Experiment 14

CODE:

```
#include <stdio.h>

void displayArray(int arr[], int n) {
   for (int i = 0; i < n; i++)
      printf("%d ", arr[i]);
   printf("\n");
}

void merge(int arr[], 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] = arr[left + i];
   for (j = 0; j < n2; j++)
      R[j] = arr[mid + 1 + j];

   i = 0;
   j = 0;
   k = left;

   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++;
    }

    printf("Merged array: ");
    displayArray(arr + left, right - left + 1);
}

void mergeSort(int arr[], int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;

        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}

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++;
            int temp = arr[i];
```

```
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    int temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;

    return i + 1;
}

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

        printf("Array after partitioning with pivot %d: ", arr[pi]);
        displayArray(arr + low, high - low + 1);

        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int main() {
    int arr[100], n, choice, i;

    printf("Enter number of elements: ");
    scanf("%d", &n);

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

    do {
        printf("\nMenu:\n");
```

```
        printf("1. Merge Sort\n");
        printf("2. Quick Sort\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        int tempArr[100];
        for (i = 0; i < n; i++) {
            tempArr[i] = arr[i];
        }

        switch (choice) {
            case 1:
                printf("Merge Sort:\n");
                mergeSort(tempArr, 0, n - 1);
                break;
            case 2:
                printf("Quick Sort:\n");
                quickSort(tempArr, 0, n - 1);
                break;
            case 3:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    } while (choice != 3);

    return 0;
}
```

Output:

```
Enter number of elements: 10
Enter the elements of the array:
12
21
10
09
0
5
97
2
5
85

Menu:
1. Merge Sort
2. Quick Sort
3. Exit
Enter your choice: 1
Merge Sort:
Merged array: 12 21
Merged array: 10 12 21
Merged array: 0 9
Merged array: 0 9 10 12 21
Merged array: 5 97
Merged array: 2 5 97
Merged array: 5 85
Merged array: 2 5 5 85 97
Merged array: 0 2 5 5 9 10 12 21 85 97
```

Menu:
1. Merge Sort
2. Quick Sort
3. Exit
Enter your choice: 2
Quick Sort:
Array after partitioning with pivot 85: 12 21 10 9 0 5 2 5 85 97
Array after partitioning with pivot 5: 0 2 5 9 12 5 21 10
Array after partitioning with pivot 2: 0 2
Array after partitioning with pivot 10: 9 5 10 21 12
Array after partitioning with pivot 5: 5 9
Array after partitioning with pivot 12: 12 21

Menu:
1. Merge Sort
2. Quick Sort
3. Exit
Enter your choice: 3
Exiting...

=== Code Execution Successful ===