COMP 2140 Assignment 4: Binary Search Trees

Robert Guderian and Helen Cameron

Due: Friday, November 23, 2018 at 4:30 p.m.

Hand-in Instructions

Go to COMP2140 in UM Learn; then, under "Assessments" in the navigation bar at the top, click on "Assignments". You will find an assignment folder called "Assignment 4". Click the link and follow the instructions. Please note the following:

- Submit ONE .java file file only. The .java file must contain all the source code that you wrote. The .java file must be named `A4<your last name><your first name>.java` (e.g., `A4CameronHelen.java`).
- Please do not submit anything else.
- We only accept homework submissions via UM Learn. Please DO NOT try to email your homework to the instructor or TA or markers — it will not be accepted.
- We reserve the right to refuse to grade the homework or to deduct marks if you do not follow instructions.
- **Assignments become late immediately after the posted due date and time.** Late assignments will be accepted up to 49 hours after that time, at a penalty of 2% per hour or portion thereof. After 49 hours, an assignment is worth 0 marks and will no longer be accepted.

How to Get Help

Your course instructor is helpful: For our office hours and email addresses, see the course website on UM Learn (on the right side of the front page).

For email, please remember to put "[COMP2140]" in the subject and use a meaningful subject, and to send from your UofM email account.

Course discussion groups on UM Learn: A discussion group for this assignment is available in the COMP 2140 course site on UM Learn (click on "Discussions" under "Communications"). Post questions and comments related to Assignment 3 there, and we will respond. Please do not post solutions, not even snippets of solutions there, or anywhere else. We strongly suggest that you read the assignment discussion group for helpful information.

Computer science help centre: The staff in the help centre can help you (but not give you assignment solutions!). See their website at http://www.cs.umanitoba.ca/undergraduate/computer-science-help-centre.php for location and hours. You can also email them at helpctr@cs.umanitoba.ca.

Programming Standards

When writing code for this course, follow the programming standards, available on this course's website on UM Learn. Failure to do so will result in the loss of marks.

Question

Remember: You will need to read this assignment many times to understand all the details of the program you need to write.

Goal: To execute ”programs” in a mini ”programming language” (Lenert) that has only integer variables, assignment statements and simple integer expressions.

Code you can use: You are permitted to use and modify the code your instructor gave you in class for binary search trees (BSTs) and their nodes. You are NOT permitted to use code from any other source, nor should you show or give your code to any other student. Any other code you need for this assignment, you must write for yourself. We encourage you to write ALL the code for this assignment yourself, based on your understanding of BSTs — you will learn the material much better if you write the code yourself.

The Programming Language Lenert

The purpose of a Lenert program is to compute values and store the values in variables. The output of a Lenert program is the values in the variables at the end of the program.

Here’s an example Lenert program

```
length = 13
width = 10
area = length * width

extension = 14
fullarea = area + extension

t1 = 13 + 63
t1 = t1 / 4
t1 = 65 - t1
```

A Lenert program consists of a sequence of **assignment statements.** Each assignment statement takes up one line — there are no multi-line assignment statements in Lenert. An assignment statement consists of a variable name (described below), followed by an equals sign (`=`), followed by the right side of the assignment statement, which can have one of the following two forms:

- It can consist solely of one operand, which can be either an integer constant or a variable name.
- It can consist of an operand, followed by an operator (`'+'`, `'-'`, `'*'`, or `'/'`), followed by a second operand, where each operand is either an integer constant or a variable name.

As you can see, **variable names** are strings of length at least one beginning with a letter and containing only letters and numbers (all letters are lower case). All variables are integer variables.

A variable is “declared” when it first appears on the left side of an assignment statement. In other words, because there are no separate variable declarations, only assignment statements, a variable declaration is an assignment statement that assigns a value to the variable for the first time. So, the first time a variable appears should be on the left side of an assignment statement. If a variable appears on the right side of an assignment statement before it has had a value assigned to it, then we say that the variable is being used before it was declared and that assignment statement contains an error.

For example, consider the following Lenert program:

```
x = 3
r = x
x = x + t
t = 36
```

Variables `x` and `r` are properly declared before they are used, but variable `t` is used before it is declared. Thus, the assignment statement `x = x + t` is erroneous.

Your Program:

You must write a program that reads in and executes Lenert programs.

Your program will use a `Table` to store "variable records". Each variable record contains a variable name and its associated value, with the variable name being the key of the variable record. The Table must be implemented as a binary search tree (BST), with the variable name being the key. (More on the Table below.)

For each assignment statement in a Lenert program:

- You must first compute the value of the right side of the assignment statement.
  - There may be none, one, or two variables on the right side. For example, `x = 3` has none, `x = 32 + t` has one, and `x = t * y` has two.
  - For each of the variables on the right side, you must find the variable in the Table to retrieve its value.
  - For an integer constant, use `Integer.parseInt()` to retrieve its value.
  - If there's an operator on the right side, then, once you have the values of its operands, you must perform the operation on the values of the two operands, to get the value of the right side.
- Then you must search in the Table for the variable on the left side of the assignment statement.
  - If you find the variable, then you change its value to the value you computed for the right side of this assignment statement. For example, consider the third assignment statement in the following:

    ```
    x1 = 23
    y = x1 + 2
    y = y * 5
    ```

    The variable on the left side, `y`, has already been declared in the previous statement, so it will already be in the Table. Thus, you will find it in the Table, and simply change the value stored with it to 125 (the value of the right side).
  - If you don't find the variable in the Table, then this assignment statement is this variable's declaration. You must add the new variable to the Table along with the value you computed for it on the right side of the assignment statement. For example, consider the third assignment statement in the following:

    ```
    x2 = 23
    y2 = x2 + 2
    z = y2 * 5
    ```

    The variable on the left side, `z`, has not yet been declared, so this assignment statement is its declaration. You will not find it in the Table, so you will have to insert it along with its value given by the right side (125).
- If a variable on the right side is not in the Table, then it has not been declared before being used, so you should print an error message, abandon this assignment statement, and go on to the next assignment statement.

When you have executed all the assignment statements in a Lenert program, you should print out all the variable names and their values.

Table Class:

You must create a Table class that is implemented as a binary search tree.

You must implement the following methods:

- A constructor, which creates a new, empty Table.
- `search(String variableName)`, which returns the variable's name and its associated value, or `null` if the variable is not in the Table.
- `insert(String variableName, int variableValue)`, which inserts the given variable and its associated value into the Table.
- `printTable()`, which simply goes through the table and prints out the variables stored in it. Each variable should be printed on a separate line containing the variable name followed by its value.

You may, of course, create other methods (and classes) as needed.

The Input:

You should prompt the user for the name of the file to use as input. (The sample output shows that keyboard input is used by the sample solution.)

The input file will contain multiple Lenert programs.

The first line of the input file will contain the number of Lenert programs contained in the file.

Each Lenert program will contain a number of assignment statements, one assignment statement per line. (There may also be blank lines anywhere in a Lenert program; simply ignore them.) You may assume that the integer constants, variable names, operator and equals sign within an assignment statement are separated from each other by at least one blank. (There may be multiple blanks between them.)

The end of a Lenert program will be indicated by a line containing a `'Q'` (and no other letters, numbers, operators, or equals signs). Notice that the `'Q'` is uppercase, which differentiates it from a Lenert variable, which contains only lowercase letters or numbers.

We will provide a larger input file, called "`A4data.txt`", a few days before the due date. In the meantime, you can use the smaller input file "`A4testData.txt`" to get your code working. (You can find both of these files in the "Assignments" section of the content browser on course web site in UM Learn.)

The Output:

The output should begin with an appropriate opening message.

Then, for each Lenert program, the output should contain an appropriate heading (number the programs), followed by any error messages generated by variables being used before they were declared, followed by a print-out of the final values of the variables after the program has been executed.

Finally, the output should end with an appropriate closing message.

For example, the output might look like the following:

```
COMP 2140 Assignment 4: Executing Lenert Programs
-------------------------------------------------


Enter the input file name (.txt files only):
A4testData.txt



Lenert Program 1
----------------
Error messages:
```

```
Final values of the variables:

    area = 130
    extension = 14
    fullarea = 144
    length = 13
    t1 = 46
    width = 10

Lenert Program 2
----------------
Error messages:
Invalid input line: variable t is used before its declaration in "x = x + t"

Final values of the variables:

    r = 3
    t = 36
    x = 3

Lenert Program 3
----------------
Error messages:

Final values of the variables:

    x = 23
    y = 125

Lenert Program 4
----------------
Error messages:

Final values of the variables:

    x2 = 23
    y2 = 25
    z = 125


---------------

Program ended normally.
```