COMP 2611 – Data Structures

Lab #6

Part 1: Binary Trees

Download the `Lab6-Part1-Files.zip` file from myeLearning. The zipped file contains a Dev-C++ project, `BinaryTree.dev`. Open the project.

(a) Write code in `UsingBinaryTree.cpp` to create the binary tree in Figure 1. Code has already been written to create the nodes. All you need to do is connect the nodes to their parents.

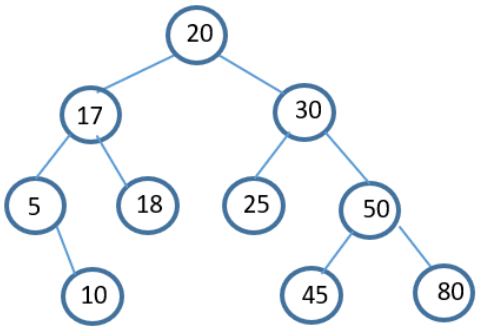


Figure 1: Binary Tree

(b) In `BinaryTree.cpp`, write the function *levelOrder* which performs a level-order traversal of a binary tree. The project includes the necessary `.h` and `.cpp` files to create and use a queue. The prototype of *levelOrder* is as follows:

```
void levelOrder (BTNode * root);
```

You can use the following algorithm to write the *levelOrder* function:

```
initialize a queue, Q
insert the root of the BT in Q
while (Q is not empty) {
      set p = dequeue (Q)
      visit p
      if left (p) is not null
            insert left(p) in Q
      if right (p) is not null
            insert right (p) in Q
}
```

(c) Write code in UsingBinaryTree.cpp to test the *levelOrder* function. Use the binary tree in Figure 1 for testing.

Part 2: Heaps

Download the `Lab6-Part2-Files.zip` file from myeLearning. The zipped file contains a Dev-C++ project, `MaxHeap.dev`. `MaxHeap.h` contains the declaration of a max-heap:

```
struct MaxHeap {
      int A [MAXHEAPSIZE];      // array to store elements of the heap
      int size;                 // number of elements in the heap
};
```

(1) `MaxHeap.h` contains declarations for several functions that operate on a max-heap. Some of these functions have already been written in `MaxHeap.cpp`. You must write the others.

(a) `MaxHeap * initMaxHeap();`

- Already written.
- Initializes a max-heap and returns the address of the max-heap.

(b) `MaxHeap * initMaxHeapFromArray (int A[], int numElements);`

- Already written.
- Initializes a max-heap and populates its elements with the elements of the array *A*. *numElements* is the number of elements in *A*.

(c) `MaxHeap * initMaxHeapFromFile (char filename[]);`

- Already written.
- Initializes a max-heap and populates its elements with data from the file specified. Assume that the data in the file is terminated by -999.

(d) `void displayMaxHeap (MaxHeap * heap);`

- Displays the values of the max-heap (as if a level-order traversal was being performed).

(e) `int sizeMaxHeap (MaxHeap * heap);`

- Returns the number of elements in the max-heap.

(f) `bool isEmptyMaxHeap (MaxHeap * heap);`

- Returns *true* if the max-heap is empty and *false*, otherwise.

(g) `bool maxHeapPropertyHolds (MaxHeap * heap, int i);`

- Returns *true* if the max-heap property holds for node *i* and *false*, otherwise.

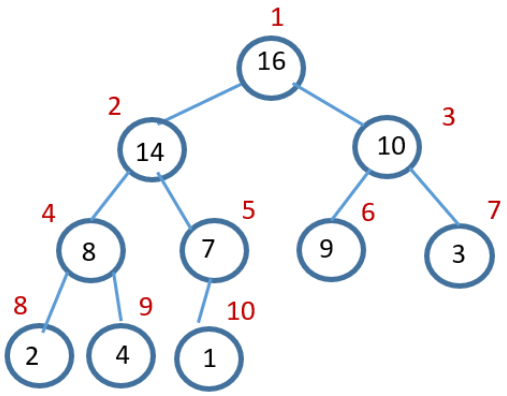
(h) bool isMaxHeap (MaxHeap * heap);

- Using the *maxHeapPropertyHolds* function, returns *true* if *heap* is indeed a max-heap and *false*, otherwise.

Hint: Check that the max-heap property holds for **all** the nodes in *heap*.

(2) For **each** of the following almost-complete binary trees, write code in `UsingMaxHeap.cpp` to determine if it is a max-heap and if so, display how many elements are present in the max-heap. Double-check your answers in (b), (c), and (d) by drawing the given trees.

(a)



(b) Values are stored in the file, `MaxHeap1.txt` (given).
(c) Values are stored in the file, `MaxHeap2.txt` (given).
(d) Values are stored in the file, `MaxHeap3.txt` (given).

End of Lab #6