Course Name:	**Data Structures Laboratory**	**Semester:**	**III**
Date of Performance:		**Batch No:**	**A1**
Faculty Name:	**Prof. Sushma Kadge**	**Roll No:**	**20**
Faculty Sign & Date:		**Grade/Marks:**	**/25**

Experiment No: 9
Title: Sorting Techniques

Aim and Objective of the Experiment:

To understand various sorting techniques.
Write a program for merge-sort. Given array elements are A[]={ 90,52,19,88,30,27,18,44}

COs to be achieved:

CO4: Demonstrate sorting and searching methods

Books/Journals/Websites referred:

1. GeeksForGeeks

Tools required:

DEV C/C++ compiler/ Code blocks C compiler

Theory:

Merge sort is a sorting algorithm that uses the divide, conquer, and combine algorithmic paradigm. Divide means partitioning the n-element array to be sorted into two sub-arrays of n/2 elements. If A is an array containing zero or one element, then it is already sorted. However, if there are more elements in the array, divide A into two sub-arrays, A1 and A2, each containing about half of the elements of A.
Conquer means sorting the two sub-arrays recursively using merge sort.
Combine means merging the two sorted sub-arrays of size n/2 to produce the sorted array of n elements. Merge sort algorithm focuses on two main concepts to improve its performance (running time):

- Σ A smaller list takes fewer steps and thus less time to sort than a large list.
- Σ As number of steps is relatively less, thus less time is needed to create a sorted list from two sorted lists rather than creating it using two unsorted lists.

Implementation details:

1. Enlist all the Steps followed and various options explored
2. Explain your program logic and methods used.
3. Explain the Importance of the approach followed by you

The program implements merge sort using the divide-and-conquer approach.
The array is recursively divided into two halves, each half is sorted individually, and then merged using the `merge()` function.
The approach ensures efficient sorting with O(n log n) complexity and demonstrates the use of recursion and array manipulation effectively.

C/C++ Code implemented:

```c
#include <stdio.h>

void merge(int a[], int left, int mid, int right)
{
    int i, j, k;
    int n1 = mid - left + 1;
    int n2 = right - mid;
    int L[n1], R[n2];
    for(i = 0; i < n1; i++)
        L[i] = a[left + i];
    for(j = 0; j < n2; j++)
        R[j] = a[mid + 1 + j];
    i = 0;
    j = 0;
    k = left;
    while(i < n1 && j < n2)
    {
        if(L[i] <= R[j])
        {
            a[k] = L[i];
            i++;
        }
        else
        {
            a[k] = R[j];
            j++;
        }
        k++;
    }
    while(i < n1)
    {
        a[k] = L[i];
        i++;
        k++;
    }
    while(j < n2)
    {
        a[k] = R[j];
        j++;
        k++;
    }
}
```

```c
void mergeSort(int a[], int left, int right)
{
    if(left < right)
    {
        int mid = left + (right - left) / 2;
        mergeSort(a, left, mid);
        mergeSort(a, mid + 1, right);
        merge(a, left, mid, right);
    }
}

int main()
{
    int a[] = {90, 52, 19, 88, 30, 27, 18, 44};
    int n = sizeof(a) / sizeof(a[0]);
    int i;
    printf("Original Array:\n");
    for(i = 0; i < n; i++)
        printf("%d ", a[i]);
    mergeSort(a, 0, n - 1);
    printf("\nSorted Array:\n");
    for(i = 0; i < n; i++)
        printf("%d ", a[i]);
    return 0;
}
```

Output/ program results after execution:

```
D:\Open Source\DS>gcc Sort.c -o Sort

D:\Open Source\DS>Sort
Original Array:
90 52 19 88 30 27 18 44
Sorted Array:
18 19 27 30 44 52 88 90
D:\Open Source\DS>
```

Post Lab Subjective/Objective type Questions:

1. Write a program for insertion sort.

Program:

```c
#include <stdio.h>

int main()
{
    int a[] = {90, 52, 19, 88, 30, 27, 18, 44};
    int n = sizeof(a) / sizeof(a[0]);
    int i, j, key;
    printf("Original Array:\n");
    for(i = 0; i < n; i++)
        printf("%d ", a[i]);
    for(i = 1; i < n; i++)
    {
        key = a[i];
        j = i - 1;
        while(j >= 0 && a[j] > key)
        {
            a[j + 1] = a[j];
            j--;
        }
        a[j + 1] = key;
    }
    printf("\nSorted Array:\n");
    for(i = 0; i < n; i++)
        printf("%d ", a[i]);
    return 0;
}
```

Output:

```
D:\Open Source\DS>gcc PSLSort.c -o PSLSort

D:\Open Source\DS>PSLSort
Original Array:
90 52 19 88 30 27 18 44
Sorted Array:
18 19 27 30 44 52 88 90
D:\Open Source\DS>
```

Conclusion:

The experiment successfully demonstrated the working of sorting algorithms using the divide-and-conquer paradigm.
Merge Sort efficiently sorted the array with **O(n log n)** complexity and ensured stable ordering.
Insertion Sort provided insight into a simpler, iterative sorting technique with **$O(n^2)$** complexity, suitable for smaller datasets.
The implementation reinforced the understanding of recursion, array manipulation, and algorithmic optimization.

Signature of faculty in-charge with Date: