LAB TASK – 4

1. Write a program to sort the array elements using Merge Sort Technique.

Code :

#include <stdio.h>

#include <stdlib.h>

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

int i, j, k;

int n1 = mid - left + 1;

int n2 = right - mid;

int leftArr[n1], rightArr[n2];

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

leftArr[i] = arr[left + i];

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

rightArr[j] = arr[mid + 1 + j];

i = 0;

j = 0;

k = left;

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

if (leftArr[i] <= rightArr[j]) {

arr[k] = leftArr[i];

i++;

}

else {

arr[k] = rightArr[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = leftArr[i];

i++;

k++;

}

while (j < n2) {

arr[k] = rightArr[j];

j++;

k++;

}

}

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 main() {

int n, i;

printf("Enter the number of elements: ");

scanf("%d", &n);

int arr[n];

printf("Enter %d elements:\n", n);

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

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

mergeSort(arr, 0, n - 1);

printf("Sorted array:\n");

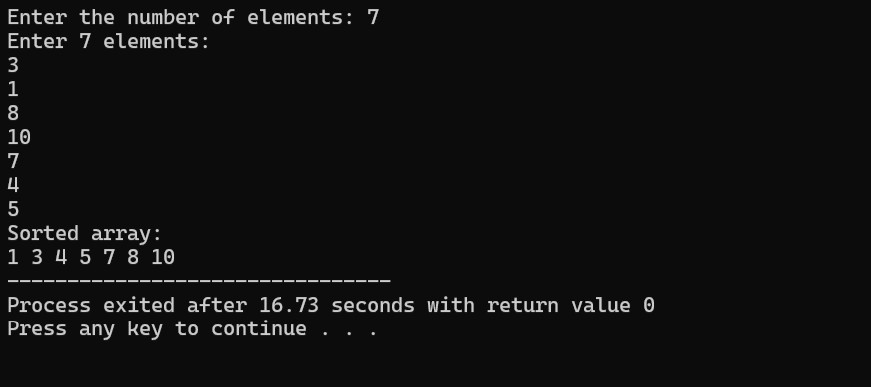
for (i = 0; i < n; i++)

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

return 0;

}

Output :



1. Write a program to sort the array elements using Bucket Sort Technique.

Code :

#include <stdio.h>

#define MAX 100

void bucketSort(int arr[], int n) {

int i,buckets[MAX] = {0};

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

buckets[arr[i]]++;

}

int index = 0;

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

while (buckets[i] > 0) {

arr[index++] = i;

buckets[i]--;

}

}

}

int main() {

int n, i;

printf("Enter the number of elements: ");

scanf("%d", &n);

int arr[n];

printf("Enter %d elements:\n", n);

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

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

printf("Original array:\n");

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

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

printf("\n");

bucketSort(arr, n);

printf("Sorted array:\n");

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

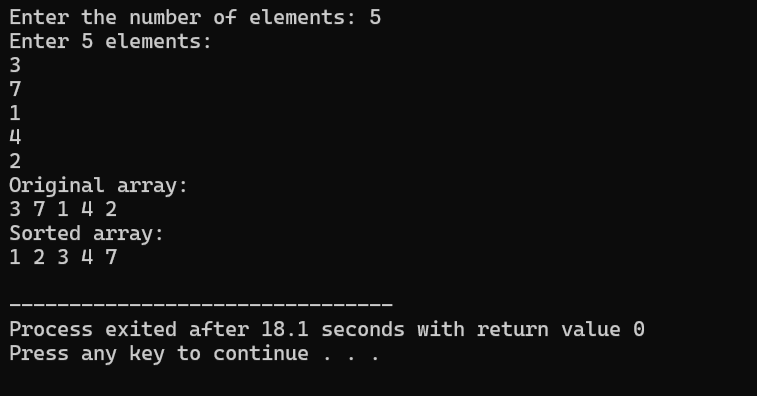
printf("%d ", arr[i]);

printf("\n");

return 0;

}

Output:



1. Write a program to sort the array elements using Quick Sort Technique.

Code:

#include <stdio.h>

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

int t = \*a;

\*a = \*b;

\*b = t;

}

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

int pivot = arr[high], i = low - 1,j;

for (j = low; j < high; 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);

}

}

int main() {

int n, i;

printf("Enter the number of elements: ");

scanf("%d", &n);

int arr[n];

printf("Enter %d elements:\n", n);

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

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

printf("Original array:\n");

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

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

printf("\n");

quickSort(arr, 0, n - 1);

printf("Sorted array:\n");

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

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

printf("\n");

return 0;

}

Output:

