Course: C **Session:** Functions **Timestamp:** 2021-1-9 19:19:49 **Register Number:** RA2031241010094

Q. Pythagorean Triples

Katya studies in a fifth grade. Recently her class studied right triangles and the Pythagorean theorem. It appeared, that there are triples of positive integers such that you can construct a right triangle with segments of lengths corresponding to triple. Such triples are called Pythagorean triples.
For example, triples (3,4,5), (5,12,13) and (6,8,10) are Pythagorean triples.Here Katya wondered if she can specify the length of some side of right triangle and find any Pythagorean triple corresponding to such length? Note that the side which length is specified can be a cathetus as well as hypotenuse.
Katya had no problems with completing this task. Will you do the same?

Input
The only line of the input contains single integer n (1<=n<=109) the length of some side of a right triangle.

Output
Print two integers m and k (1<=m,k<=1018), such that n, m and k form a Pythagorean triple, in the only line.In case if there is no any Pythagorean triple containing integer n, print -1 in the only line. If there are many answers, print any of them.

Source Code

```
#include <stdio.h>
int main()
{
  int i,n,m,k;
  scanf("%d",&n);
  if(n<3)
  {
    printf("-1");
  }
  else
  {
  if(n%2==0)
     {
       m=n*n/4-1;
       k=n*n/4+1;
     }
     else
     {
       m=n*n/2;
       k=n*n/2+1;
     }
     printf("%d %d",m,k);
  }
       return 0;
}
```

Sample Input

3

Sample Output

4 5

Result

Thus, Program " **Pythagorean Triples** " has been successfully executed

Course: C **Session:** Functions **Timestamp:** 2021-1-9 19:20:35 **Register Number:** RA2031241010094

Q. Prime Numbers Between Two Integers

Program to Display Prime Numbers Between Intervals Using Function
Note:
Include upper and lower limits in the output.

Source Code

```
#include <stdio.h>
int prime(int n);
int main()
{
  int n1,n2,i,f;
  scanf("%d\n%d",&n1,&n2);
  for(i=n1;i<=n2;i++)
  {
    f=prime(i);
    if(f==0)
      printf("%d\n",i);
  }
return 0;
}
int prime(int n)
{
  int j,f=0;
  for(j=2;j<=n/2;j++)
  {
    if(n%j==0)
    {
      f=1;
    }
  }
  return f;
}
```

Sample Input

```
13
29
```

Sample Output

```
13
17
19
23
29
```

Result

Thus, Program " **Prime Numbers Between Two Integers** " has been successfully executed

Course: C **Session:** Functions **Timestamp:** 2021-1-9 19:21:04 **Register Number:** RA2031241010094

Q. Devu and friendship testing

Devu has n weird friends. Its his birthday today, so they thought that this is the best occasion for testing their friendship with him. They put up conditions before Devu that they will break the friendship unless he gives them a grand party on their chosen day. Formally, ith friend will break his friendship if he does not receive a grand party on dith day.

Devu despite being as rich as Gatsby, is quite frugal and can give at most one grand party daily. Also, he wants to invite only one person in a party. So he just wonders what is the maximum number of friendships he can save. Please help Devu in this tough task !!

Input

The first line of the input contains an integer T denoting the number of test cases. The description of T test cases follows.
First line will contain a single integer denoting n.
Second line will contain n space separated integers where ith integer corresponds to the day dith as given in the problem.
Output

Print a single line corresponding to the answer of the problem.

Constraints

1<= T <= 104
1 <= n<= 50
1 <= di<= 100

Explanation
Example case 1. Devu can give party to second friend on day 2 and first friend on day 3, so he can save both his friendships.

Example case 2. Both the friends want a party on day 1, and as the Devu can not afford more than one party a day, so he can save only one of the friendships, so answer is 1.

Source Code

```
#include <stdio.h>
int main()
{
  int i,j,n,n1,sum[100],p;
  scanf("%d",&n1);
  for(i=0;i<n1;i++)

  {
    scanf("%d",&n);
    int a;
    for(j=0;j<100;j++)
      sum[j]=0;
    for(j=0;j<n;j++)
    {
      scanf("%d",&a);
      sum[a-1]=1;
    }
    p=0;
    for(j=0;j<100;j++)
    {
      if(sum[j]==1)
        p=p+1;
    }
    printf("%d\n",p);
  }
 return 0;
}
```

Sample Input

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

Sample Output

```
2
1
```

Result

Thus, Program " **Devu and friendship testing** " has been successfully executed

Course: C **Session:** Functions **Timestamp:** 2021-1-9 19:21:43 **Register Number:** RA2031241010094

Q. Caravan

Most problems on SRM University eLab highlight eLab love for programming but little is known about his love for racing sports. He is an avid Formula 1 fan.

He went to watch this years Indian Grand Prix at New Delhi. He noticed that one segment of the circuit was a long straight road. It was impossible for a car to overtake other cars on this segment. Therefore, a car had to lower down its speed if there was a slower car in front of it. While watching the race, Chef started to wonder how many cars were moving at their maximum speed.

Formally, youre given the maximum speed of N cars in the order they entered the long straight segment of the circuit. Each car prefers to move at its maximum speed. If thats not possible because of the front car being slow, it might have to lower its speed. It still moves at the fastest possible speed while avoiding any collisions. For the purpose of this problem, you can assume that the straight segment is infinitely long.

Count the number of cars which were moving at their maximum speed on the straight segment.

Input

The first line of the input contains a single integer T denoting the number of test cases to follow. Description of each test case contains 2 lines. The first of these lines contain a single integer N, the number of cars. The second line contains N space separated integers, denoting the maximum speed of the cars in the order they entered the long straight segment.

Output

For each test case, output a single line containing the number of cars which were moving at their maximum speed on the segment.

Source Code

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

 return 0;
}
```

Sample Input

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

Sample Output

```
1
2
2
```

Result

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

Course: C **Session:** Functions **Timestamp:** 2021-1-9 19:35:48 **Register Number:** RA2031241010094

Q. Recursion 7 : Sum of elements in Array

Write a program to compute the sum of elements in an array using recursion.

Input and Output Format:

Input consists of n+1 integers.

Refer sample input and output for formatting specifications.

All text in bold corresponds to input and the rest corresponds to output.

Source Code

```
#include <stdio.h>
int main()
{
  int i,n,sum=0;
    scanf("%d",&n);
  int a[n];
  for(i=0;i<n;i++)
    scanf("%d ",&a[i]);
  for(i=0;i<n;i++)
  {
    sum=sum+a[i];
  }
 printf("%d",sum);
 return 0;
}
```

Sample Input

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

Sample Output

```
15
```

Result

Thus, Program " **Recursion 7 : Sum of elements in Array** " has been successfully executed

Course: C **Session:** Functions **Timestamp:** 2021-1-9 19:37:44 **Register Number:** RA2031241010094

Q. Sum of array elements using Functions

Write a C program to find the sum of the elements in an array.

Input Format:

Input consists of n+1 integers. The first integer corresponds to n , the size of the array. The next n integers correspond to the elements in the array. Assume that the maximum value of n is 15.

Output Format:

Refer sample output for details.

Source Code

```
#include <stdio.h>
int main()
{
  int i,n,sum=0;
  scanf("%d",&n);
  int a[n];
  for(i=0;i<n;i++)
    scanf("%d",&a[i]);
  for(i=0;i<n;i++)
  {
    sum=sum+a[i];
  }
  printf("%d",sum);
 return 0;
}
```

Sample Input

```
5
2 3 6 8 1
```

Sample Output

```
20
```

Result

Thus, Program " **Sum of array elements using Functions** " has been successfully executed

Course: C **Session:** Functions **Timestamp:** 2021-1-9 19:39:54 **Register Number:** RA2031241010094

Q. Turbo Sort

Given the list of numbers, you are to sort them in non decreasing order.
Input

t ? the number of numbers in list, then t lines follow [t <= 10^6].
Each line contains one integer: N [0 <= N <= 10^6]
Output

Output given numbers in non decreasing order.

Source Code

```
#include <stdio.h>
int main()
{
  int i,j,temp,t,a[1000000];
  scanf("%d",&t);
  for(i=0;i<t;i++)
  {
    scanf("%d",&a[i]);
  }
  for(i=0;i<t;i++)
  {
    for(j=i+1;j<t;j++)
    {
      if(a[i]>a[j])
      {
        temp=a[i];
        a[i]=a[j];
        a[j]=temp;
      }
    }
  }
  for(i=0;i<t;i++)
  {
    printf("%d\n",a[i]);
  }
  return 0;
}
```

Sample Input

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

Sample Output

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

Result

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

Course: C **Session:** Functions **Timestamp:** 2021-1-9 19:40:56 **Register Number:** RA2031241010094

Q. Random order

Petr, Nikita G. and Nikita are the most influential music critics in Saint-Petersburg. They have recently downloaded their favorite band's new album and going to listen to it. Nikita claims that the songs of entire album should be listened strictly in the same order as they are given, because there is the secret message from the author in the songs' order. Petr, being chaotic, does not think so, hence he loves listening to songs in a random order. Petr is pretty good in convincing other people, so after a two-hours discussion Nikita accepted listening in random order(the discussion's duration was like three times longer than the album's one). In this context random order means following: There are N songs in the album. In the very beginning random song is chosen(here and further "random song" means that every song has equal probability to be chosen). After some song is over the next one is chosen randomly and independently of what have been played before. Nikita G., being the only one who is not going to drop out from the university, wonders, what is the expected number of songs guys have to listen to until every song is played at least once.

Source Code

```
#include <stdio.h>
int main()
{
  int n,t,i;
  double x;
  scanf("%d",&t);
  while(t--)
  {
    scanf("%d",&n);
    x=1;
    for(i=2;i<=n;i++)
      x=x+(1.0)/i;
    printf("%1f\n",n*x);
  }
 return 0;
}
```

Sample Input

```
3
1
2
3
```

Sample Output

```
1.000000
3.000000
5.500000
```

Result

Thus, Program " **Random order** " has been successfully executed