1. **Write a program to sort the array elements using Merge Sort Technique. Derive its time complexity.**

#include <stdio.h>

void merge(int arr[], int L, int mid, int R){

int n1 = mid - L + 1;

int n2 = R - mid;

int Left[n1], Right[n2];

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

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

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

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

}

int i = 0, j = 0, k = L;

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

if (Left[i] <= Right[j]){

arr[k++] = Left[i++];

}else{

arr[k++] = Right[j++];

}

}while (i < n1){

arr[k++] = Left[i++];

}while (j < n2){

arr[k++] = Right[j++];

}

}

void mergeSort(int arr[], int L, int R){

if (L < R){

int mid = (L + R) / 2;

mergeSort(arr, L, mid);

mergeSort(arr, mid + 1, R);

merge(arr, L, mid, R);

}

}

int main(){

int n;

printf("Enter no. of elements: ");

scanf("%d", &n);

int arr[n];

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

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

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

}

mergeSort(arr, 0, n - 1);

printf("Sorted array: ");

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

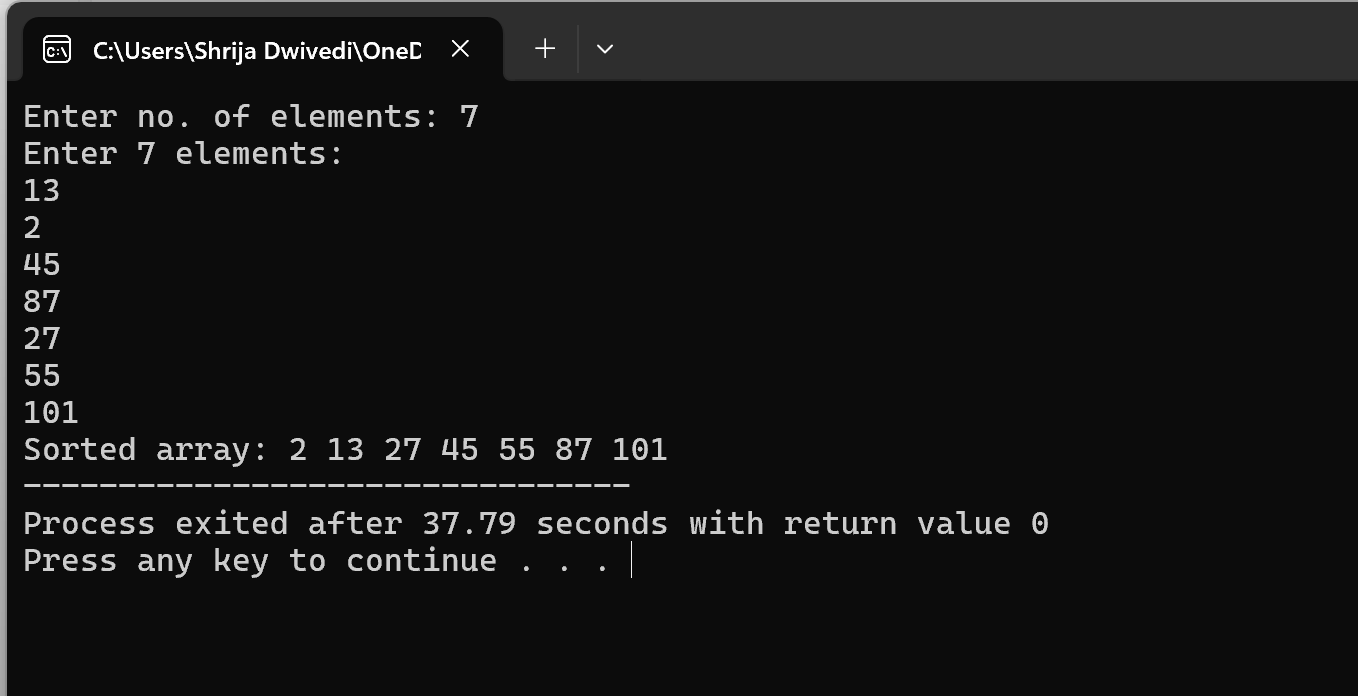
printf("%d ", arr[i]);

}

return 0;

}

Output:



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

Derive its time complexity.

#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++)

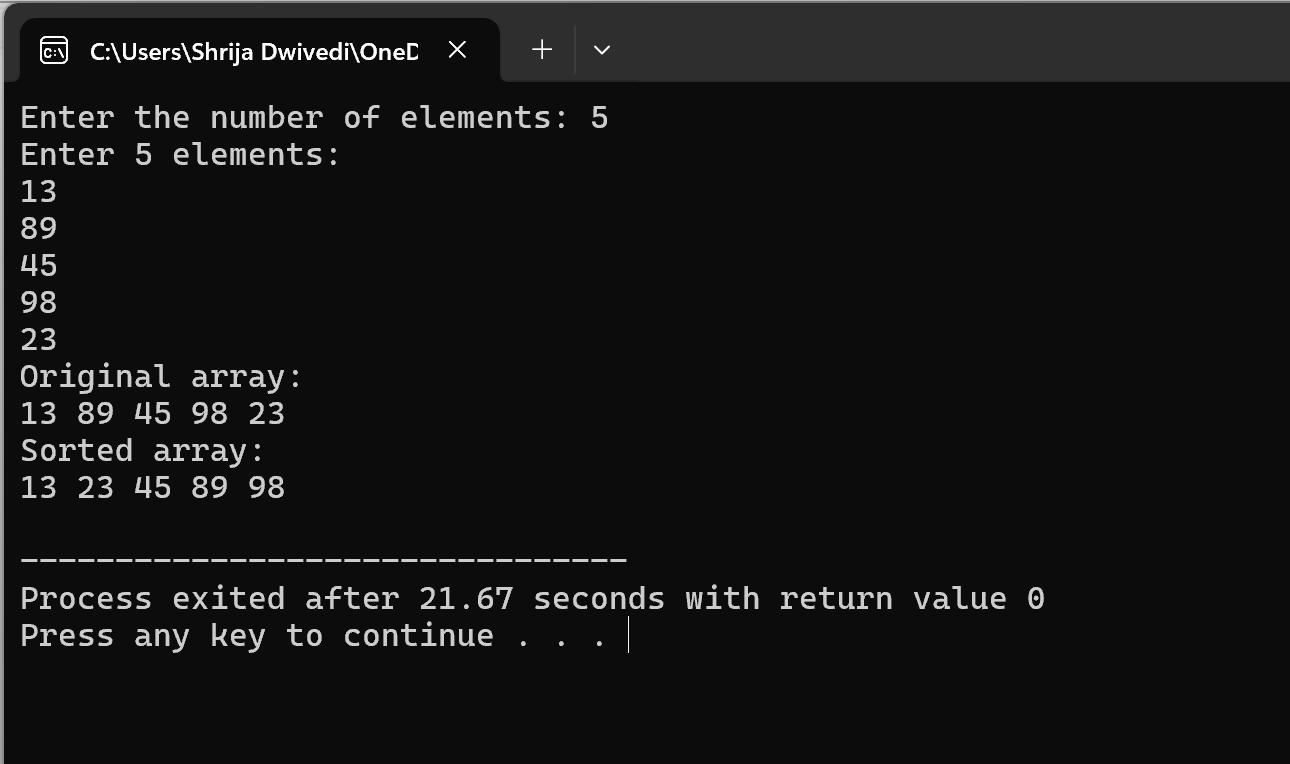
printf("%d ", arr[i]);

printf("\n");

return 0;

}

Output:



1. **Write a program to sort the array elements using Quick Sort Technique.**  
   **Derive its time complexity**

#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:

