

**Data Structure and Algorithm (MCA 271)**

**Lab Practical –**

***BY***

**Himanshu Heda (24225013)**

**SUBMITTED TO**

**Prof. Vandna Kansal**

**SCHOOL OF SCIENCES**

**2024-2025**

**Program Description:**

**Code of the program**

**Output**: - Paste the o/p of the program.

1. Quick Sort : --

#include <stdio.h>

// Function to swap two elements

void swap(int \*a, int \*b) {

    int temp = \*a;

    \*a = \*b;

    \*b = temp;

}

// Partition function to rearrange elements around the pivot

int partition(int arr[], int low, int high) {

    int pivot = arr[high]; // Choose the last element as the pivot

    int i = low - 1;       // Index of the smaller element

    for (int j = low; j < high; j++) {

        // If the current element is smaller than or equal to the pivot

        if (arr[j] <= pivot) {

            i++; // Increment the index of the smaller element

            swap(&arr[i], &arr[j]);

        }

    }

    // Place the pivot in the correct position

    swap(&arr[i + 1], &arr[high]);

    return i + 1; // Return the partition index

}

// Quick Sort function

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

    if (low < high) {

        // Partition the array and get the pivot index

        int pi = partition(arr, low, high);

        // Recursively sort elements before and after partition

        quickSort(arr, low, pi - 1);

        quickSort(arr, pi + 1, high);

    }

}

// Function to print an array

void printArray(int arr[], int size) {

    for (int i = 0; i < size; i++)

        printf("%d ", arr[i]);

    printf("\n");

}

// Main function

int main() {

    int arr[] = {38, 27, 43, 3, 9, 82, 10};

    int n = sizeof(arr) / sizeof(arr[0]);

    printf("Given array is:\n");

    printArray(arr, n);

    quickSort(arr, 0, n - 1);

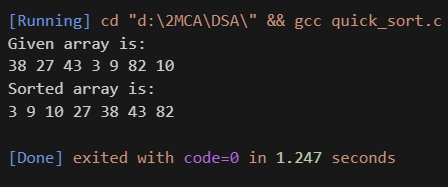
    printf("Sorted array is:\n");

    printArray(arr, n);

    return 0;

}

OUTPUT : --



1. Merge Sort : --

#include <stdio.h>

#include <stdlib.h>

// Function to merge two halves of an array

void merge(int arr[], int left, int mid, int right) {

    int i, j, k;

    int n1 = mid - left + 1; // Size of the left subarray

    int n2 = right - mid;    // Size of the right subarray

    // Create temporary arrays

    int \*L = (int \*)malloc(n1 \* sizeof(int));

    int \*R = (int \*)malloc(n2 \* sizeof(int));

    // Copy data to temporary arrays L[] and R[]

    for (i = 0; i < n1; i++)

        L[i] = arr[left + i];

    for (j = 0; j < n2; j++)

        R[j] = arr[mid + 1 + j];

    // Merge the temporary arrays back into arr[left..right]

    i = 0; // Initial index of the left subarray

    j = 0; // Initial index of the right subarray

    k = left; // Initial index of the merged subarray

    while (i < n1 && j < n2) {

        if (L[i] <= R[j]) {

            arr[k] = L[i];

            i++;

        } else {

            arr[k] = R[j];

            j++;

        }

        k++;

    }

    // Copy the remaining elements of L[], if any

    while (i < n1) {

        arr[k] = L[i];

        i++;

        k++;

    }

    // Copy the remaining elements of R[], if any

    while (j < n2) {

        arr[k] = R[j];

        j++;

        k++;

    }

    // Free the temporary arrays

    free(L);

    free(R);

}

// Function to implement merge sort

void mergeSort(int arr[], int left, int right) {

    if (left < right) {

        int mid = left + (right - left) / 2; // Find the middle point

        // Sort first and second halves

        mergeSort(arr, left, mid);

        mergeSort(arr, mid + 1, right);

        // Merge the sorted halves

        merge(arr, left, mid, right);

    }

}

// Function to print an array

void printArray(int arr[], int size) {

    for (int i = 0; i < size; i++)

        printf("%d ", arr[i]);

    printf("\n");

}

// Main function

int main() {

    int arr[] = {38, 27, 43, 3, 9, 82, 10};

    int arr\_size = sizeof(arr) / sizeof(arr[0]);

    printf("Given array is:\n");

    printArray(arr, arr\_size);

    mergeSort(arr, 0, arr\_size - 1);

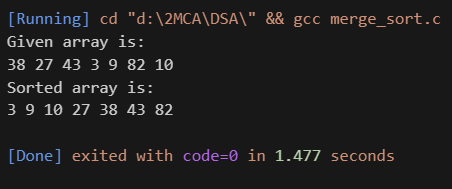
    printf("Sorted array is:\n");

    printArray(arr, arr\_size);

    return 0;

}

OUTPUT : --

****