Program:

Merge Sort

```
#include <stdio.h>
void mergeSort(int *, int);
void merge(int *, int, int *, int);
int main(){
    int arr[100], i, size;
    printf("Enter number of elements in the array: ");
    scanf("%d", &size);
    printf("Enter %d numbers \n", size);
    for (i = 0; i < size; i++)
        scanf("%d", &arr[i]);
    mergeSort(arr, size);
    printf("The sorted elements are: ");
    for (i = 0; i < size; i++)
        printf("%d\t", arr[i]);
}
void mergeSort(int *arr, int size){
    int mid;
    if (size == 1)
        return;
    else{
        mid = size / 2;
        mergeSort(arr, mid);
        mergeSort(arr + mid, size - mid);
        merge(arr, mid, arr + mid, size - mid);
    }
}
void merge(int *arr1, int size1, int *arr2, int size2){
    int temp_arr[100], p1, p2, pt;
    p1 = p2 = pt = 0;
    while (p1 < size1 && p2 < size2)
        temp_arr[pt++] = (arr1[p1] < arr2[p2]) ? arr1[p1++] : arr2[p2++];
    while (p1 < size1)
```

```
        temp_arr[pt++] = arr1[p1++];
    for (p1 = 0; p1 < pt; p1++)
        arr1[p1] = temp_arr[p1];
}
```

OUTPUT:

```
➜ root@kali  ~/Documents/Class/PCC-SEM-3/Data-Structures/Expt-9-mergeSort  ./a.out
Enter number of elements in the array: 6
Enter 6 numbers
4 10 15 3 2 1
The sorted elements are: 1      2       3       4       10      15      #
```

Quick Sort

```
#include <stdio.h>
void quickSort(int arr[], int left, int right);
int main(){
    int arr[100], arr2[100], length;
    printf("Enter number of elements in the array:\n");
    scanf("%d", &length);
    printf("Enter %d numbers\n", length);
    for (int i = 0; i < length; i++)
        scanf("%d", &arr[i]);
    quickSort(arr, 0, length - 1);
    printf("The Sorted Values are:\n");
    for (int i = 0; i < length; i++)
        printf("%d \t", arr[i]);
    return 0;
}

void quickSort(int arr[], int left, int right)
{
    int pivot, nxt_pvt, temp_left, temp_right;
    temp_left = left;
    temp_right = right;
    pivot = arr[left];
    while (left < right)
    {
        while (arr[right] >= pivot && left < right)
            right--;
        if (left != right)
        {
            arr[left] = arr[right];
            left++;
        }
        while (arr[left] <= pivot && left < right)
            left++;
```

```
        if (left != right)
        {
            arr[right] = arr[left];
            right--;
        }
    }
    arr[left] = pivot;
    nxt_pvt = left;
    left = temp_left;
    right = temp_right;

    if (left < nxt_pvt)
        quickSort(arr, left, nxt_pvt - 1);
    if (right > nxt_pvt)
        quickSort(arr, nxt_pvt + 1, right);
}
```

OUTPUT: