CSE 2020 LABORATORY -- Weeks 8 and 9 – Spring 2021

Instructor: Kerstin Voigt

Exercise 1: Implement class BinarySearchTree in a header file BinarySearchTree.h as detailed in the lecture video (and/or Weiss's textbook in Chapter 4). Pay particular attention to those member functions that apply recursion, as they are good examples to motivate and teach a recursive approach to processing a data structure. Take the time to understand the details of the insert and remove functions.

In order to print a more "structural" representation of a BinarySearchTree object, add the following member functions to the class:

In the public interface of class BinarySearchTree:

```
void printInternal() const
{
    if (isEmpty())
        cout << "Empty tree" << endl;
    else
        printInternal(root, 0);
}
```

In the private section of class BinarySearchTree:

```
void printInternal(BinaryNode *t, int offset) const
{
    for (int i = 1; i <= offset; i++)
        cout << "..";
    if (t == 0)
    {
        cout << "#" << endl;
        return;
    }
    cout << t->element << endl;

    printInternal(t->left, offset + 1);
    printInternal(t->right, offset + 1);
    return;
}
```

Exercise 2: Test your class BinarySearchTree{} with an implementation of the int main() that you have seen in the lecture video. In this program, a user specified number of randomly generated integers are inserted into a BinarySearchTree, the content of the BST is printed, and 5 values are removed from the structure. After the removals, the reduced BST is printed again.

Due to the random generation of sample data, expect your BST to look different from the trees you have seen in the video.

TO OBTAIN CREDIT FOR THIS LAB

By Friday, March 26, at 11:59pm submit via Blackboard portal under Blackboard portal the following files: (1) Copy of your header file BinarySearchTree.h, (2) a copy of your file with the 'int main()' function (e.g., BinarySearchMain.cpp), (3) copy of a screenshot of your running program. Make sure that all files are plain .h or .cpp files , no formatted text please. Please avoid submitting zipped folders.

Failure to compile? - In case you did not manage to produce a program that compiles error free (despite your utmost effort), submit the request code files in their current form and (2) a screenshot of the compilation attempt and resulting errors.

File naming: For ease identification of your submitted work, make sure that you adhere to the following **file naming convention:** for each file XYZ.h , or XYZ.cpp you plan to submit, make a copy of this file with new name

Lastname_Firstname_ABCD_XYZ.h (.cpp)

where ABCD are the last four digits of your student id. **Your work will not be graded if you do not adhere to this naming convention.**