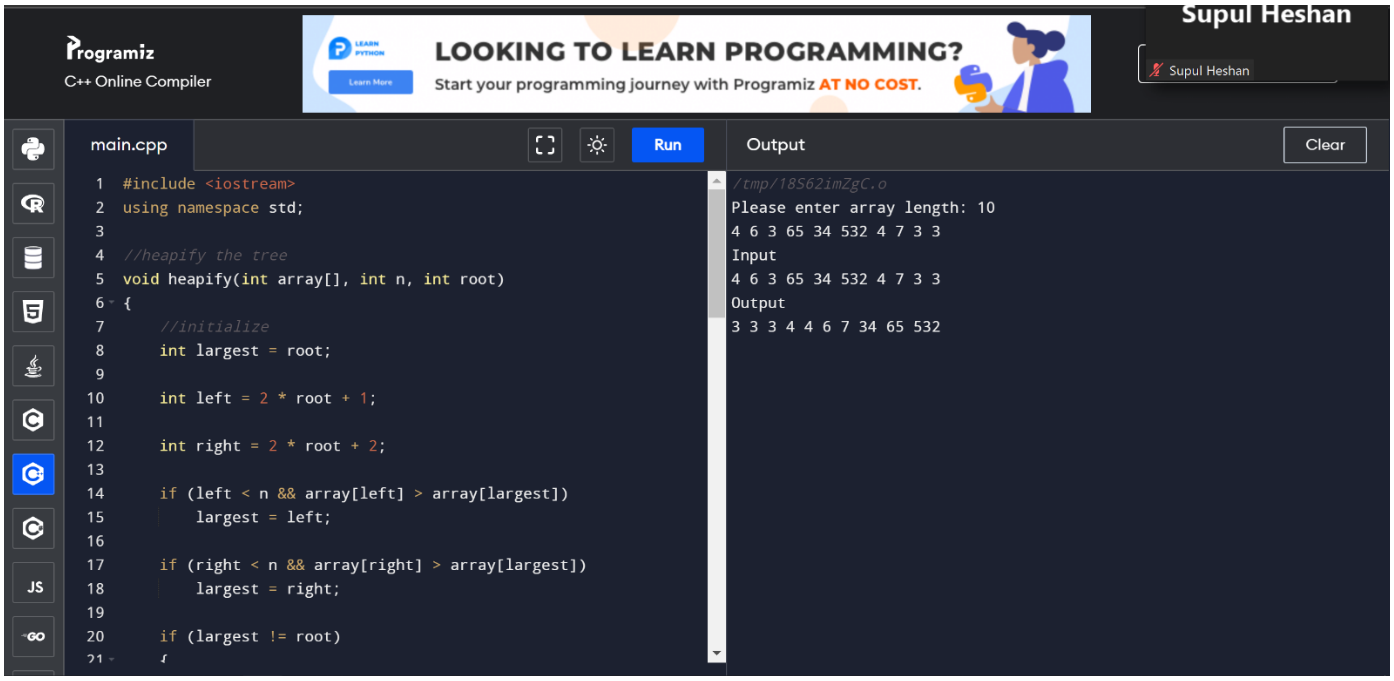
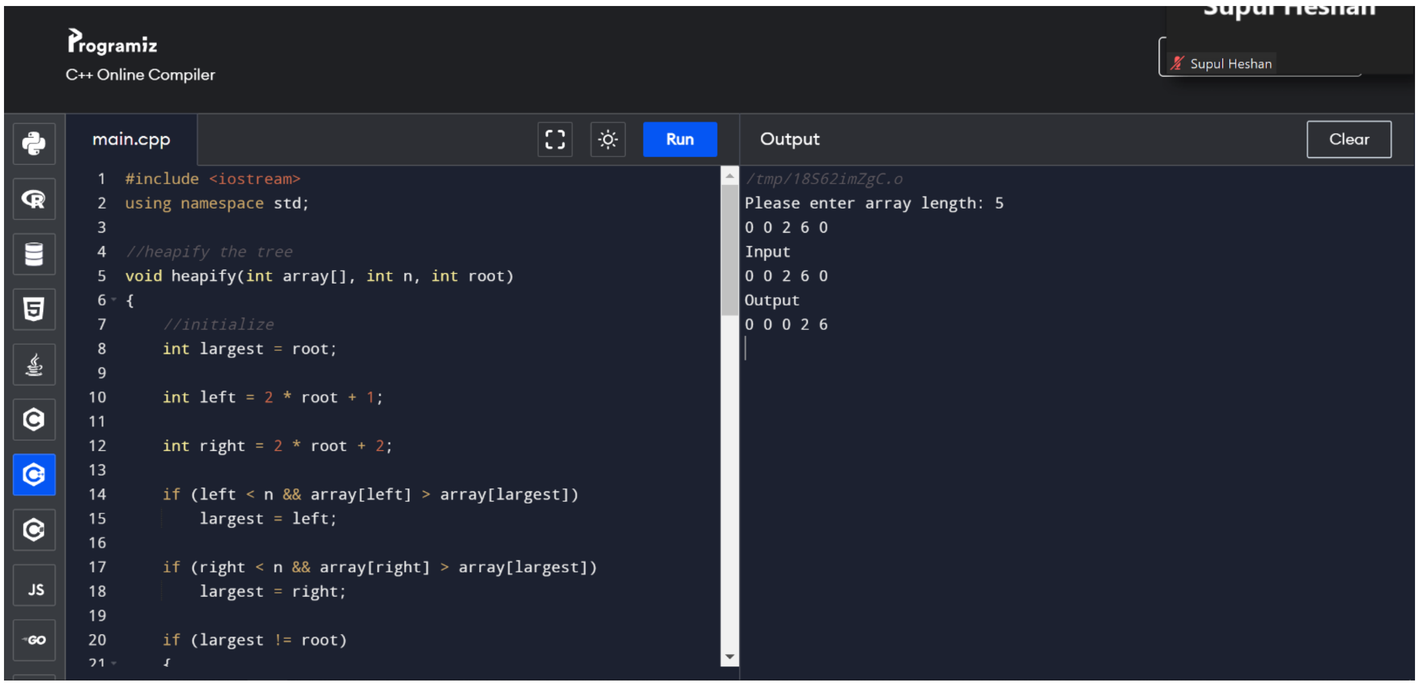
200650U – Thilakarathne D.L.J - CS2023 – Lab 8

Github link - https://github.com/LasithaJananjaya/CS2023-Data-Structures-and-Algorithms---Workspace/tree/main/in-class-lab8


Programiz
C++ Online Compiler
LOOKING TO LEARN PROGRAMMING?
Start your programming journey with Programiz AT NO COST.
Supul Heshan
main.cpp
Run
Output
Clear

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

//heapify the tree
void heapify(int array[], int n, int root)
{
    //initialize
    int largest = root;

    int left = 2 * root + 1;

    int right = 2 * root + 2;

    if (left < n && array[left] > array[largest])
        largest = left;

    if (right < n && array[right] > array[largest])
        largest = right;

    if (largest != root)
    {
```

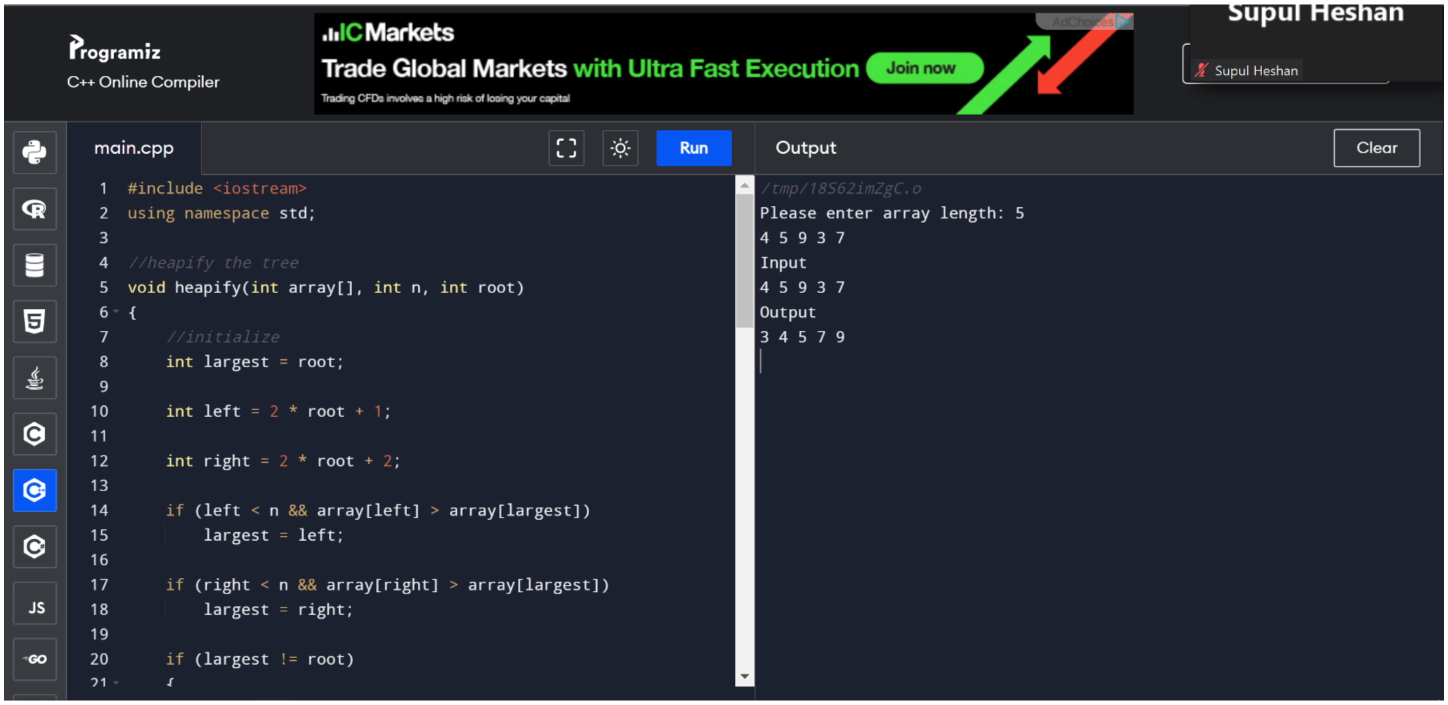
```
/tmp/18S62imZgC.o
Please enter array length: 10
4 6 3 65 34 532 4 7 3 3
Input
4 6 3 65 34 532 4 7 3 3
Output
3 3 3 4 4 6 7 34 65 532
```



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

//heapify the tree
void heapify(int array[], int n, int root)
{
    //initialize
    int largest = root;

    int left = 2 * root + 1;

    int right = 2 * root + 2;

    if (left < n && array[left] > array[largest])
        largest = left;

    if (right < n && array[right] > array[largest])
        largest = right;

    if (largest != root)
    {
```

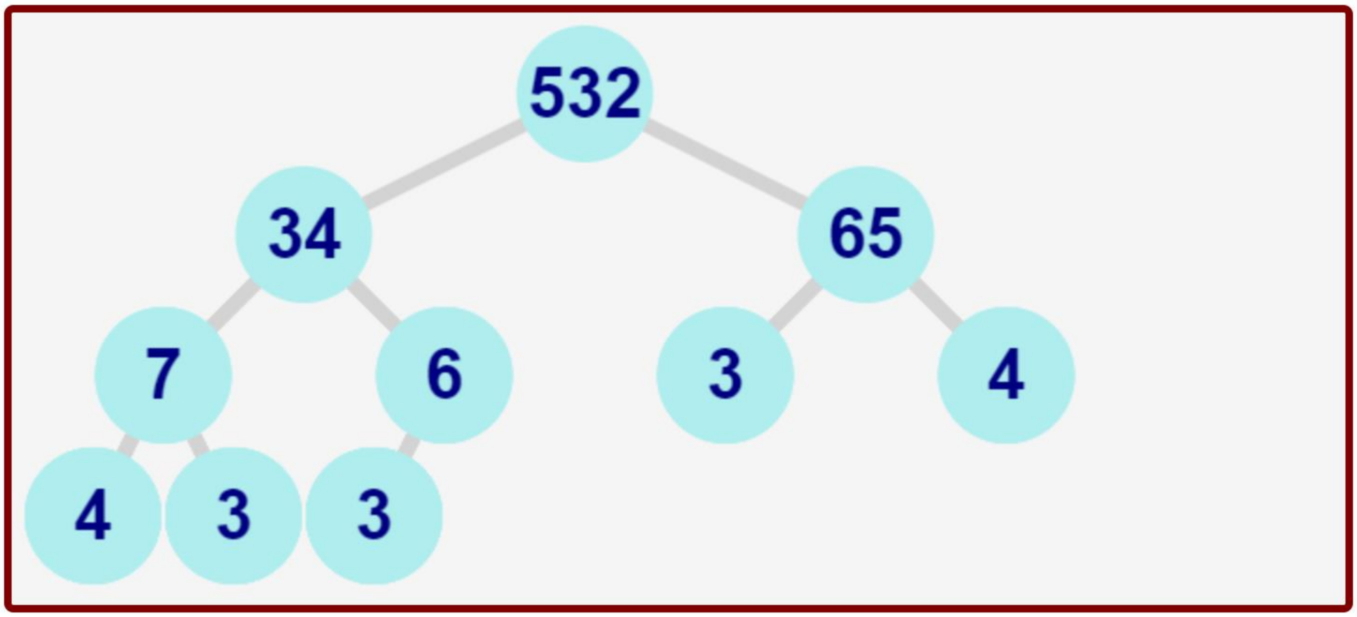
```
/tmp/18S62imZgC.o
Please enter array length: 5
0 0 2 6 0
Input
0 0 2 6 0
Output
0 0 0 2 6
```

200650U – Thilakarathne D.L.J - CS2023 – Lab 8



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

//heapify the tree
void heapify(int array[], int n, int root)
{
    //initialize
    int largest = root;

    int left = 2 * root + 1;

    int right = 2 * root + 2;

    if (left < n && array[left] > array[largest])
        largest = left;

    if (right < n && array[right] > array[largest])
        largest = right;

    if (largest != root)
    {
```

```
/tmp/1SS62imZgC.o
Please enter array length: 5
4 5 9 3 7
Input
4 5 9 3 7
Output
3 4 5 7 9
```



Heap sort is a sorting algorithm that utilizes a binary heap data structure to rearrange the input array. This technique involves iteratively exchanging the heap's root element with the last element in the heap and subsequently performing heapification on the remaining elements until the entire array is sorted. The algorithm boasts a time complexity of O(n log n), where n represents the number of elements in the input array. Additionally, it exhibits an optimal space complexity of O(1). However, it should be noted that Heap sort's efficiency is maximized when sorting large datasets, as its overhead can lead to slower sorting times when processing small datasets in comparison to other sorting algorithms.