Assignment No: 3

Name: Kedar Kale

Batch: B12

Roll No: SCETTYB129

Program: Design and implement Merge Sort algorithm using Divide and Conquer method for a given input. Determine the time required to search an element.

Algorithm:

Step 1 − if it is only one element in the list it is already sorted, return.
Step 2 − divide the list recursively into two halves until it can no more be divided.
Step 3 − merge the smaller lists into new list in sorted order.

Pseudocode

```
procedure mergesort( var a as array )
   if ( n == 1 ) return a

   var l1 as array = a[0] ... a[n/2]
   var l2 as array = a[n/2+1] ... a[n]

   l1 = mergesort( l1 )
   l2 = mergesort( l2 )

   return merge( l1, l2 )
end procedure

procedure merge( var a as array, var b as array )

   var c as array
   while ( a and b have elements )
      if ( a[0] > b[0] )
         add b[0] to the end of c
         remove b[0] from b
      else
         add a[0] to the end of c
         remove a[0] from a
      end if
   end while

   while ( a has elements )
      add a[0] to the end of c
```

```
    remove a[0] from a
  end while

  while ( b has elements )
    add b[0] to the end of c
    remove b[0] from b
  end while

  return c

end procedure
```

Program:

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


int merge(int arr[],int low,int mid,int high){
        int b[100],i,j,k;
        i = low;
        j = mid + 1;
        k = low;

        while((i<=mid)&&(j<=high)){
                if(arr[i]<arr[j]){
                        b[k] = arr[i];
                        k++;
                        i++;
                }
                else{
                        b[k] = arr[j];
                        k++;
                        j++;
                }
        }

        if(i>mid){
                while(j<=high){
                        b[k] = arr[j];
                        k++;
                        j++;
                }
        }
        else if(j>high){
                while(i<=mid){
                        b[k] = arr[i];
                        k++;
                        i++;
```

```
                }
        }

        while(low<=high){
                arr[low] = b[low];
                low++;
                }

}


int mergesort(int arr[],int low,int high){
int mid;
        if(low<high){
                mid = (low+high)/2;
                mergesort(arr,low,mid);
                mergesort(arr,mid+1,high);
                merge(arr,low,mid,high);
        }
}



int main()
{
        int n, i, j, count;
        cout<<"\nEnter the number of data element to be sorted: ";
        cin>>n;

        int a[n],freq[n];
        for(i = 0; i < n; i++)
        {
                cout<<"Enter element "<<i+1<<": ";
                cin>>a[i];
        }

        mergesort(a, 0, n-1);


        cout<<"\nSorted Data: ";
        for (i = 0; i < n; i++)
     cout<<" "<<a[i];

   for(i=0; i<n; i++)
   {
      count = 1;
      for(j=i+1; j<n; j++)
      {
         if(a[i] == a[j])
         {
```

```
            count++;
            freq[j] = 0;
         }
      }

      if(freq[i] != 0)
      {
         freq[i] = count;
      }
   }

   cout<<"\nUnique elements in the array are: ";
   for(i=0; i<n; i++)
   {
      if(freq[i] == 1)
      {
         cout<<" "<<a[i];
      }
   }
         return 0;
}
```

Input/Output:

```
Enter the number of data element to be sorted: 5
Enter element 1: 33
Enter element 2: 4
Enter element 3: 56
Enter element 4: 1
Enter element 5: 1

Sorted Data: 1 1 4 33 56

Unique elements in the array are: 4 33 56
```