EXPERIMENT [19](heap sort)

CODE:

#include <stdio.h>

// Function to heapify a subtree rooted at index i

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

int largest = i; // Initialize largest as root

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

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

// 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

int temp = arr[i];

arr[i] = arr[largest];

arr[largest] = temp;

// Recursively heapify the affected sub-tree

heapify(arr, n, largest);

}

}

// Main function to perform heap sort

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

// Build max heap (rearrange array)

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

heapify(arr, n, i);

// One by one extract elements from heap

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

// Move current root to end

int temp = arr[0];

arr[0] = arr[i];

arr[i] = temp;

// call max heapify on the reduced heap

heapify(arr, i, 0);

}

}

// Function to print array

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

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

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

printf("\n");

}

// Driver code

int main() {

int arr[100], n;

printf("Enter number of elements: ");

scanf("%d", &n);

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

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

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

printf("Original array: ");

printArray(arr, n);

heapSort(arr, n);

printf("Sorted array using Heap Sort: ");

printArray(arr, n);

return 0;

}

OUTPUT:

