Experiment 2.1

Student Name:Rajiv Paul **UID:20BCS1812**
Branch: CSE **Section/Group:6B**
Semester: 3rd **Date of Performance: 29/09/2021**
Subject Name:Java Program Lab **Subject Code:20CSP-219**

Q1. Write a program to sort an array of floating-point numbers in descending order using merge sort ?

1) Aim/Overview of the practical:

To write a program to sort an array of floating-point numbers in descending order using merge sort.

2) Software required:

Vs Code

3) Source Code:

```cpp
   #include <iostream>
#include <algorithm>
using namespace std;

void merge(float arr[], int l, int m, int h){

    int n1=m-l+1, n2=h-m;
    float left[n1],right[n2];
    for(int i=0;i<n1;i++)
        left[i]=arr[i+l];
    for(int j=0;j<n2;j++)
        right[j]=arr[m+1+j];
    int i=0,j=0,k=l;
    while(i<n1 && j<n2){
        if(left[i]>=right[j])
            arr[k++]=left[i++];
        else
            arr[k++]=right[j++];
    }
    while(i<n1)
        arr[k++]=left[i++];
    while(j<n2)
        arr[k++]=right[j++];
}

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

int main() {

    float a[]={9.3,5.5,30.6,15.6,7.8};
    int l=0,r=4;
    cout<<"\nThe numbers arranged in descending order are: ";

    mergeSort(a,l,r);
    for(float x: a){
        cout<<x<<" ";
    }
}
```

4. Output:

```
The numbers arranged in descending order are: 30.6 15.6 9.3 7.8 5.5
```

Q2. Write a C/C++ program to sort a list of elements using the merge sort algorithm.

1) **Aim/Overview of the practical:**

To write a C/C++ program to sort a list of elements using the merge sort algorithm.

2) Software required:

Vs Code

3) Source Code:

```
#include <iostream>
using namespace std;

void merge(int arr[], int start, int mid, int end)
{

    int len1 = mid - start + 1;
    int len2 = end - mid;

    int leftArr[len1], rightArr[len2];

    for (int i = 0; i < len1; i++)
        leftArr[i] = arr[start + i];
    for (int j = 0; j < len2; j++)
        rightArr[j] = arr[mid + 1 + j];

    int i, j, k;
    i = 0;
    j = 0;
    k = start;
```

```
    while (i < len1 && j < len2)
    {
        if (leftArr[i] <= rightArr[j])
        {
            arr[k] = leftArr[i];
            i++;
        }
        else
        {
            arr[k] = rightArr[j];
            j++;
        }
        k++;
    }

    while (i < len1)
    {
        arr[k] = leftArr[i];
        i++;
        k++;
    }

    while (j < len2)
    {
        arr[k] = rightArr[j];
        j++;
        k++;
    }
}
```

```
void mergeSort(int arr[], int start, int end)
{
    if (start < end)
    {
        int mid = start + (end - start) / 2;

        mergeSort(arr, start, mid);
        mergeSort(arr, mid + 1, end);

        merge(arr, start, mid, end);
    }
}

void display(int arr[], int size)
{
    for (int i = 0; i < size; i++)
        cout << arr[i] << " ";
    cout << endl;
}
```

```
int main()
{
    int arr[] = {5, 3, 10, 7,  9, 1};
    int size = sizeof(arr) / sizeof(arr[0]);

    cout << "\nOriginal array: ";
    display(arr, size);

    mergeSort(arr, 0, size - 1);

    cout << "\nSorted array: ";
    display(arr, size);
    return 0;
}
```

4. Output: