CPSC 331 (Spring 2023)
Assignment 3
Binary Trees
(8% of final mark)
Due: Monday May 29 @ 11:59PM

Lead TA: The lead TA for this assignment is Mahtab Movahedian Attar (mahtab.movahedianatt@ucalgary.ca)

Objectives
1. Understand the properties of binary trees.
2. Understand the practicality of binary search trees.

Part I – Extending the BST Class
Starting from the BST class provided on D2L, add the following methods to the class. You may not change the implementation provided except as suggested in this assignment. You just need to add these methods:

1. `int height()`: returns the height of the tree
2. `Node parent(T value)`: returns a reference to the parent of a node containing value
3. `int level(T value)`: returns the level of the node containing value. If there is more than one node with the same value, it returns the highest level
   - You may need to change the Node class to add a Boolean "visited" field
4. `boolean isComplete()`: returns true if the tree is complete; must call a recursive method `recIsComplete(root, index)`
   - For this one, note that if you number the nodes as in the array representation (root is at index i, then left child is at index $2*i + 1$and the right is at $2*i +2$.) Then, in a complete binary tree the index of any node cannot be $\geq$ number of nodes n
5. `boolean isPerfect()`: returns true if the tree is perfect
6. `boolean hasDoubles()`: returns true if the tree has duplicate values
   -You will need to assume that the values are drawn from a fixed set such as 0 to m-1 for some value m
   - You also need to capitalize on the "visited" field

7. Analyze the complexity for each of these methods by providing their Big-Oh. No justification is necessary.
8. Prove using Mathematical induction that in a complete tree T, the index of every node is less than n, where n is the number of nodes in T. You can assume in your proof that in a *perfect* tree, the index of the left-most leaf is $2^y - 1$, where y is the level of the leaf, and the index of the right-most leaf is $2^{y+1} - 2$, where y is the level of the leaf.
9. Prove that your recursive `recIsComplete()` method is correct by providing a bound function and using structural induction to show the property that `recIsComplete(root, index)` returns true if the tree with root `root` is complete; returns false otherwise.

Part II – Height of Random Binary Trees
The problem of a BST is that the tree can become imbalanced resulting in a height that would be O(n). We will run an experiment to see what heights random trees have. For each of the following tree sizes: 500, 1000, and 5000 nodes, generate 100 random trees using the BST class. Measure

the height of each of these 100 random trees. Calculate the average heigh of the 100 trees for each of the three tree sizes. In addition, find for each experiment if any of the random trees is complete, perfect, or has doubles. Discuss what you observe from these statistics.

You can use some of the methods from Part I where needed. For the purpose of this experiment, it is sufficient to use Integer for the generic type T.

Deliverables
Submit to the appropriate dropbox:

- Java source code
- A report (maximum: two typed page) that includes:
    - Your Big-Oh for the methods you added in Part I, no explanation is necessary
    - Your proof of correctness for isComplete()
    - Your stats and comments on these stats in Part II

Submission: Submit the deliverables to the appropriate dropbox. The TA may provide extra submission requirements or instructions.

Collaboration: You are advised to work on this assignment in pairs. Groups of more than two members are not allowed. You may change your partner from the previous assignment. Any discussion with your colleagues outside your team regarding the assignment must be kept at a very high level.

Late Submission Policy: Late submissions will be penalized as follows:
-12.5% for each late day or portion of a day.

Academic Misconduct: Any similarities between submissions will be further investigated for academic misconduct. Your final submission must be your own team's original work. While you are encouraged to discuss the assignment with colleagues outside your team, this must be limited to conceptual and design decisions. Code sharing by any means with colleagues that are not in your team is prohibited. This includes and is not limited to looking at someone else's paper or screen. Any re-used code of excess of 5 lines must be cited and have its source acknowledged. Failure to credit the source will also result in a misconduct investigation.

D2L Marks: Any marks posted on D2L are tentative and are subject to change (UP or DOWN).

Marking Rubric (110 points total)

Code	**Total 50 points**
Code compiles	5
Code runs	5
Height method	5
Parent Of method	5
Level method	5
Is Complete method	5
Is Perfect method	5
Has Doubles method	5
Code is modular	5
Code contains enough documentation	5
Report	**Total 60 Points**
3 cases of 100 random trees generated (5 points each)	15
Statistics calculations are correct and reported properly	9
Complexity of added methods (1 point each)	6
Thoughtfully discussed the statistics	5
Math Induction Proof	10
Proof of correctness for recIsComplete() (7.5 for bound function)	15

Submissions that do not compile will receive no points. Code that compiles but does not run will not receive more than 5/100.