**ASSIGNMENT-4**

**EXPERIMENT – 4: Write a program to implement Merge Sort algorithm.**

#include <stdio.h>

void merge(int arr[], int p, int q, int r);

void mergeSort(int arr[], int l, int r);

int main() {

int array[50];

int n,i;

printf("\nEnter the Size of the array:");

scanf("%d",&n);

printf("\nEnter the elements of the array:-\n");

for(i=0;i<n;i++){

scanf("%d",&array[i]);

}

mergeSort(array, 0,n - 1);

printf("Sorted array in ascending order: \n");

for (int i = 0; i < n; i++) {

printf("%d ", array[i]);

}

printf("\n");

printf("Sorted array in descending order: \n");

for (int i = n-1; i >= 0; i--) {

printf("%d ", array[i]);

} }

void merge(int arr[], int p, int q, int r) {

int n1 = q - p + 1;

int n2 = r - q;

int L[n1], M[n2];

for (int i = 0; i < n1; i++)

L[i] = arr[p + i];

for (int j = 0; j < n2; j++)

M[j] = arr[q + 1 + j];

int i, j, k;

i = 0;

j = 0;

k = p;

while (i < n1 && j < n2) {

if (L[i] <= M[j]) {

arr[k] = L[i];

i++;

} else {

arr[k] = M[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = M[j];

j++;

k++;

} }

void mergeSort(int arr[], int l, int r) {

if (l < r) {

int m = l + (r - l) / 2;

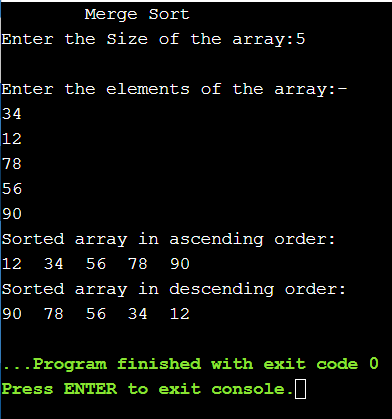
mergeSort(arr, l, m);

mergeSort(arr, m + 1, r);

merge(arr, l, m, r);

} }

**OUTPUT-**



**EXPERIMENT – 5: Write a program to demonstrate the Quick Sort algorithm.**

#include <stdio.h>

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

int partition(int array[], int low, int high);

void quickSort(int array[], int low, int high);

int main() {

int array[50];

int n,i;

printf("\tQuick Sort\t");

printf("\nEnter the Size of the array:");

scanf("%d",&n);

printf("\nEnter the elements of the array:-\n");

for(i=0;i<n;i++){

scanf("%d",&array[i]);

}

quickSort(array, 0, n - 1);

printf("Sorted array in ascending order: \n");

for (int i = 0; i < n; i++) {

printf("%d ", array[i]);

}

printf("\n");

printf("Sorted array in descending order: \n");

for (int i = n-1; i >= 0; i--) {

printf("%d ", array[i]);

}

}

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

int t = \*a;

\*a = \*b;

\*b = t;

}

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

int pivot = array[high];

int i = (low - 1);

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

if (array[j] <= pivot) {

i++;

swap(&array[i], &array[j]);

}

}

swap(&array[i + 1], &array[high]);

return (i + 1);

}

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

if (low < high) {

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

quickSort(array, low, pi - 1);

quickSort(array, pi + 1, high);

}

}

**OUTPUT-**

