|  |  |
| --- | --- |
| **Homework** | **Splay Trees & RB Trees** |
|  |  |

1. Splay trees:
   1. What is a splay operation?

*The splay operation brings a node to the root and its children closer to the root through a combination of zigs (single Left or Right rotation), zig-zigs (double Left or double Right rotation), and zig-zags (Left-Right or Right-Left Rotation).*

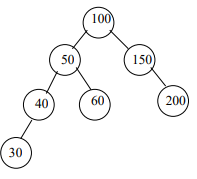
* 1. Why to prefer splay trees?

*Splay trees are particularly useful if you want to store you data somewhere else and don’t plan to access every data element equally. By using a splay tree, only the localized(regularly accessed) data will be reached in logarithmic time.*

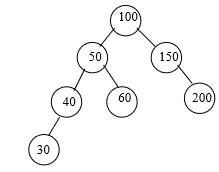
* 1. Which of the following options is an application of splay trees?
     1. cache Implementation
     2. networks
     3. send values
     4. receive values
  2. When we have red-black trees and AVL trees that can perform most of operations in logarithmic times, then what is the need for splay trees?
  3. What is the disadvantage of using splay trees?

*A side effect of the blind balancing that the splay tree is only roughly balanced. The height of the tree can be linear, which would result in worst case O(n) time complexity for search.*

1. Given the BST below, show the BST that would result after inserting the key of value 10 if splaying is performed starting at the node that was inserted.



1. Given the BST below, show the BST that would result after inserting the key of value 180 if splaying is performed starting at the node that was inserted.



1. A nice property of splay trees is that each of Find, Insert and Delete takes O(logn) time. True/False?

*False. Worst case analysis for these operations is amortized O(log(n)), since the m data elements that are regularly accessed will searched and deleted in O(log(m)) time, but the remaining k elements could take a worst case of O(log(m)+k)) time.*

1. The keys of value N, N-1, N-2..., 4, 3, 2, 1 are inserted in this order in a splay tree. What is the ﬁnal conﬁguration of the tree? What is the cost in Big-Oh notation of each insert operation?  
     
   *Linear - O(1)*
2. Assume now that the next 100 operations will be a mix of only Find(1), Find(2) and Find(3), i.e., search in the tree for either the key of value 1, or the key of value 2, or the key of value 3. After each successful search, splaying is performed from the node where the key was found. What are the 3 tree conﬁgurations that are possible after these 100 operations are completed? What is the cost in Big-Oh notation of each Find operation?
3. The next operation is Find(N). In terms of Big-Oh notation, is any of the 3 conﬁgurations above faster than the others? If so, what is its Big-Oh time complexity? If not, what is the Big-Oh time complexity of Find(N) in the three conﬁgurations?
4. Show the result of inserting 50 into the Red-Black tree depicted below:

**30**

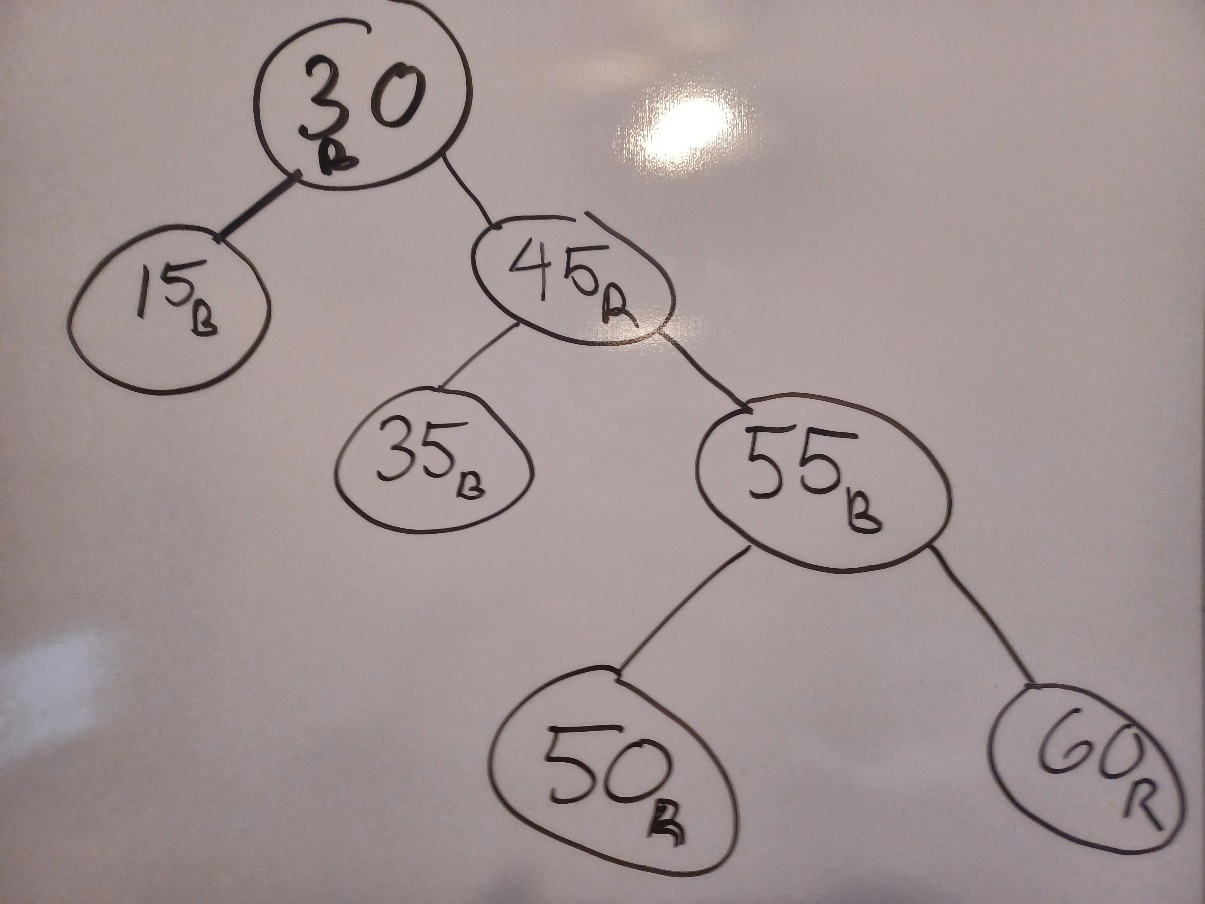
/ \

**15** *45*

/ \

**35** **60**

/

 *55*

1. Show the result of inserting 65 into the Red-Black tree depicted below:

**30**

/ \

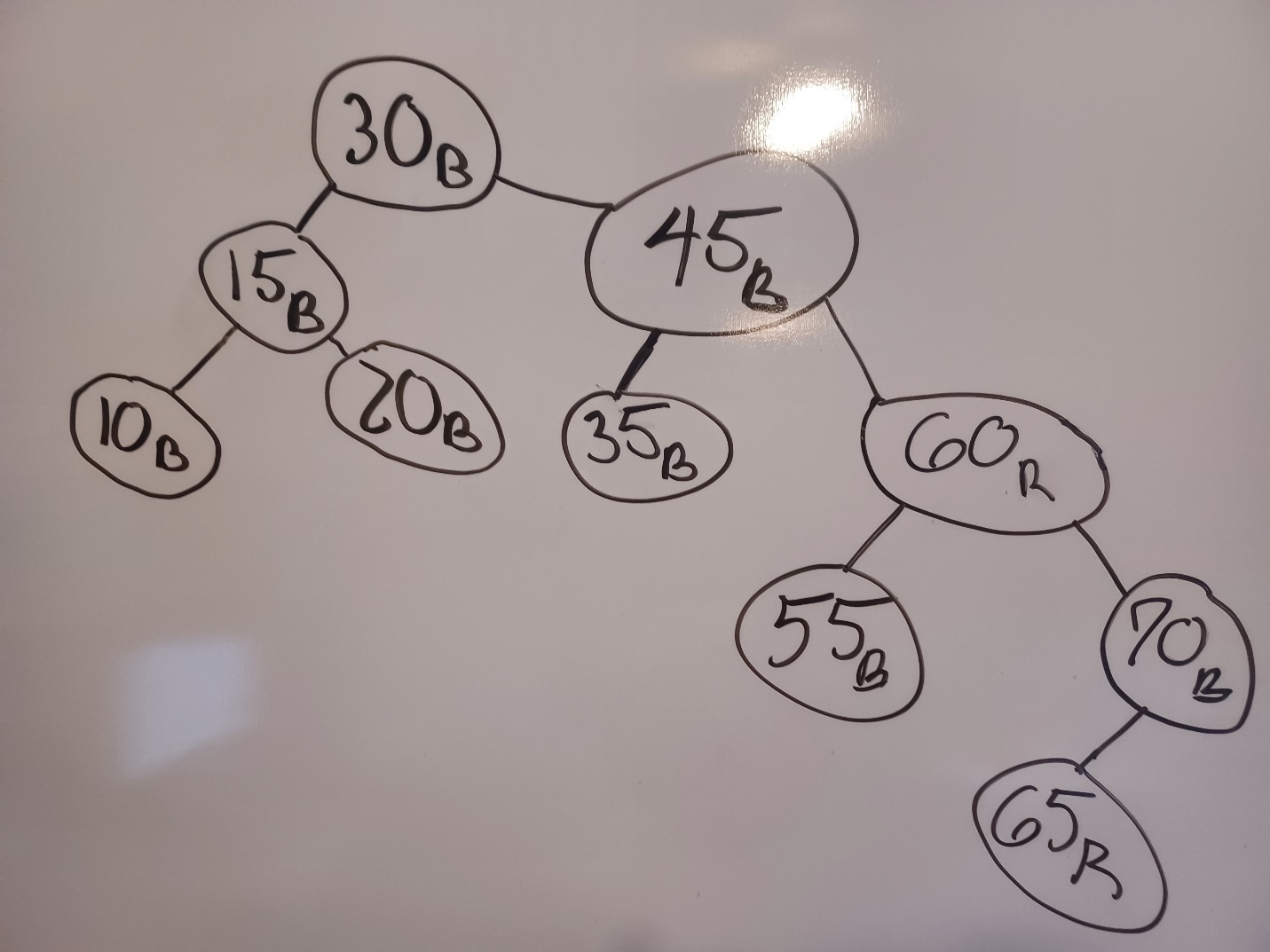
*15* *45*

/ \ / \

**10** **20** **35** **60**

/ \

*55* *70*

**

1. Review the attached SplayTreeAssignment and RedBlackTreeAssignment classes. Add comments stating the purpose of each of the methods implemented in these classes and submit your java files.
2. Execute the below operations in main method and create a flow chart for each of the listing the method names and the statements which are executed as a result of these operations.
   1. Splay tree
      1. tree.insert(1);
      2. tree.searchTree(44);
      3. tree.deleteNode(89);
   2. RB tree
      1. bst.insert(80);
      2. bst.deleteNode(25);

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **AVL Tree Implementation** | Points |
| **Part A** | No |  |  |
|  | 1 | Answers to problems 1 thru 9 | 40 |
|  | 2 | Problem 10 | 30 |
|  | 3 | Problem 11 | 30 |
|  |  |  | 100 |