Implement Heap sort to sort the given set of values using max or min heap

Code

```
#include <iostream>
using namespace std;

// Function to heapify a subtree rooted at index
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) {
        std::swap(arr[i], arr[largest]);

        // Recursively heapify the affected subtree
        heapify(arr, n, largest);
    }
}

// Function to implement heap sort
void heapSort(int arr[], int n) {
```

```
    // Build heap
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

    // Extract elements one by one
    for (int i = n - 1; i >= 0; i--) {
        std::swap(arr[0], arr[i]);  // Move current root to end
        heapify(arr, i, 0);  // Call heapify on the reduced heap
    }
}

// Function to print array
void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++)
        std::cout << arr[i] << " ";
    std::cout << std::endl;
}

// Driver code
int main() {
    int arr[] = {98,65,34,78,21,5,7};
    int n = sizeof(arr) / sizeof(arr[0]);

    std::cout << "Original array: ";
    printArray(arr, n);

    heapSort(arr, n);

    std::cout << "Sorted array: ";
    printArray(arr, n);

    return 0;
```

}

Output

```
C:\Users\Gauri Saraf\Documents\heap.exe
Original array: 98 65 34 78 21 5 7
Sorted array: 5 7 21 34 65 78 98

--------------------------------
Process exited after 0.5891 seconds with return value 0
Press any key to continue . . .
```