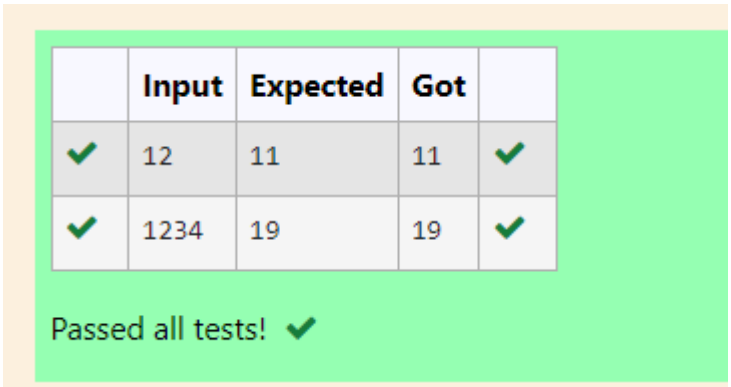
```
    count++;
//print(rev);
count++;
printf("%d",count);
}

int main(){
    int n;
    scanf("%d",&n);
    reverse(n);
}
```

RESULT:



	Input	Expected	Got	
✔	12	11	11	✔
✔	1234	19	19	✔

Passed all tests! ✔

1-Number of Zeros in a Given Array

Problem Statement

Given an array of 1s and 0s this has all 1s first followed by all 0s. Aim is to find the number of 0s. Write a program using Divide and Conquer to Count the number of zeroes in the given array.

Input Format

First Line Contains Integer m – Size of array

Next m lines Contains m numbers – Elements of an array

Output Format

First Line Contains Integer – Number of zeroes present in the given array.

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

int divide(int arr[],int lesser,int greater)
{
   if(arr[greater]==1)
   {
      return 0;
   }
   if(arr[lesser]==0)
   {
      return greater-lesser+1;
   }
   int mid=(lesser+greater)/2;
   int left=divide(arr,lesser,mid);
   int right=divide(arr,mid+1,greater);
   return left+right;
}
int main()
{
   int size;
   scanf("%d",&size);
   int arr[size];
   for(int i=0;i<size;i++)
   {
      scanf("%d",&arr[i]);
   }
   int count=divide(arr,0,size-1);
   printf("%d\n",count);
}
```

	Input	Expected	Got	
✔	5 1 1 1 0 0	2	2	✔
✔	10 1 1 1 1 1 1 1 1 1 1	0	0	✔

2-Majority Element

Given an array `nums` of size `n`, return *the majority element*.

The majority element is the element that appears more than `⌊n / 2⌋` times. You may assume that the majority element always exists in the array.

Example 1:

```
Input: nums = [3,2,3]
Output: 3
```

Example 2:

```
Input: nums = [2,2,1,1,1,2,2]
Output: 2
```

Constraints:

- `n == nums.length`
- `1 <= n <= 5 * 10^4`
- `-2^31 <= nums[i] <= 2^31 - 1`

program:

```
#include<stdio.h>
int divide(int a[],int x,int y,int z)
{
   if(x==y)
   {
      return a[x];
   }
int center=(x+y)/2;
int sinister=divide(a,x,center,z);
int opp=divide(a,center+1,y,z);
int lcount=0,rcount=0;
for(int i=0;i<z;i++)
{if(a[i]==sinister)
lcount++;
if(a[i]==opp)
rcount++;
```

```
}
if(lcount>(z/2))
return sinister;else
      return opp;
}
int main()
{
   int length;
   scanf("%d",&length);
   int arr[length];
   for(int i=0;i<length;i++)
   {
      scanf("%d",&arr[i]);
   }
   int low=0,high=length-1;
   int majority=divide(arr,low,high,length);
   printf("%d",majority);
}
```

	Input	Expected	Got	
✔	3 3 2 3	3	3	✔

Passed all tests! ✔

Problem Statement:
Given a sorted array and a value x, the floor of x is the largest element in array smaller than or equal to x. Write divide and conquer algorithm to find floor of x.

Input Format

First Line Contains Integer n – Size of array
Next n lines Contains n numbers – Elements of an array
Last Line Contains Integer x – Value for x

Output Format

First Line Contains Integer – Floor value for x

program:

```
#include<stdio.h>
int findmax(int arr[],int high,int low,int x);
int main(){
   int size;
   scanf("%d",&size);
   int arr[size];
   for(int i=0;i<size;i++){
      scanf("%d",&arr[i]);
   }
   int t;
   scanf("%d",&t);
   int mid=(0+(size-1)/2);
   int left=findmax(arr,0,mid,t);
   int right=findmax(arr,mid+1,size-1,t);
   if(left>right){
      printf("%d",left);
   }
   else{
      printf("%d",right);
   }
```

```
}
int findmax(int arr[],int low,int high,int x){
   int element=0;
   for(int i=0;i<high;i++){
      if(arr[i]<=x){
         if(arr[i]>element){
            element=arr[i];
         }
      }
   }
   return element;
}
```

	Input	Expected	Got	
✔	3 1 3 5 4	1	1	✔
✔	10 1 4 6 8 12 14 15 20 21 25 1	1	1	✔
✔	10 1 2 3 5 11 14 16 24 28 29 0	0	0	✔
✔	10 0 2 3 7 13 14 15 20 24 25 10	1	1	✔

Passed all tests! ✔

4-Two Elements sum to x

Problem Statement:
Given a sorted array of integers say arr[] and a number x. Write a recursive program using divide and conquer strategy to check if there exist two elements in the array whose sum = x. If there exist such two elements then return the numbers, otherwise print as "No".
Note: Write a Divide and Conquer Solution
Input Format
First Line Contains Integer n – Size of array
Next n lines Contains n numbers – Elements of an array
Last Line Contains Integer x – Sum Value
Output Format
First Line Contains Integer – Element1
Second Line Contains Integer – Element2 (Element 1 and Elements 2 together sums to value "x")

Answer: (penalty regime: 0 %)

program:

```
#include <stdio.h>

int findpair(int arr[],int l,int r,int x){

if(l>=r){

   return 0;

}

int i=l;

int j=r;

while(i<j){

   int sum=arr[i]+arr[j];

   if(sum==x){

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

      return 1;

   }else if(sum<x){

      i++;

   }else{

      j--;

   }

}
```

```
int m=(l+r)/2;
return findpair(arr,l,m,x)||findpair(arr,m+1,r,x);
}
int main(){
int n;
scanf("%d",&n);
int arr[n];
for(int i=0;i<n;i++){
   scanf("%d",&arr[i]);
}
int x;
scanf("%d",&x);
if(!findpair(arr,0,n-1,x)){
   printf("No\n");
}
}
```

	Input	Expected	Got	
✔	4 2 4 8 10 14	4 10	4 10	✔
✔	5 2 4 6 8 10 100	No	No	✔

Passed all tests! ✔

5-Implementation of Quick Sort

Write a Program to Implement the Quick Sort Algorithm

Input Format:
The first line contains the no of elements in the list-n
The next n lines contain the elements.

Output:
Sorted list of elements

For example:

Input	Result
5 67 34 12 98 78	12 34 67 78 98

program:

```
#include <stdio.h>
int partition(int a[],int l,int h){
int piv=a[l];
int i=l,j=h,t;
while(i<j){
  while(a[i]<=piv && i<h){
     i++;
  }
  while(a[j]>piv && j>l){
     j--;
  }
  if(i<j){
     t=a[i];
     a[i]=a[j];
     a[j]=t;
  }
```

```
}
t=a[l];
a[l]=a[j];
a[j]=t;
return j;
}
void Quicksort(int a[],int l,int h){
if(l<h){
  int par=partition(a,l,h);
  Quicksort(a,l,par-1);
  Quicksort(a,par+1,h);
}
}
int main(){
int n;
scanf("%d",&n);
int a[n];
for(int i=0;i<n;i++){
  scanf("%d",&a[i]);
}
Quicksort(a,0,n-1);
for(int i=0;i<n;i++){
  printf("%d ",a[i]);
}
}
```

	Input	Expected	Got
✔	5 67 34 12 98 78	12 34 67 78 98	12 34 67 78 98
✔	10 1 56 78 90 32 56 11 10 90 114	1 10 11 32 56 56 78 90 90 114	1 10 11 32 56 56 78 90 90
✔	12 9 8 7 6 5 4 3 2 1 10 11 90	1 2 3 4 5 6 7 8 9 10 11 90	1 2 3 4 5 6 7 8 9 10 11 9

1-DP-Playing with Numbers

Playing with Numbers:

Ram and Sita are playing with numbers by giving puzzles to each other. Now it was Ram term, so he gave Sita a positive integer 'n' and two numbers 1 and 3. He asked her to find the possible ways by which the number n can be represented using 1 and 3.Write any efficient algorithm to find the possible ways.

Example 1:

Input: *6*
Output:*6*
Explanation: *There are 6 ways to 6 represent number with 1 and 3*

1+1+1+1+1+1
3+3
1+1+1+3
1+1+3+1
1+3+1+1
3+1+1+1

Input Format

First Line contains the number n

Output Format

Print: The number of possible ways 'n' can be represented using 1 and 3

```
#include <stdio.h>


long long int countWays(int n) {
```