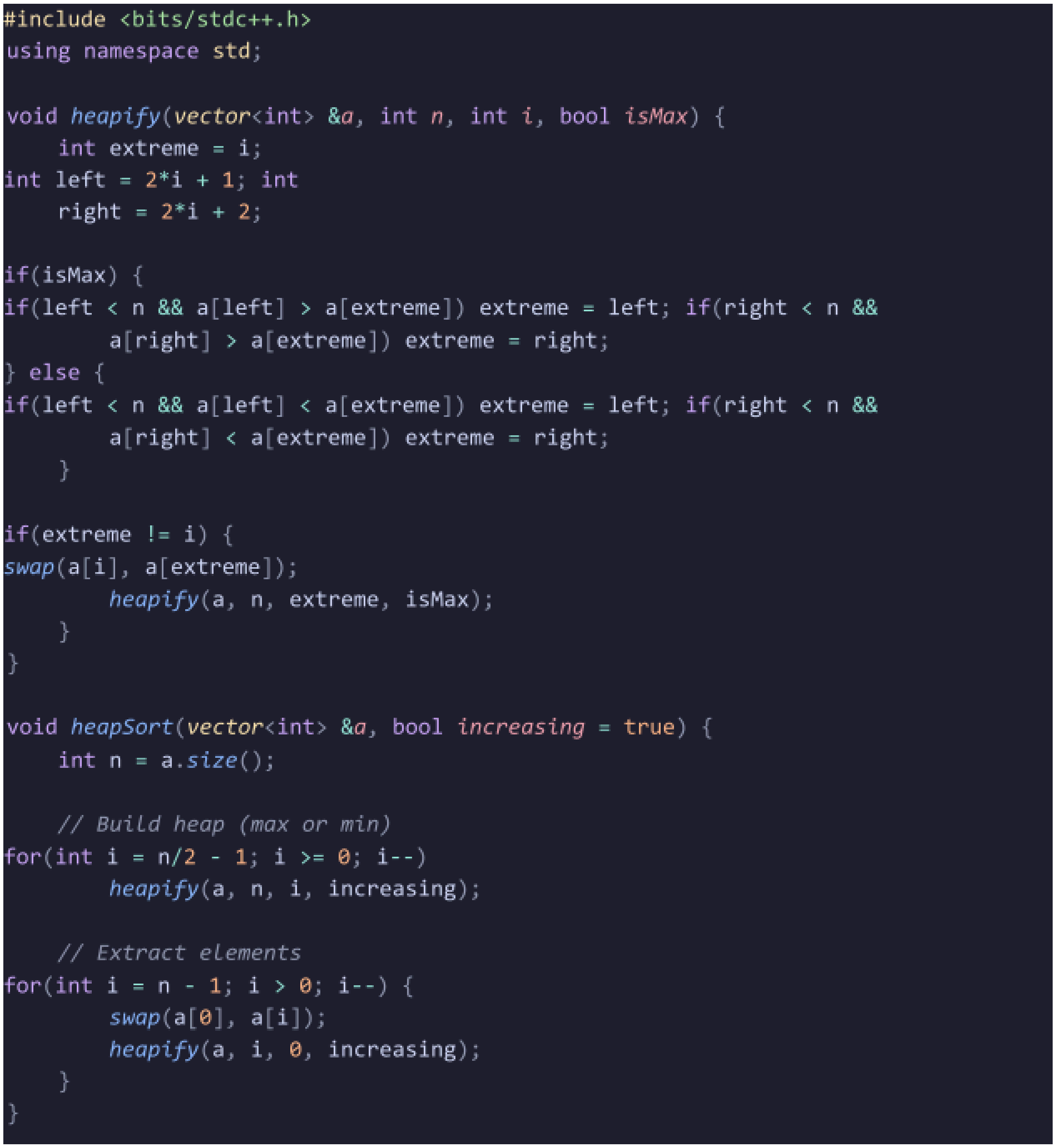
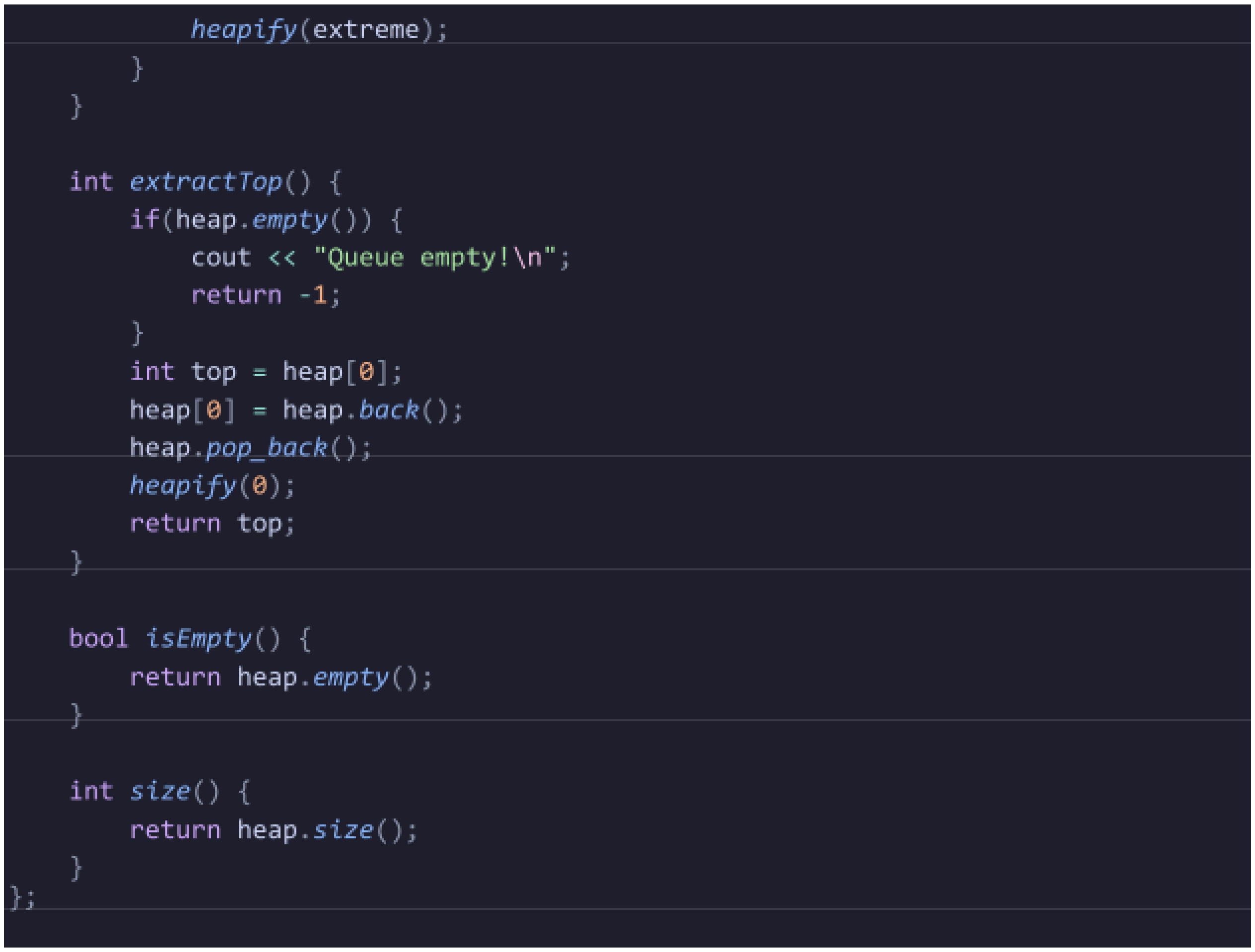
Data Structures (UCS301) Assignment

NAME-Suneet Arora **ROLL NO-1024030174**



```cpp
#include <bits/stdc++.h>
using namespace std;

void heapify(vector<int> &a, int n, int i, bool isMax) {
    int extreme = i;
int left = 2*i + 1; int
    right = 2*i + 2;

if(isMax) {
if(left < n && a[left] > a[extreme]) extreme = left; if(right < n &&
        a[right] > a[extreme]) extreme = right;
} else {
if(left < n && a[left] < a[extreme]) extreme = left; if(right < n &&
        a[right] < a[extreme]) extreme = right;
    }

if(extreme != i) {
swap(a[i], a[extreme]);
        heapify(a, n, extreme, isMax);
    }
}

void heapSort(vector<int> &a, bool increasing = true) {
    int n = a.size();

    // Build heap (max or min)
for(int i = n/2 - 1; i >= 0; i--)
        heapify(a, n, i, increasing);

    // Extract elements
for(int i = n - 1; i > 0; i--) {
        swap(a[0], a[i]);
        heapify(a, i, 0, increasing);
    }
}
```

```cpp
class HeapPriorityQueue {
    vector<int> heap;
    bool isMax; // true = max-heap, false = min-heap

public:
    HeapPriorityQueue(bool isMaxHeap = true) {
        isMax = isMaxHeap;
    }

    int compare(int a, int b) {
        return isMax ? (a > b) : (a < b);
    }

    void insert(int val) {
        heap.push_back(val);
        int i = heap.size() - 1;

        while(i > 0) {
            int parent = (i - 1) / 2;
            if(compare(heap[i], heap[parent])) {
                swap(heap[i], heap[parent]);
                i = parent;
            } else break;
        }
    }

    int getTop() {
        if(heap.empty()) {
            cout << "Queue empty!\n";
            return -1;
        }
        return heap[0];
    }

    void heapify(int i) {
        int n = heap.size();
        int extreme = i; int
        left = 2*i + 1; int
        right = 2*i + 2;

        if(left < n && compare(heap[left], heap[extreme])) extreme = left;
        if(right < n && compare(heap[right], heap[extreme])) extreme = right;

        if(extreme != i) {
            swap(heap[i], heap[extreme]);
```



```cpp
            heapify(extreme);
        }
    }

    int extractTop() {
        if(heap.empty()) {
            cout << "Queue empty!\n";
            return -1;
        }
        int top = heap[0];
        heap[0] = heap.back();
        heap.pop_back();
        heapify(0);
        return top;
    }

    bool isEmpty() {
        return heap.empty();
    }

    int size() {
        return heap.size();
    }
};
```