#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) {

int temp = arr[i];

arr[i] = arr[largest];

arr[largest] = temp;

// Recursively heapify the affected subtree

heapify(arr, n, largest);

}

}

// Heap Sort function

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

// Build max heap

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

heapify(arr, n, i);

// Extract elements from heap one by one

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 heapify on the reduced heap

heapify(arr, i, 0);

}

}

int main() {

int arr[100], n;

// Input: number of elements

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

scanf("%d", &n);

// Input: array elements

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

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

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

// Perform Heap Sort

heapSort(arr, n);

// Output: sorted array

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

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

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

printf("\n");

return 0;

}