Experiment-20 : Heap Sort

20.Write a C Program To Implement Heap Sort.

Code :

#include <stdio.h>

// Function to swap two numbers

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

int temp = \*a;

\*a = \*b;

\*b = temp;

}

// Function to heapify a subtree rooted with node i which is an index in arr[]

void heapify(int arr[], int n, int i) {

int largest = i; // Initialize largest as root

int left = 2 \* i + 1; // left = 2\*i + 1

int right = 2 \* i + 2; // right = 2\*i + 2

// If left child is larger than root

if (left < n && arr[left] > arr[largest])

largest = left;

// If right child is larger than largest so far

if (right < n && arr[right] > arr[largest])

largest = right;

// If largest is not root

if (largest != i) {

swap(&arr[i], &arr[largest]);

// Recursively heapify the affected sub-tree

heapify(arr, n, largest);

}

}

// main function to do heap sort

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

// Build heap (rearrange array)

for (int i = n / 2 - 1; i >= 0; i--)

heapify(arr, n, i);

// One by one extract an element from heap

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

// Move current root to end

swap(&arr[0], &arr[i]);

// call max heapify on the reduced heap

heapify(arr, i, 0);

}

// Function to print an array

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

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

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

printf("\n");

}

// Driver program

int main() {

int arr[] = {12, 11, 13, 5, 6, 7};

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

printf("Original array: \n");

printArray(arr, n);

heapSort(arr, n);

printf("Sorted array: \n");

printArray(arr, n);

return 0;

}

Output :

