6.1 A Simple BST

Due 13 Oct 2020 by 23:59 **Points** 0 **Submitting** an external tool

For this assignment, you write a simple binary search tree (BST). (compare zyBook Chapter 7.4) We keep it simple by just storing *int* values in each tree node, and by not balancing our tree. Neither do we delete nodes. More precisely, you are supposed to implement a class BST with the following interface:

```
class BST{
    public:
        BST();
        ~BST();
        void insertKey(int newKey);
        bool hasKey(int searchKey);
        std::vector<int> inOrder();
        int getHeight();
    private:
        // your implementation goes here
};
```

Here, the function *inOrder()* returns a vector containing all numbers stored in the tree according to an in-order traversal. *getHeight()* returns the height of the tree; this is 0 for an empty tree and 1 for a tree with 1 node. (For more nodes, it depends on the tree structure.)

For your implementation, you will need another class for the nodes in the tree, containing an *int* value and two pointers to the left and right subtree. You can submit your program as one file or as many files.

Next to the *BST* class, write a *main()* program that reads a sequence of numbers from the user (ended by a single letter). These numbers must be stored in the BST; call *insertKey()* for the numbers in the order in which they appear in the user input. Next, ask the user for a number to be searched in the BST, and print whether or not the number is stored in the tree. Then, the numbers stored in the BST have to be printed in sorted order. Finally, the height of the tree must be printed. (Then, we have exercised the whole interface of *BST*.) There is no need for error handling on the user input.

A few correct executions of the program are shown here:

```
Enter the numbers to be stored (end with a letter): 8 4 6 1 5 q
Which number do you want to look up? 2
2 is in the tree: no
The numbers in sorted order: 1 4 5 6 8
Height of the tree: 4

Enter the numbers to be stored (end with a letter): 1 2 3 4 5 q
```

```
Which number do you want to look up? 2
2 is in the tree: yes
The numbers in sorted order: 1 2 3 4 5
Height of the tree: 5

Enter the numbers to be stored (end with a letter): q
Which number do you want to look up? 2
2 is in the tree: no
The numbers in sorted order:
Height of the tree: 0
```

One important thing: your BST must **no**t create **memory leaks**. This means that your destructor must *delete* all objects that have been created with *new* while keys have been inserted. Your TA will have a look at this and might ask you to fix your memory leaks before your program can get accepted.

The keys stored in the BST must be stored in node objects that have been dynamically created using *new*. No other means of storage are allowed. (That is, no container structure from the C++ STL library nor any other, non-dynamic storage. The use of arrays for this assignment will be punished by public stoning!)

This tool was successfully loaded in a new browser window. Reload the page to access the tool again.