Practical 10: Sorting techniques

Name: Sutariya Savankumar

Roll no: MA065

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**

Code

```
#include <stdio.h>
#include <stdlib.h>

void swap(int *x, int *y){
    int temp = *x;
    *x = *y;
    *y = temp;
}

void bubbleSort(int arr[], int n){
    for (int i = 0; i < n - 1; i++){
        for (int j = 0; j < n - i - 1; j++){
            if (arr[j] > arr[j + 1]){
                swap(&arr[j], &arr[j + 1]);
            }
        }
    }
}

void insertionSort(int arr[], int n){
    int i, key, j;
    for (i = 1; i < n; i++){
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key){
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
```

```
}

int partition(int arr[], int low, int high){
    int pivot = arr[high];
    int i = (low - 1);
    for (int j = low; j <= high - 1; j++){
        if (arr[j] <= pivot){
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}

void quickSort(int arr[], int low, int high){
    if (low < high){
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

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

int main(){

    // Collecting an unsorted array from the user
    printf("Enter the number of elements in the array: ");
    int n;
    scanf("%d", &n);

    printf("Enter the elements of the array: ");
    int arr[n];
    for (int i = 0; i < n; i++){
        scanf("%d", &arr[i]);
    }

    // Sorting the array using bubble sort
    bubbleSort(arr, n);
    printf("Sorted array using bubble sort: ");
    printArray(arr, n);
```

```
    // Sorting the array using insertion sort
    insertionSort(arr, n);
    printf("\nSorted array using insertion sort: ");
    printArray(arr, n);

    // Sorting the array using quick sort
    quickSort(arr, 0, n - 1);
    printf("\nSorted array using quick sort: ");
    printArray(arr, n);

    // Sorting the array using merge sort
    mergeSort(arr, 0, n - 1);
    printf("\nSorted array using merge sort: ");
    printArray(arr, n);
}
```

Output

```
Savan@Savan MINGW64 /c/Drive/Study/MCA/DSsubmission/MA065_Savan (master)
$ gcc prac10-01.c -o p1

Savan@Savan MINGW64 /c/Drive/Study/MCA/DSsubmission/MA065_Savan (master)
$ ./p1
Enter the number of elements in the array: 5
Enter the elements of the array: 3 7 4 2 5
Sorted array using bubble sort: 2 3 4 5 7
Sorted array using insertion sort: 2 3 4 5 7
Sorted array using quick sort: 2 3 4 5 7
Sorted array using merge sort: 2 3 4 5 7
```