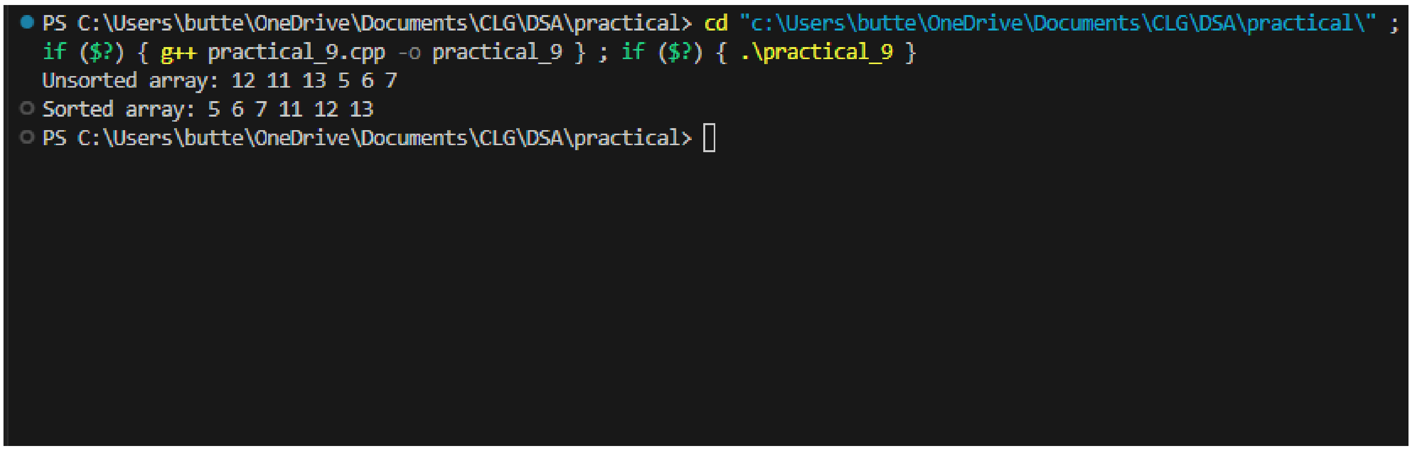
Practical 9
Source Code:-

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

// Function to heapify a subtree rooted with node i
void heapify(vector<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]); // Swap root with largest

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

// Function to perform heap sort
void heapSort(vector<int>& arr) {
    int n = arr.size();

    // Build a maxheap
    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--) {
        swap(arr[0], arr[i]); // Move current root to end
        heapify(arr, i, 0); // Call max heapify on the reduced heap
    }
}

// Function to print an array
void printArray(const vector<int>& arr) {
    for (int val : arr)
        cout << val << " ";
    cout << endl;
}

int main() {
    vector<int> arr = {12, 11, 13, 5, 6, 7};

    cout << "Unsorted array: ";
    printArray(arr);

    heapSort(arr);

    cout << "Sorted array: ";
    printArray(arr);

    return 0;
}
```

Output:-



```
PS C:\Users\butte\OneDrive\Documents\CLG\DSA\practical> cd "c:\Users\butte\OneDrive\Documents\CLG\DSA\practical\" ;
if ($?) { g++ practical_9.cpp -o practical_9 } ; if ($?) { .\practical_9 }
Unsorted array: 12 11 13 5 6 7
Sorted array: 5 6 7 11 12 13
PS C:\Users\butte\OneDrive\Documents\CLG\DSA\practical>
```