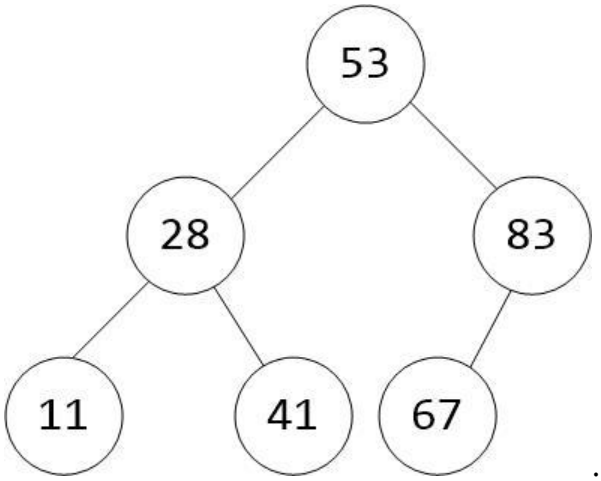
CMSC 315 Data Structures and Algorithms
Programming Project 3

The third programming project involves writing a program to read in the preorder representation of a binary tree and determine whether it is a balanced binary search tree. Given the binary tree shown below:



Its preorder representation would be `(53 (28 (11 * *) (41 * *)) (83 (67 * *) *))`. The asterisks represent null children.

The program should repeatedly prompt the user for a binary tree. It should then display the tree using indentation. For the above tree, its indented representation would be as follows:

```
53
   28
      11
      41
   83
      67
```

It should then categorize the binary tree in one of three ways:

- It is not a binary search tree
- It is a balanced binary search tree
- It is a binary search tree but it is not balanced

If the tree is not a balanced binary search tree, a binary search tree containing the same set of values should be constructed and displayed in the indented format shown above. It is not expected that the existing tree be rebalanced, just that a new tree with the same set of values be constructed. Then the height of the original tree and the balanced binary search tree should be displayed.

Shown below is a sample session that involves each of the above cases:

```
Enter a binary tree: (53 (28 (11 * *) (41 * *)) (83 (67 * *) *))
   53
      28
         11
         41
      83
         67
It is a balanced binary search tree
More trees? Y or N: Y
Enter a binary tree: (63 (51 (20 (13 * *) *) *) *)
   63
      51
         20
            13
It is a binary search tree but it is not balanced
   20
      13
      51
         63
Original tree has height 3
Balanced tree has height 2
More trees? Y or N: Y
Enter a binary tree: (13 (53 * *) (11 (59 * *) *))
   13
      53
      11
         59
It is not a binary search tree
   13
      11
      53
         59
Original tree has height 2
Balanced tree has height 2
More trees? Y or N: N
```

In addition, the program should verify that the tree input has valid syntax. Each of the following errors should be detected;

- Incomplete Tree
- Data is Not an Integer
- Extra Characters at the End
- Missing Left Parenthesis
- Missing Right Parentheses

The program should consist of three classes. The first class should be the class that defines the binary tree. It should be an immutable class with the following public methods:

- A constructor that accepts a string containing the preorder representation of a binary tree and constructs a binary tree
- A constructor that accepts an array list of integers and constructs a balanced binary search tree containing those values

- A method that outputs the binary tree in indented form
- A method that returns whether the tree is a binary search tree.
- A method that returns whether the tree is balanced
- A method that returns the height of the tree
- A method that returns an array list of the values in the tree

Additional private methods may be added as needed.

The second class should be a class that defines a checked exception that is thrown when a tree with invalid syntax is input. The third class is the class that contains the main method and accepts the user input and displays the results.

You are to submit two files.

1. The first is a `.zip` file that contains all the source code for the project. The `.zip` file should contain only source code and nothing else, which means only the `.java` files. If you elect to use a package the `.java` files should be in a folder whose name is the package name. Every outer class should be in a separate `.java` file with the same name as the class name. Each file should include a comment block at the top containing your name, the project name, the date, and a short description of the class contained in that file.
2. The second is a Word document (PDF or RTF is also acceptable) that contains the documentation for the project, which should include the following:
    a. A UML class diagram that includes all classes you wrote. Do not include predefined classes.
    b. A test plan that includes test cases that you have created indicating what aspects of the program each one is testing
    c. A short paragraph on lessons learned from the project