Written Homework #4
CS 163: Data Structures

Alves Silva, Otavio Augusto
PSU ID: 902840168

1. Write the algorithm for each of the following

- **Determine if a tree is FULL**
  1. Using a recursion function, which will be called height, to calculate the height of a tree.
     a. Example: height(node * & root)
  2. The recursion function, which will determine if the tree is FULL or not, takes the root as argument.
     a. Example: isFull(node * & root)
  3. The base case is checked.
     a. If root is null, we have to return the value true because an empty tree is a full tree.
  4. The heights of each root children are checked.
     a. Example: left_node = height(root->left)
     b. Example: right_node = height(root->right)
  5. Return the following statements with the logical operator AND.
     a. We check if the height is the same.
        - Example: left_node == right_node.
     b. Call the function isFull recursively for each node.
        - Example: isFull(root->left)
        - Example: isFull(root->right)
  6. Therefore, the smaller sub-problem will return the following statement
     a. Return (left_node == right_node && isFull (root->left) && isFull (root->right).

  If the following algorithm returns the value 1, the tree is a FULL tree

- **Determine if a tree is a complete tree**

  1. Using a recursion function, which will be called height, to calculate the height of a tree.
     a. Example: height(node * & root)
  2. The recursion function, which will determine if the tree is complete or not, takes the root as argument.
     a. Example: isComplete(node * & root)
  3. The base case is checked.
     a. If root is null, we have to return the value true because an empty tree is a complete tree.

4. The heights of each root children are checked.
    a. Example: left_node = height(root->left)
    b. Example: right_node = height(root->right)
5. Return the following statements with the logical operator AND.
    a. We check if the height is the same.
        i. Example: left_node == right_node.
    b. Check if the node has a right node and doesn't have a left node.
        i. Example: if(root->right && !root->left)
            1. If is true, the Tree is not complete, and return zero.
        ii. Else, we have to check if the following nodes have the same height..
            1. Example: return (left_node == right_node && isComplete(root->left) && isComplete(root->right).

If the following algorithm returns the value 1, the tree is a complete tree.

2. Use gdb

I created a pointer of character, which should be an array of characters, to create a segmentation fault by passing a group of characters without have done an allocation for this. With the gdb, I use some breakpoints that helped find the exact location of my mistake. I think that gdb would have an step-by-step flow of the program that would help the programmer to see exactly what is happening as well as the address of the memory to keep track of the stack.

3. Deleting from a BST

The three cases for removing an item from a BST are:

1. A node that doesn't have a children.
    a. We can use both approaches. However, recursion can be easily used, we have just to change the parent pointer to NULL.
2. A node that have one child.
    a. We can use the both approaches. However, recursion can be easily used, we have just to let the parent of the node deleted adopt the child.
3. A node that have two children.
    a. We have to do it iteratively, because we have to find a node(Inoder successor) to replace the deleted node. However, to do it we have to use a loop until the left pointer is NULL. Therefore, the iteration is better away to complete this quest.

4. Using recursion with classes

- Wrapper function: It's a function that has the main purpose of call a second function. Normally, it's done when the second function doesn't have permissions to access some variables or other functions.

- If a wrapper function is used by a class or ADT. It has to be placed in the public section and the functions, which the wrapper will call, are placed in the private section.