CSE 330 Homework Assignment 3 , Spring 2018

Instructor: Kerstin Voigt

(Total number of points: 25)

In Lab7 we added code to our BinarySearchTree.h file so that it would be ready for the addition of iterators. The plan is to use the binary search tree structure for the implementation of ADT Set, and for this purpose, the BinarySearchTree class needs to have its own iterator. The implementation of iterators for the BinarySearchTree will share many similarities with the implementation of list iterators in List.h.

At the end of Lab7, everyone should have left an enhanced version of the binary search tree implementation in a file **`BinarySearchTreeLab7.h.`** If you do not have your own file with code that compiles, you may download a copy from the HW3 area on Blackboard.

For this homework, you will be **filling the missing code** into the following member functions of **class iterator** (within class BinarySearchTree):

```
bool is_root(BinaryNode * t)
```

- returns true when t is a pointer to the BinaryNode that is the "root"; the root is the only BinaryNode that has nullptr as its parent;

```
bool is_left_child(BinaryNode * t)
```

- returns true when t is a pointer to a BinaryNode that is the left child of its parent; test whether t's parent's left child is the same as t;

```
bool is_right_child(BinaryNode * t)
```

- analogous to is_left_child;

```
BinaryNode * leftmost(BinaryNode * t)
```

- starting at t, follow the left children and return a pointer to the deepest leftmost child;

```
BinaryNode * follow_parents_until_leftchild(BinaryNode * t)
```

- starting at t, follow the parent links upwards until a BinaryNode is reached which is a left child; return a pointer to this left child's parent;

Implement the bodies of these functions so that they behave as described (3 points for each function).

Next, we will complete the implementation of the BinarySearchTree iterator by filling in the body of member function **operator ++() of class iterator**. Today's lecture will have prepared you for this task which is rather complex and should be approached "one case" at a time. You will do this by implementing the following **algorithm:**

Let `current` be a pointer to the BinaryNode of the BinarySearchTree that is currently in focus. Operator ++() updates `current` as follows:

Examine `current` …

Case 1: if current is the root node, then, if current has right child, set current to the leftmost node that can be reached from this right child; if current does not have a right child, current is set to nullptr and iterator stops (the "end" has been reached).

Case 2: if current is a left child, distinguish two cases: (a) if current is a leaf (no left and no right child), set current to current's parent; (b) if current has a right child, set current to the leftmost node that can be reached from this right child;

(the case of current having a left child and no right child need not be considered; iteration will never move to this node)

Case 3: if current is a right child, distinguish two cases: (a) if current has a right child, then set current to the leftmost node that can be reached from current's right child; (b) if current does not have a right child, move up the parent links until a node is reached that is a left child; set current to this left child's parent.

After having updated current according to one of these three cases, operator ++() returns the iterator object itself (with: return *this;)

(10 points for operator ++)

Test your iterator implementation with the following int main() in a file that you may want to call HW3.cpp (do not forget the add the necessary #include's):

```
int main()
{
      BinarySearchTree<int> mybst;
      int next;
      for (int i = 1; i <= 10; i++)
      {
            cout << "Integer: ";
            cin >> next;
            cout << endl;
            mybst.insert(next);
      }
      cout << endl << "Values entered" << endl;

      mybst.printTree();
      cout << endl;
      mybst.printInternal();
      cout << endl << endl;

      cout << "And with iterators ..." << endl;
      BinarySearchTree<int>::iterator itr = mybst.begin();

      for (; itr != mybst.end(); ++itr)
            cout << *itr << endl;
      cout << endl << endl;

      cout << "Now doing some removals ..." << endl;
      for (int i = 1; i <= 3; i++)
      {
            cout << "Remove? ";
            cin >> next;
            cout << endl;
            mybst.remove(next);
      }

      cout << endl;
      mybst.printTree();
      cout << endl;
      mybst.printInternal();
      cout << endl << endl;

      cout << "And with iterators ..." << endl;
      itr = mybst.begin();
      for (; itr != mybst.end(); ++itr)
            cout << *itr << endl;
      cout << endl << endl;

      return 0;
}
```

Submit on Thursday, May 24, 2018, at the beginning of the lecture: (1) A hardcopy of your completed file BinarySearchTree.h, (2) a copy of HW3.cpp, just for the sake of completeness, and (3) a typescript or screenshot that demonstrates the successful compilation and running of your int main().

Should you not get the point of having program that compiles without error, still submit items (1) and (2), and include a typescript or screenshot of the compilation attempt and error listing.