ASSIGNMENT 9: HEAP SORT

Code:

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

// function to heapify the tree
void heapify(int arr[], int n, int root)
{
    int largest = root;   // root is the largest element
    int l = 2 * root + 1; // left = 2*root + 1
    int r = 2 * root + 2; // right = 2*root + 2

    // If left child is larger than root
    if (l < n && arr[l] > arr[largest])
        largest = l;

    // If right child is larger than largest so far
    if (r < n && arr[r] > arr[largest])
        largest = r;

    // If largest is not root
    if (largest != root)
    {
        // swap root and largest
        swap(arr[root], arr[largest]);

        // Recursively heapify the sub-tree
        heapify(arr, n, largest);
    }
```

```
}

// implementing heap sort
void heapSort(int arr[], int n)
{
    // build heap
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

    // extracting elements from heap one by one
    for (int i = n - 1; i >= 0; i--)
    {
        // Move current root to end
        swap(arr[0], arr[i]);

        // again call max heapify on the reduced heap
        heapify(arr, i, 0);
    }
}

/* print contents of array - utility function */
void displayArray(int arr[], int n)
{
    for (int i = 0; i < n; ++i)
        cout << arr[i] << " ";
    cout << "\n";
}
```

```
// main program
int main()
{
    int heap_arr[] = {4, 17, 3, 12, 9, 6};
    int n = sizeof(heap_arr) / sizeof(heap_arr[0]);
    cout << "Input array" << endl;
    displayArray(heap_arr, n);

    heapSort(heap_arr, n);

    cout << "Sorted array" << endl;

    displayArray(heap_arr, n);
}
```

OUTPUT:

```
Input array
4 17 3 12 9 6
Sorted array
3 4 6 9 12 17
```