DATA STRUCTURES

Division	**CS(AIML) -A**
Batch	**2**
GR-no	**12311493**
Roll no	**54**
Name	**Atharva Kangralkar**

Assignment 4:

Write a C/C++ Program to implement Quick sort and Merge Sort

Code:-

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

void merge(int arr[], int low, int mid, int high) {
    int temp[100];
    int left = low, right = mid + 1, k = 0;

    while (left <= mid && right <= high) {
        if (arr[left] <= arr[right]) {
            temp[k++] = arr[left++];
        } else {
            temp[k++] = arr[right++];
        }
    }
    while (left <= mid) temp[k++] = arr[left++];
    while (right <= high) temp[k++] = arr[right++];

    for (int i = low, j = 0; i <= high; i++, j++) {
        arr[i] = temp[j];
    }
}

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

```
}


int partition(int arr[], int low, int high) {
    int pivot = arr[low], left = low + 1, right = high;
    while (left <= right) {
        while (left <= right && arr[left] <= pivot) left++;
        while (left <= right && arr[right] > pivot) right--;
        if (left < right) {
            int temp = arr[left];
            arr[left] = arr[right];
            arr[right] = temp;
        }
    }
    arr[low] = arr[right];
    arr[right] = pivot;
    return right;
}


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

void displayArray(int arr[], int n) {
```

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

int main() {
    int n, choice;

    printf("Enter the size of the array: ");
    scanf("%d", &n);
    int arr[n];

    printf("Enter the elements of the array:\n");
    for (int 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);

        switch (choice) {
            case 1:
                mergeSort(arr, 0, n - 1);
```

```
            printf("Sorted array using Merge Sort:\n");
            displayArray(arr, n);
            break;
        case 2:
            quickSort(arr, 0, n - 1);
            printf("Sorted array using Quick Sort:\n");
            displayArray(arr, n);
            break;
        case 3:
            printf("Exiting program...\n");
            break;
        default:
            printf("Invalid choice! Please enter a valid option.\n");
      }
   } while (choice != 3);

   return 0;
}
```

Code Screenshot:-

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

void merge(int arr[], int low, int mid, int high) {
    int temp[100];
    int left = low, right = mid + 1, k = 0;

    while (left <= mid && right <= high) {
        if (arr[left] <= arr[right]) {
            temp[k++] = arr[left++];
        } else {
            temp[k++] = arr[right++];
        }
    }
    while (left <= mid) temp[k++] = arr[left++];
    while (right <= high) temp[k++] = arr[right++];

    for (int i = low, j = 0; i <= high; i++, j++) {
        arr[i] = temp[j];
    }
}

void mergeSort(int arr[], int low, int high) {
    if (low < high) {
        int mid = low + (high - low) / 2;
        mergeSort(arr, low, mid);
```

```
        mergeSort(arr, low, mid);
        mergeSort(arr, mid + 1, high);
        merge(arr, low, mid, high);
    }
}


int partition(int arr[], int low, int high) {
    int pivot = arr[low], left = low + 1, right = high;
    while (left <= right) {
        while (left <= right && arr[left] <= pivot) left++;
        while (left <= right && arr[right] > pivot) right--;
        if (left < right) {
            int temp = arr[left];
            arr[left] = arr[right];
            arr[right] = temp;
        }
    }
    arr[low] = arr[right];
    arr[right] = pivot;
    return right;
}


void quickSort(int arr[], int low, int high) {
    if (low < high) {
```

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

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

int main() {
    int n, choice;

    printf("Enter the size of the array: ");
    scanf("%d", &n);
    int arr[n];

    printf("Enter the elements of the array:\n");
    for (int 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);

        switch (choice) {
            case 1:
                mergeSort(arr, 0, n - 1);
                printf("Sorted array using Merge Sort:\n");
                displayArray(arr, n);
                break;
            case 2:
                quickSort(arr, 0, n - 1);
                printf("Sorted array using Quick Sort:\n");
                displayArray(arr, n);
                break;
            case 3:
                printf("Exiting program...\n");
                break;
            default:
                printf("Invalid choice! Please enter a valid option
                    .\n");
        }
        }
    } while (choice != 3);

    return 0;
}
```

Output:-

```
Enter the size of the array: 5
Enter the elements of the array:
4
8
1
0
9

Menu:
1. Merge Sort
2. Quick Sort
3. Exit
Enter your choice: 1
Sorted array using Merge Sort:
0 1 4 8 9

Menu:
1. Merge Sort
2. Quick Sort
3. Exit
Enter your choice: 2
Sorted array using Quick Sort:
0 1 4 8 9
```