Practical no: 1

Name: Saloni Vishwakarma
Batch-Roll no: C1-13
Subject: DAA Lab

Aim: To implement Merge sort and Quick sort using C.

1) Merge Sort (Code and Output):

```
#include <stdio.h>
#include <stdlib.h>
void printArray(int A[], int size);
void merge(int arr[], int l,int m, int r);
void mergeSort(int arr[],int l, int r);
int main()
{
   int n;
   printf("\n Enter the number of elements: ");
   scanf("%d",&n);
   int arr[n];
   printf("\n Enter the elements: ");
   for(int i=0;i<n;i++)
   scanf("%d",&arr[i]);
   printf("\n Before Merge Sort:-");
   printf("\n Given array is : ");
   printArray(arr, n);
   mergeSort(arr, 0, n - 1);
   printf("\n After Merge Sort:-");
   printf("\n Sorted array is: ");
   printArray(arr,n);
   return 0;
}

void merge(int arr[], int l,int m, int r)
{
   int i, j, k;
   int n1 = m - l + 1;
   int n2 = r - m;
   int L[n1], R[n2];
```

```
    for (i = 0; i < n1; i++)
    L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
    R[j] = arr[m + 1 + j];
    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2)
    {
        if (L[i] <= R[j])
        {
            arr[k] = L[i];
            i++;
        }
        else
        {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }
    while (j < n2)
    {
        arr[k] = R[j];
        j++;
        k++;
    }
}

void mergeSort(int arr[],int l, int r)
{
    if (l < r)
    {
        int m = l + (r - l) / 2;
        mergeSort(arr, l, m);
```

```
    mergeSort(arr, m + 1, r);
    merge(arr, l, m, r);
  }
}

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

```
Enter the number of elements: 7

Enter the elements: 65 56 23 89 12 7 45

Before Merge Sort:-
Given array is : 65 56 23 89 12 7 45

After Merge Sort:-
Sorted array is: 7 12 23 45 56 65 89


...Program finished with exit code 0
Press ENTER to exit console.
```

2) Quick Sort (Code and Output):

```
#include <stdio.h>
// main function
int main()
{
  int n;
  printf("\n Enter the size of array: ");
  scanf("%d",&n);
  int data[n];
  printf("\n Enter the elements of array: ");
  for(int i=0;i<n;i++)
  scanf("%d",&data[i]);
```

```
    printf("\n Array before Quick Sort: ");
    printArray(data, n);
    // perform quicksort on data
    quickSort(data, 0, n - 1);
    printf("\n\n Array after Quick Sort: ");
    printArray(data, n);
}

// function to swap elements
void swap(int *a, int *b)
{
    int t = *a;
    *a = *b;
    *b = t;
}

// function to find the partition position
int partition(int array[], int low, int high)
{
    int pivot = array[high];   // select the rightmost element as pivot
    int i = (low - 1);    // pointer for greater element

    /* traverse each element of the array compare them with the pivot*/
    for (int j = low; j < high; j++) {
        if (array[j] <= pivot) {
            /* if element smaller than pivot is found swap it with the greater element pointed by i*/
            i++;
            swap(&array[i], &array[j]); // swap element at i with element at j
        }
    }

    // swap the pivot element with the greater element at i
    swap(&array[i + 1], &array[high]);

    // return the partition point
    return (i + 1);
}

void quickSort(int array[], int low, int high)
{
```

```
    if (low < high) {
        /* find the pivot element such that elements smaller than pivot are on left of pivot and
        elements greater than pivot are on right of pivot*/
    int pi = partition(array, low, high);

    // recursive call on the left of pivot
    quickSort(array, low, pi - 1);
    // recursive call on the right of pivot
    quickSort(array, pi + 1, high);
    }
}

// function to print array elements
void printArray(int array[], int size) {
    for (int i = 0; i < size; ++i)
    printf("%d ", array[i]);
}
```

```
Enter the size of array: 6

Enter the elements of array: 34 56 12 90 76 45

Array before Quick Sort: 34 56 12 90 76 45

Array after Quick Sort: 12 34 45 56 76 90

...Program finished with exit code 0
Press ENTER to exit console.
```

Conclusion: We have successfully studied and implemented Merge sort and Quick sort using recursion in C.