Practical – 10 : Sorting techniques

Question - 1 : **Write a Program to collect an unsorted array from the user. Implement sorting of the array using following techniques.**

- **bubble sort**
- **quick sort.**
- **insertion sort**
- **Merge sort**

Source Code :

```c
#include <stdio.h>
void bubble(int a[], int n)
{
    int b;
    for (int i = 0; i < n - 1; i++)
    {
        for (int j = 0; j < n - 1; j++)
        {
            if (a[j] > a[j + 1])
            {
                b = a[j];
                a[j] = a[j + 1];
                a[j + 1] = b;
            }
        }
    }
}
void display(int a[], int n)
{
    for (int i = 0; i < n; i++)
    {
        printf(" %d", a[i]);
```

```
    }
}
void userinput(int a[], int n)
{
    for (int i = 0; i < n; i++)
    {
        printf(" Enter value for %d : ", i + 1);
        scanf("%d", &a[i]);
    }
}
int partition(int a[], int l, int h)
{
    int pivot = a[h];
    int i = (l - 1);

    for (int j = l; j <= h - 1; j++)
    {
        if (a[j] < pivot)
        {

            i++;

            int b = a[i];
            a[i] = a[j];
            a[j] = b;
        }
    }

    int b = a[i + 1];
    a[i + 1] = a[h];
    a[h] = b;
```

```
    return (i + 1);
}
void quick(int a[], int l, int h)
{
    if (l < h)
    {
        int i = partition(a, l, h);
        quick(a, l, i - 1);
        quick(a, i + 1, h);
    }
}
void insertion(int a[], int n)
{
    int i, temp, j;
    for (i = 1; i < n; i++)
    {
        temp = a[i];
        j = i - 1;

        while (j >= 0 && a[j] > temp)
        {
            a[j + 1] = a[j];
            j = j - 1;
        }
        a[j + 1] = temp;
    }
}
void merge(int a[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;
```

Practical – 10 : Sorting techniques

```
int L[n1], R[n2];

for (i = 0; i < n1; i++)
{
    L[i] = a[l + i];
}
for (j = 0; j < n2; j++)
{
    R[j] = a[m + 1 + j];
}

i = 0;
j = 0;
k = l;

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++;
    }
}

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

int main()
{
    int n;

    printf("\n Enter size : ");
    scanf("%d", &n);
```

```
    int a[n];

    printf("\n-----Bubble Sort-----\n");
    userinput(a, n);
    bubble(a, n);
    display(a, n);

    printf("\n-----Quick Sort-----\n");
    userinput(a, n);
    quick(a, 0, n - 1);
    display(a, n);

    printf("\n-----Insertion Sort-----\n");
    userinput(a, n);
    insertion(a, n);
    display(a, n);

    printf("\n-----Merge Sort-----\n");
    userinput(a, n);
    mergeSort(a, 0, n - 1);
    display(a, n);

    return 0;
}
```

Output :

PS D:\Alyani\MA003_P10> gcc -o l10p1 l10p1.c

PS D:\Alyani\MA003_P10> ./l10p1

Practical – 10 : Sorting techniques

Enter size : 5

-----Bubble Sort-----

Enter value for 1 : 9

Enter value for 2 : 3

Enter value for 3 : 4

Enter value for 4 : -2

Enter value for 5 : -1

-2 -1 3 4 9

-----Quick Sort-----

Enter value for 1 : 1

Enter value for 2 : 8

Enter value for 3 : 4

Enter value for 4 : 7

Enter value for 5 : 6

1 4 6 7 8

-----Insertion Sort-----

Enter value for 1 : 3

Enter value for 2 : 5

Enter value for 3 : 4

Enter value for 4 : 1

Enter value for 5 : -7

-7 1 3 4 5

Practical – 10 : Sorting techniques

-----Merge Sort-----

Enter value for 1 : 1

Enter value for 2 : 4

Enter value for 3 : 0

Enter value for 4 : 9

Enter value for 5 : 77

0 1 4 9 77