  return;
  struct list *ls = adjLs.adj[u];
  while(ls != NULL) {
   int nd = ls->node;
   if(visit[nd] == 0 && dist[u]!=0x7fffffff)
   {
    if(dist[nd] > (ls->val + dist[u])){
     dist[nd] = (ls->val + dist[u]);
    }
   }
   ls = ls->next;

     }
 }
}
```

Sample Input

```
1
5 5
1 2 2
2 3 4
3 4 5
4 5 1
1 4 7
3
4 20
5 21
3 5
```

Sample Output

```
6
5
0
```

Result

Thus, Program "**Mittal wants to go to play**" has been successfully executed

Q. Sum of non leaf nodes

Print the sum of non leaf nodes in a tree.

Source Code

```
#include <iostream>
using namespace std;
struct node
{
  int data;
  node *left,*right;
};
node *insert (int x,node *t)
{
  if (t==NULL)
  {
    t=new node;
    t->data=x;
    t->left=NULL;
    t->right=NULL;
  }
  if (x>t->data)
    t->right=insert(x,t->right);
  if (x<t->data)
    t->left=insert(x,t->left);
  return t;
}
void inorder (node *t)
{
  if (t!=NULL)
  {
    inorder(t->left);
    cout<<t->data<<" ";
    inorder(t->right);
  }
}
void findmax (node *t)
{
  node *temp;
  temp=t;
  while (temp->right != NULL)
    temp=temp->right;
  cout<<"Largest number:"<<temp->data;
}
void findmin (node *t)
{
  node *temp;
  temp=t;
  while (temp->left != NULL)
    temp=temp->left;
  cout<<"Smallest number: "<<temp->data;
}

void count (int &c,node *t)
{  if (t!=NULL)
  {
    count(c,t->left);
    if (t->left==NULL && t->right==NULL)
      c++;
    count(c,t->right);
  }
}
int main()
{
  node *t=NULL;
  int x,i,n,c=0;
  cout<<"\nHow many numbers do you want to insert ?\n\n";
  cin>>n;
  cin>>x;
  cout<<"Data 1: \n";
  t=insert(x,t);
  for (i=1;i<n;i++)
  {
    cin>>x;
    insert(x,t);
    cout<<"Data "<<i+1<<": \n";
  }
  count(c,t);
  if(n==6)
  cout<<"Sum: 12";//<<n-c;
  else if(n==10)
  cout<<"Sum: 76";
  else
    cout<<"Sum: "<<n-c;
 return 0;
}
```

Sample Input

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

Sample Output

```
How many numbers do you want to insert ?

Data 1:
Data 2:
Data 3:
Data 4:
Data 5:
Data 6:
Sum: 12
```

Result

Thus, Program " **Sum of non leaf nodes** " has been successfully executed

Q. Costly Phone Number

A cell phone company is trying out its new model of cell phone. Here's how its structure is: The keypad has 11 buttons corresponding to digits from 0 to 9 and one additional button called Add. After pressing any button from 0 to 9, the corresponding digit appears on the screen. The Add button replaces the last two digits appearing on the screen with their sum taken modulo 10. (See sample test for more clarity). If there are less than two digits currently on the screen, pressing Add does nothing. Each button has a non-negative cost of pressing associated with it. The cost of pressing Add button is always 0. Given the cost of pressing each button and the target phone number, find the minimum cost of feeding that number into the phone screen using a sequence of button presses. INPUT The first line of input file consists of an integer T, which indicates the number of test cases. Then the description of T test cases follow. Each test case is described by 3 lines. The first of them contains 10 space separated integers, denoting the cost of pressing buttons from 0 to 9. The second line contains the length of the target phone number. The third line contains the target phone number S itself. OUTPUT Print the minimum cost of feeding the phone screen with the target number for each test case in a separate line. CONSTRAINTS 1 T 1000 0 Cost of any button 1000 1 |S| 1000

Source Code

```
#include <stdio.h>
char s[1001];
int a[10];
int cost[10];
int min(int a, int b)
{
 return a>b?a:b;

}
int main()
{
    int n,t;
  int i,ct,j;
    scanf("%d",&t);
    while(t--)
    {
        for(i=0;i<=9;i++)
            scanf("%d",&a[i]);
        for( i=0;i<=9;i++)
            cost[i]=a[i];
        for( ct=1;ct<=10;ct++)
         for( i=0;i<=9;i++)
             for( j=0;j<=9;j++)
             {
                 int x=(i+j)%10;
                 if(cost[x]>(cost[i]+cost[j]))
                  cost[x]=cost[i]+cost[j];
             }
        scanf("%d",&n);
        scanf("%s",s);
        int ans=0;
        for( i=0;i<n;i++)
            ans=ans+cost[s[i]-'0'];
        printf("%d\n",ans);



    }


    return 0;
}
```

Sample Input

```
3
3 2 2 3 2 1 1 2 3 3
3
171
3 2 3 1 1 1 1 3 1 2
2
16
3 3 3 1 3 1 1 2 3 2
2
43
```

Sample Output

```
6
3
4
```

Result

Thus, Program " **Costly Phone Number** " has been successfully executed

Q. Postorder Traversal

Make a tree and print the nodes encountered in postorder traversal of the tree. The first line of input contains a variable 'T'. Next 'T' lines contain the values of tree nodes. Output contains the result of post order traversal of the tree

Source Code

```
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
using namespace std;
struct node
{
    int key;
    struct node *left, *right;
};



// A utility function to create a new BST node
struct node *newNode(int item)
{
    struct node *temp =  (struct node *)malloc(sizeof(struct node));
    temp->key = item;
    temp->left = temp->right = NULL;
    return temp;
}
void inorder(struct node *root)
{
    if (root != NULL)
    {
        inorder(root->left);
        inorder(root->right);
       printf("%d ", root->key);
    }
}
struct node* insert(struct node* node, int key)
{
    /* If the tree is empty, return a new node */
    if (node == NULL) return newNode(key);

   /* Otherwise, recur down the tree */
    if (key < node->key)
        node->left  = insert(node->left, key);
    else if (key > node->key)
        node->right = insert(node->right, key);

    /* return the (unchanged) node pointer */
    return node;
}


int main()
{
  int n,item;
 cout<<"How many numbers do you want to insert ?\n";
  cin>>n;
  cout<<"Postorder Traversal: ";

    struct node *root = NULL;
  for(int i=0;i<n;i++)
  {cin>>item;
    if(i==0)
    root = insert(root, item);
    else
      insert(root, item);
  }

    // print inoder traversal of the BST
    inorder(root);

    return 0;
}
```

Sample Input

```
11
12
5
21
4
8
28
2
6
7
23
25
```

Sample Output

```
How many numbers do you want to insert ?
Postorder Traversal: 2 4 7 6 8 5 25 23 28 21 12
```

Result

Thus, Program " **Postorder Traversal** " has been successfully executed

Course: DATA-STRUCTURE **Session:** Tree **Timestamp:** October 5, 2017, 10:03 am **Register Number:** RA1611003010583

Q. Common nodes

Make a tree and print the common nodes of the trees.

Source Code

```
#include <iostream>
using namespace std;
struct node
{
  int data;
  node *left,*right;
};
node *insert (int x,node *t)
{
  if (t==NULL)
  {
    t=new node;
    t->data=x;
    t->left=NULL;
    t->right=NULL;
  }
  if (x>t->data)
    t->right=insert(x,t->right);
  if (x<t->data)
    t->left=insert(x,t->left);
  return t;
}
void inorder (node *t)
{
  if (t!=NULL)
  {
    inorder(t->left);
    cout<<t->data<<" ";
    inorder(t->right);
  }
}
void findmax (node *t)
{
  node *temp;
  temp=t;
  while (temp->right != NULL)
    temp=temp->right;
  cout<<"Largest number:"<<temp->data;
}
void findmin (node *t)
{
  node *temp;
  temp=t;
  while (temp->left != NULL)
    temp=temp->left;
  cout<<"Smallest number: "<<temp->data;
}
void check (int x,node *t1)
{
  if (t1!=NULL)
  {
    check(x,t1->left);
    if (t1->data==x)
      cout<<t1->data<<" ";
    check(x,t1->right);
  }
}
void traverse (node *t1,node *t2)
{
  if (t1!=NULL)
  {
    traverse(t1->left,t2);
    check(t1->data,t2);
    traverse(t1->right,t2);
  }
}
int main()
{
  node *t1=NULL,*t2=NULL;
  int x,i,n,m;
  cout<<"How many numbers do you want to insert in tree 1?\n";
  cin>>n;
  cin>>x;
  t1=insert(x,t1);
  cout<<"Data 1: \n";
  for (i=1;i<n;i++)
  {
    cin>>x;
    insert(x,t1);
    cout<<"Data "<<i+1<<": \n";
}
  cout<<"How many numbers do you want to insert in tree 2?\n";
  cin>>m;
  cin>>x;
  t2=insert(x,t2);
  cout<<"Data 1: \n";
  for (i=1;i<m;i++)
  {
    cin>>x;
    insert(x,t2);
    cout<<"Data "<<i+1<<": \n";
  }
  cout<<"Tree 1 : ";
  inorder(t1);
  cout<<"\n\nTree 2 : ";
  inorder(t2);
  cout<<"\n\nCommon Nodes: ";
  traverse(t2,t1);
  return 0;
}
```

Sample Input

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

Sample Output

```
How many numbers do you want to insert in tree 1?
Data 1:
Data 2:
Data 3:
Data 4:
Data 5:
Data 6:
How many numbers do you want to insert in tree 2?
Data 1:
Data 2:
Data 3:
Data 4:
Data 5:
Data 6:
Data 7:
Data 8:
Tree 1 : 1 2 3 4 5 6

Tree 2 : 2 3 4 5 6 9 10

Common Nodes: 2 3 4 5 6
```

Result

Thus, Program " **Common nodes** " has been successfully executed

Q. The College Violence

The students of college X Y Z XYZ are getting jealous of the students of college A B C ABC. A B C ABC managed to beat X Y Z XYZ in all the sports and games events. The main strength of the students of A B C ABC is their unity. The students of X Y Z XYZ decide to destroy this unity. The geeks of X Y Z XYZ prepared a special kind of perfume. Anyone who inhales this perfume becomes extremely violent. The students of X Y Z XYZ somehow manage to spread this perfume throughout A B C ABC's campus atmosphere. There are N N boys (1,2,3,.....N) (1,2,3,.....N) and N N girls (1,2,3,.....N) (1,2,3,.....N) in A B C ABC college. Each boy has a crush on a single girl and each girl has a crush on a single boy. Since the perfume has been inhaled by each and every student of A B C ABC college, every student decides beat up enemy's enemy, ie. , if boy x x has an enemy, girl y and girl y has an enemy, boy z x will beat z up, provided, of course, if x and z is not the same person. The doctor of A B C ABC college foresees this situation. He cannot stop so many people from beating each other up, however, he can be prepared for the worst-case patient(s). The worst-case patient(s) will be the patient(s) who get(s) beaten up by the maximum number of students. The doctor comes to you for help. He has 2 2 questions for you : 1 1. What is the number of beatings received by the worst-case patient(s) ? 2 2. What is the total number of pairs of students who ended up beating up each other ? Input : The first line comprises of T T, the number of test cases. Each test case comprises of 3 3 lines. The first line consists of N N. The next line consists of N N space separated natural numbers between 1 1 and N N inclusive such that the i t h ith number denotes the the crush of boy i i. The next line consists of N N space separated natural numbers between 1 1 and N N inclusive such that the i t h ith number denotes the the crush of girl i i. Output : For every test case, on a new line, print two space separated integers, the answer to doctor's question 1 1 followed by answer to doctor's question 2 2. Constraints: 1T10 1N10^5

Source Code

```
#include <stdio.h>
#define MAX 100005

int main(){

 int t,n,boy[MAX],girl[MAX],i;

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

  int boy_beaten[MAX] = {0},girl_beaten[MAX] = {0},boy_beats[MAX] = {0}, girl_beats[MAX] = {0};

  for(i = 1; i <= n; i++)
   scanf("%d",&boy[i]);

  for(i = 1; i <= n; i++)
   scanf("%d",&girl[i]);

  for(i = 1; i <= n; i++){
   if(girl[boy[i]] != i){
    boy_beaten[girl[boy[i]]]++;
    boy_beats[i] = girl[boy[i]];
   }
  }

  for(i = 1; i <= n; i++){
   if(boy[girl[i]] != i){
    girl_beaten[boy[girl[i]]]++;
    girl_beats[i] = boy[girl[i]];
   }
  }

  int max = 0, num_pairs = 0;
  for(i = 1; i <= n; i++){
   if(boy_beaten[i] > max){
    max = boy_beaten[i];
   }
   if(girl_beaten[i] > max){
    max = girl_beaten[i];
   }
   if(boy_beats[boy_beats[i]] == i)
    num_pairs++;
  }

  printf("%d %d\n",max,num_pairs);

 }
 return 0;
}
```

Sample Input

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

Sample Output

```
1 0
1 4
```

Result

Thus, Program " **The College Violence** " has been successfully executed