DEC 29:

INSERTION SORT:-

The task is to complete the **insert()** function which is used to implement Insertion Sort.

Example 1:

```
Input:
N = 5
arr[] = { 4, 1, 3, 9, 7}
Output:
1 3 4 7 9
```

Example 2:

```
Input:
N = 10
arr[] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1}
Output:
1 2 3 4 5 6 7 8 9 10
```

Your Task:
You don't have to read input or print anything. Your task is to complete the function **insert()** and **insertionSort()** where **insert()** takes the array, it's size and an index i and **insertionSort()** uses insert function to sort the array in ascending order using insertion sort algorithm.

Expected Time Complexity: O(N*N).
Expected Auxiliary Space: O(1).

Constraints:

1 <= N <= 1000

1 <= arr[i] <= 1000

```
//{ Driver Code Starts
// C program for insertion sort
#include <stdio.h>
#include <math.h>

/* Function to print an array */
void printArray(int arr[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}


// } Driver Code Ends
class Solution
{
    public:
    void insert(int arr[], int i)
    {
        //code here
    }
     public:
    //Function to sort the array using insertion sort algorithm.
    void insertionSort(int arr[], int n)
    {
        //code here
        iterative method

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

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

                if(arr[i]<arr[j]){

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



            }



        }
```

```
        //Reccursive method



        if(n==1){
            return ;
        }

        for(int i=0;i<n-1;i++){

            if(arr[i]>arr[i+1]){
                 int temp=arr[i];
                 arr[i]=arr[i+1];
                 arr[i+1]=temp;
            }
        }


        insertionSort(arr,n-1);









    }
};

//{ Driver Code Starts.
int main()
{
    int arr[1000],n,T,i;

    scanf("%d",&T);

    while(T--){

    scanf("%d",&n);

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

    Solution ob;
    ob.insertionSort(arr, n);
    printArray(arr, n);
    }
    return 0;
}

// } Driver Code Ends
```

Bubble Sort:-

Given an Integer **N** and a list **arr**. Sort the array using bubble sort algorithm.

Example 1:

```
Input:
N = 5
arr[] = {4, 1, 3, 9, 7}
Output:
1 3 4 7 9
```

Example 2:

```
Input:
N = 10
arr[] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1}
Output:
1 2 3 4 5 6 7 8 9 10
```

Your Task:

You don't have to read input or print anything. Your task is to complete the function **bubblesort()** which takes the array and it's size as input and sorts the array using bubble sort algorithm.

Expected Time Complexity: O(N^2).

Expected Auxiliary Space: O(1).

Constraints:

1 <= N <= 10^3

1 <= arr[i] <= 10^3

```
//{ Driver Code Starts
//Initial Template for C++

// C program for implementation of Bubble sort
#include <stdio.h>
```

```cpp
#include <bits/stdc++.h>
using namespace std;

// swapping the elements
void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}


// } Driver Code Ends
//User function Template for C++

class Solution
{
    public:
    //Function to sort the array using bubble sort algorithm.
    void bubbleSort(int arr[], int n)
    {
        // Your code here

      // Iterative method

        for(int i=0;i<n-1;i++){

            for(int j=0;j<(n-1)-i;j++){

                if(arr[j]>arr[j+1]){

                      swap(arr[j],arr[j+1]);

                }


            }



        }


        //Recursive method

        if(n==1){
            return;
        }

        for(int i=0;i<n-1;i++){

            if(arr[i]>arr[i+1]){
```

```
                swap(arr[i],arr[i+1]);
            }
        }

        bubbleSort(arr,n-1);

    }
};


//{ Driver Code Starts.

/* Function to print an array */
void printArray(int arr[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

// Driver program to test above functions
int main()
{
    int arr[1000],n,T,i;

    scanf("%d",&T);

    while(T--){

    scanf("%d",&n);

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

    Solution ob;

    ob.bubbleSort(arr, n);
    printArray(arr, n);
    }
    return 0;;
}
// } Driver Code Ends
```

Merge Sort :-

Given an array arr[], its starting position l and its ending position r. Sort the array using merge sort algorithm.

Example 1:

```
Input:
N = 5
arr[] = {4 1 3 9 7}
Output:
1 3 4 7 9
```

Example 2:

```
Input:
N = 10
arr[] = {10 9 8 7 6 5 4 3 2 1}
Output:
1 2 3 4 5 6 7 8 9 10
```

Your Task:
You don't need to take the input or print anything. Your task is to complete the function **merge()** which takes arr[], l, m, r as its input parameters and modifies arr[] in-place such that it is sorted from position l to position r, and function **mergeSort()** which uses merge() to sort the array in ascending order using merge sort algorithm.

Expected Time Complexity: O(nlogn)
Expected Auxiliary Space: O(n)

Constraints:
$1 <= N <= 10^5$
$1 <= arr[i] <= 10^3$

CODE SECTION:-

```cpp
//{ Driver Code Starts
#include <stdio.h>
#include <bits/stdc++.h>
using namespace std;



/* Function to print an array */
void printArray(int arr[], int size)
{
```

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


// } Driver Code Ends
class Solution
{
    public:
    void merge(int arr[], int l, int m, int r)
    {
        // Your code here
        int i = l, j = m+1, ci = 0;
        int *temp = new int[r - l + 1];
        while(i <= m && j <= r){
            if(arr[i] <= arr[j]){
                temp[ci++] = arr[i++];
            }else {
                temp[ci++] = arr[j++];
            }
        }
        while(i <= m ){
            temp[ci++] = arr[i++];
        }
        while(j <= r){
            temp[ci++] = arr[j++];
        }
        for(int k = 0; k < r-l+1; k++){
            arr[l + k] = temp[k];
        }

    }

    void mergeSort(int arr[], int l, int r)
    {
        //code here
        if(r > l){
            int m = l + (r - l) /2;
            mergeSort(arr, l, m );
            mergeSort(arr, m + 1, r);
            merge(arr, l, m, r);
        }



    }
};

//{ Driver Code Starts.
```

```
int main()
{
    int n,T,i;

    scanf("%d",&T);

    while(T--){

    scanf("%d",&n);
    int arr[n+1];
    for(i=0;i<n;i++)
      scanf("%d",&arr[i]);

    Solution ob;
    ob.mergeSort(arr, 0, n-1);
    printArray(arr, n);
    }
    return 0;
}
// } Driver Code Ends
```

Quick Sort:-

Quick Sort is a Divide and Conquer algorithm. It picks an element as a pivot and partitions the given array around the picked pivot.
Given an array arr[], its starting position is low (the index of the array) and its ending position is high(the index of the array).

Note: The **low** and **high** are inclusive.

Implement the partition() and quickSort() functions to sort the array.

Example 1:

```
Input:
N = 5
arr[] = { 4, 1, 3, 9, 7}
Output:
1 3 4 7 9
```

Example 2:

```
Input:
N = 9
```

```
arr[] = { 2, 1, 6, 10, 4, 1, 3, 9, 7}
Output:
1 1 2 3 4 6 7 9 10
```

Your Task:
You don't need to read input or print anything. Your task is to complete the functions **partition()** and **quickSort()** which takes the array arr[], low and high as input parameters and partitions the array. Consider the last element as the pivot such that all the elements less than(or equal to) the pivot lie before it and the elements greater than it lie after the pivot.

Expected Time Complexity: O(N*logN)
Expected Auxiliary Space: O(1)

Constraints:
1 <= N <= 10^3
1 <= arr[i] <= 10^4

CODE SECTION:-

```cpp
//{ Driver Code Starts
#include <stdio.h>
#include <bits/stdc++.h>
using namespace std;

/* Function to print an array */
void printArray(int arr[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

// } Driver Code Ends
class Solution
{
    public:
    //Function to sort an array using quick sort algorithm.
    void quickSort(int arr[], int low, int high)
    {

        // code here
        // if(high==1){
```

```
        //     return ;
        // }

        if(low == high || low>high){
            return ;
        }

        int j;
        if(low<high){

            j=partition(arr,low,high);
            quickSort(arr,low,j-1);
            quickSort(arr,j+1,high);

        }
    }

    public:
    int partition (int a[], int low, int high)
    {
       // Your code here
     int pivot = a[low];
      int l=low,r=high;
      while(l<r)
      {
          while(a[l]<=pivot)l++;
          while(a[r]>pivot)r--;
          if(l<r) swap(a[l],a[r]);
      }
      swap(a[low],a[r]);
      return r;
    }
};
//{ Driver Code Starts.
int main()
{
    int arr[1000],n,T,i;
    scanf("%d",&T);
    while(T--){
    scanf("%d",&n);
    for(i=0;i<n;i++)
      scanf("%d",&arr[i]);
      Solution ob;
    ob.quickSort(arr, 0, n-1);
    printArray(arr, n);
    }
    return 0;
}
// } Driver Code Ends
```