**CSCD320 Homework3,Spring 2014, Eastern Washington University. Cheney, Washington.**

**Name:** Brandon Fowler **EWU ID: 00639348** **Due:** 11:59pm, April 27, 2014 (Sunday)

**Please follow these rules strictly:**

1. Write your name and EWUID on **EVERY** page of your submission.

2. Verbal discussions with classmates are encouraged, but each student must independently write his/her own solutions, without referring to anybody else's solution.

3. The deadline is sharp. Late submissions will **NOT** be accepted (it is set on the Canvas system). Send in whatever you have by the deadline.

4. Submission must be computer typeset in the **PDF** format and sent to the Canvas system. I encourage you all to use the LaTeX system for the typesetting, as what I am doing for this homework as well as the class slides. LaTeX is a free software used by publishers for professional typesetting and are also used by nearly all the computer science and math professionals for paper writing.

5. Your submission PDF file must be named as:

**firstname_lastname_EWUID_cscd320_hw3.pdf**

(1) We use the underline ' ' not the dash '-'.
(2) All letters are in the lower case including your name and the filename's extend.
(3) If you have middle name(s), you don't have to put them into the submission's filename.

6. Sharing any content of this homework and its keys in any way with anyone who is not in this class of this quarter is NOT permitted.

---

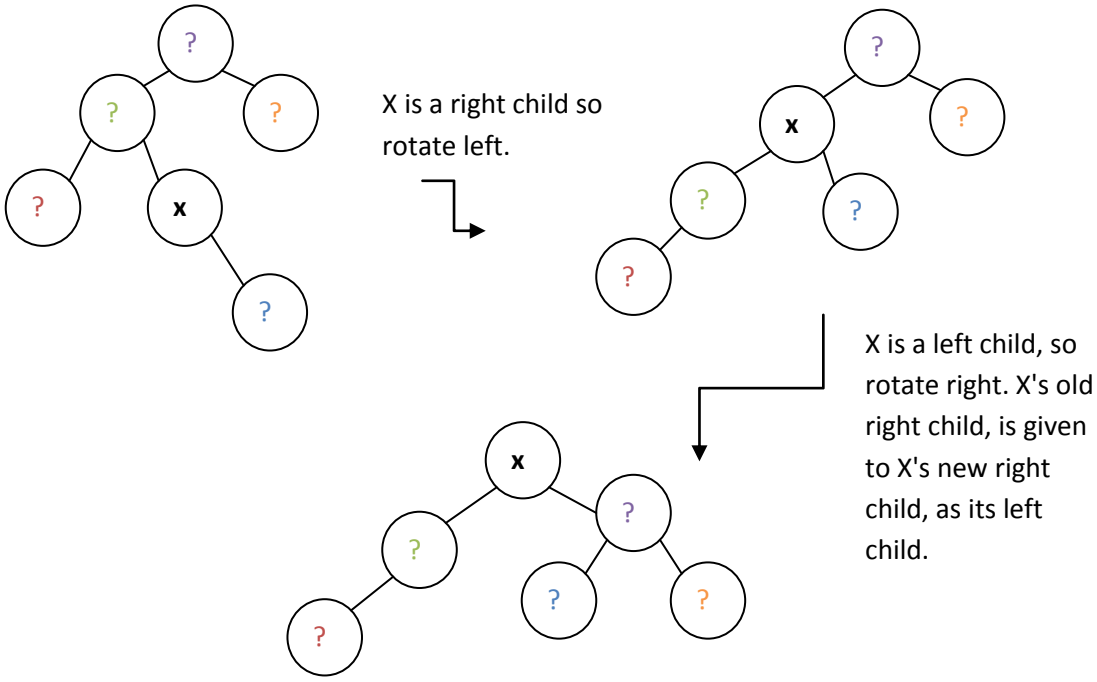
**CSCD320 Homework3,Spring 2014, Eastern Washington University. Cheney, Washington.**

**Name:** Brandon Fowler **EWU ID: 00639348** **Due:** 11:59pm, April 27, 2014 (Sunday)

**Problem 1** (25 points)**.** *Given an arbitrary node* `x` *of a binary search tree (BST), show how to re-structure the BST using* O(n) *time, so that* `x` *becomes the global root of the BST after the re-structuring. Describe and show the pseudocode of your algorithmic idea. Explain why your algorithm's time cost is* O(n)*. You can assume each node has the* `LeftChild`, `RightChild`, *and* `Parent` *links.* **Hint***: You may want to use tree rotations. If you use tree rotations, you don't have to give the detailed code of rotations, but instead just clarify what rotation on which node is being used.*

I would use tree rotation to accomplish this task. To begin I would check to see if x is a right child node or a left child node of its parent. If x is a right child node, then I would rotate x left, pulling x up, and x's parent down as x's new left child. x's previous left child, would become the right child of x's new left child. If x is a left child node of its parent, then I would rotate right, pulling x up, and x's parent down as x's new right child. x's previous right child, would become the left child of x's new right child. I would continue checking and rotating in this manner until x's parent is null, at which point x will be the root of the binary search tree. In the worst case, if the binary search tree is balanced very poorly, x may need to rotate through every other node in the tree, in order to reach the top. It therefore follows that the time cost for this algorithm is O(n).

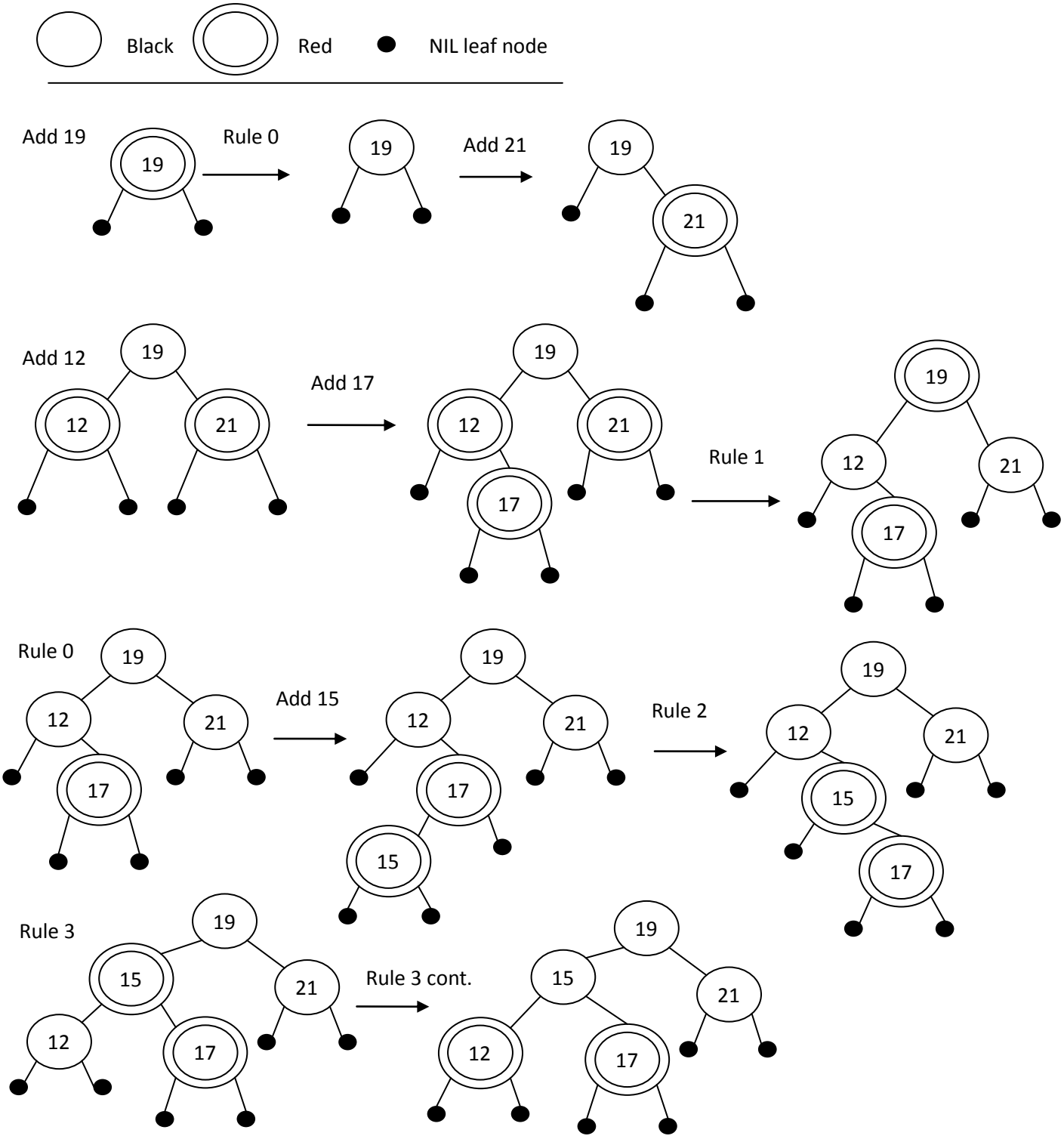
Example:



**CSCD320 Homework3,Spring 2014, Eastern Washington University. Cheney, Washington.**

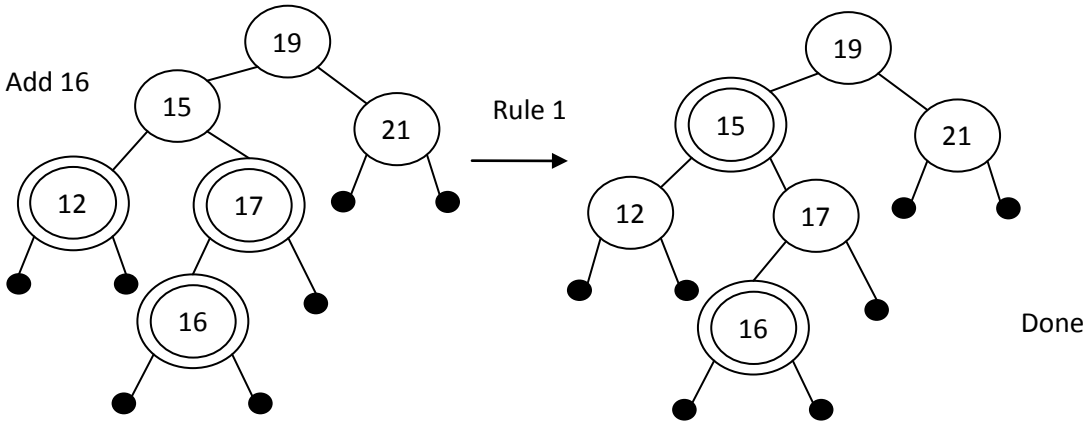
**Name:** Brandon Fowler **EWU ID: 00639348** **Due:** 11:59pm, April 27, 2014 (Sunday)

**Problem 2** (25 points)**.** *Show the trace of the construction of the Red-Black tree for the sequence* 19, 21, 12, 17, 15, 16*. That is, you need to draw the state of the tree after inserting each number.*

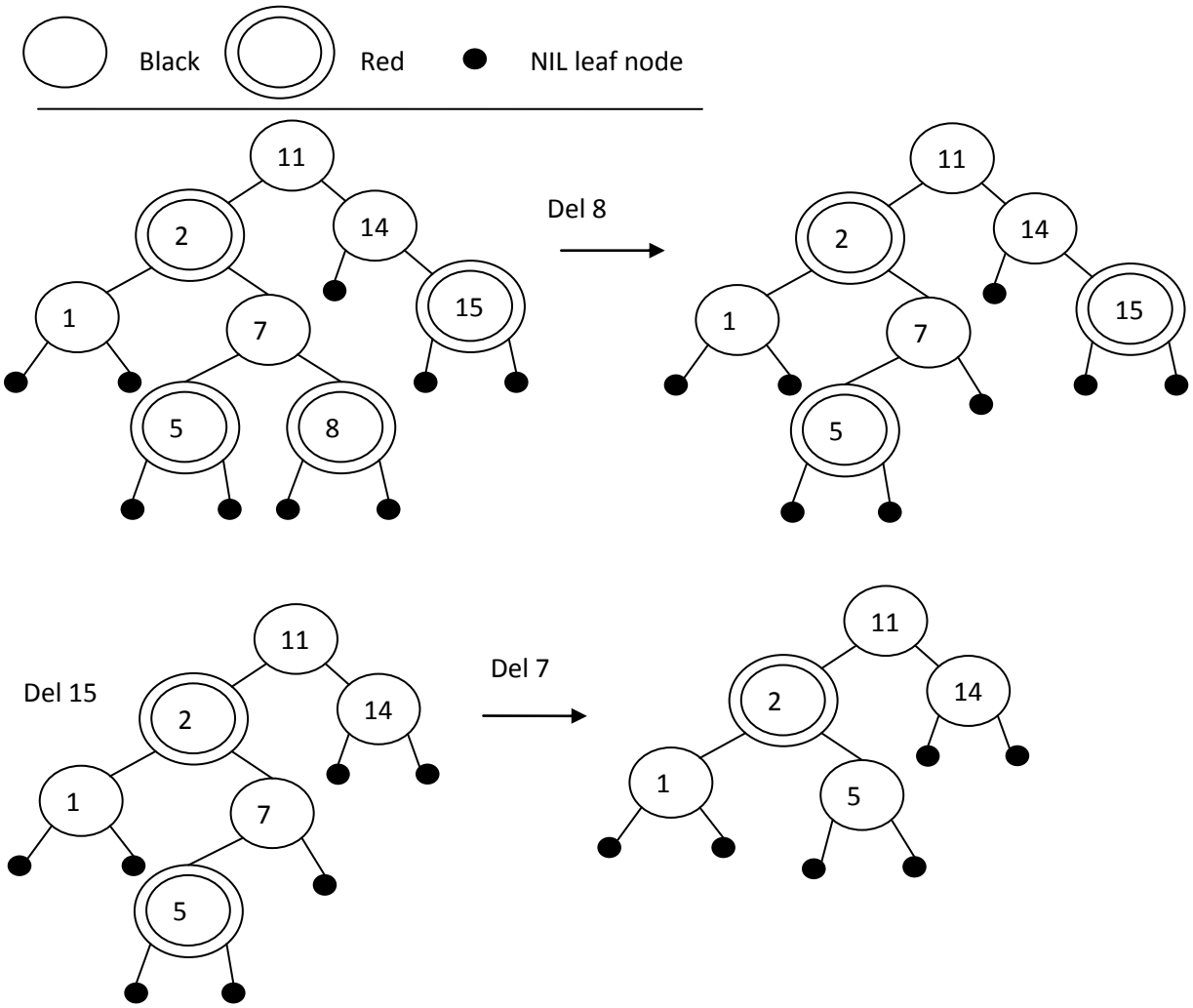


**CSCD320 Homework3,Spring 2014, Eastern Washington University. Cheney, Washington.**

**Name:** Brandon Fowler **EWU ID: 00639348** **Due:** 11:59pm, April 27, 2014 (Sunday)

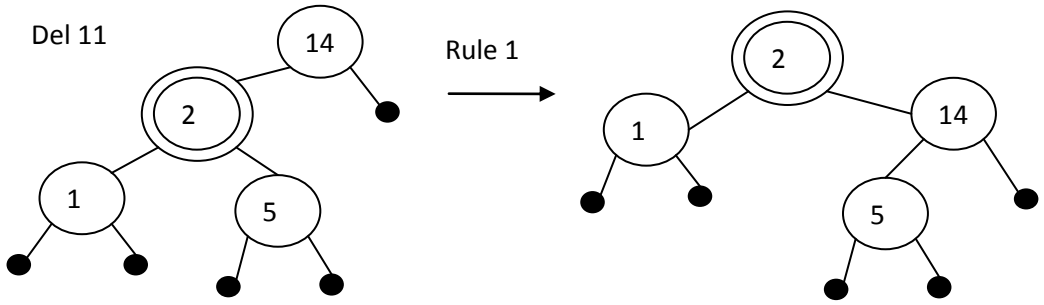


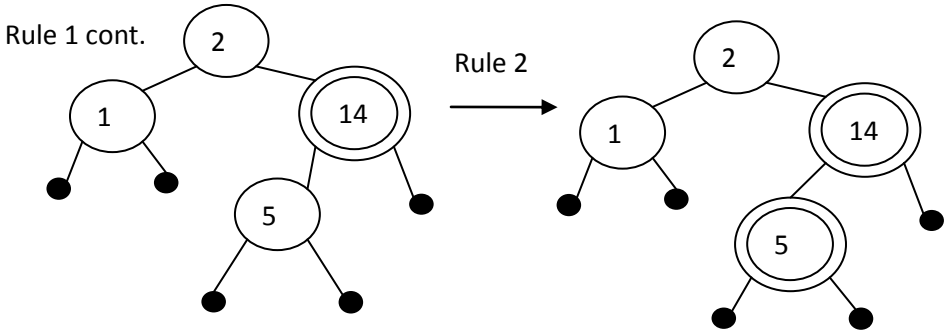

**Problem 3** (25 points)**.** *Show the trace of deleting the nodes from the Red-Black tree below in the order of* 8, 15, 7, 11, 2, 5, 1, 14*. That is, show the state of the tree after deleting each node.*

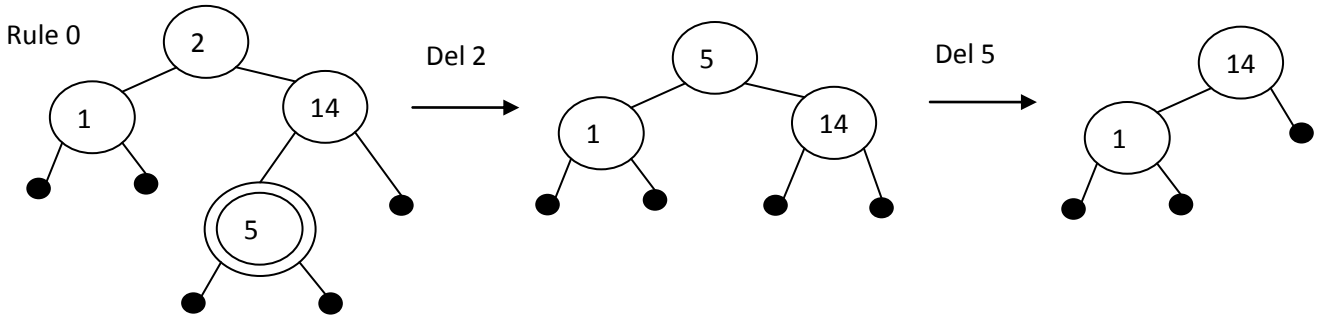


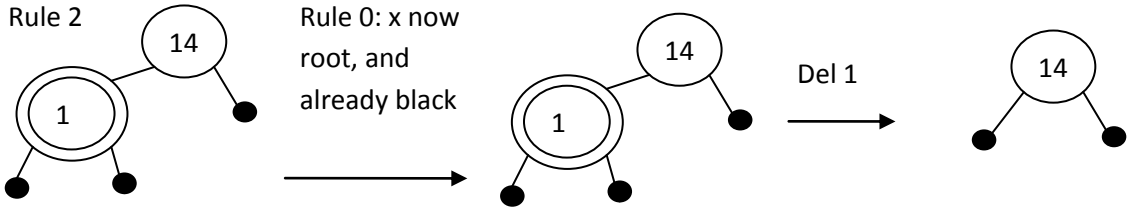
**CSCD320 Homework3,Spring 2014, Eastern Washington University. Cheney, Washington.**

**Name:** Brandon Fowler **EWU ID: 00639348** **Due:** 11:59pm, April 27, 2014 (Sunday)



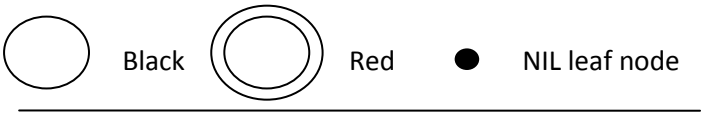

Del 14, then done.

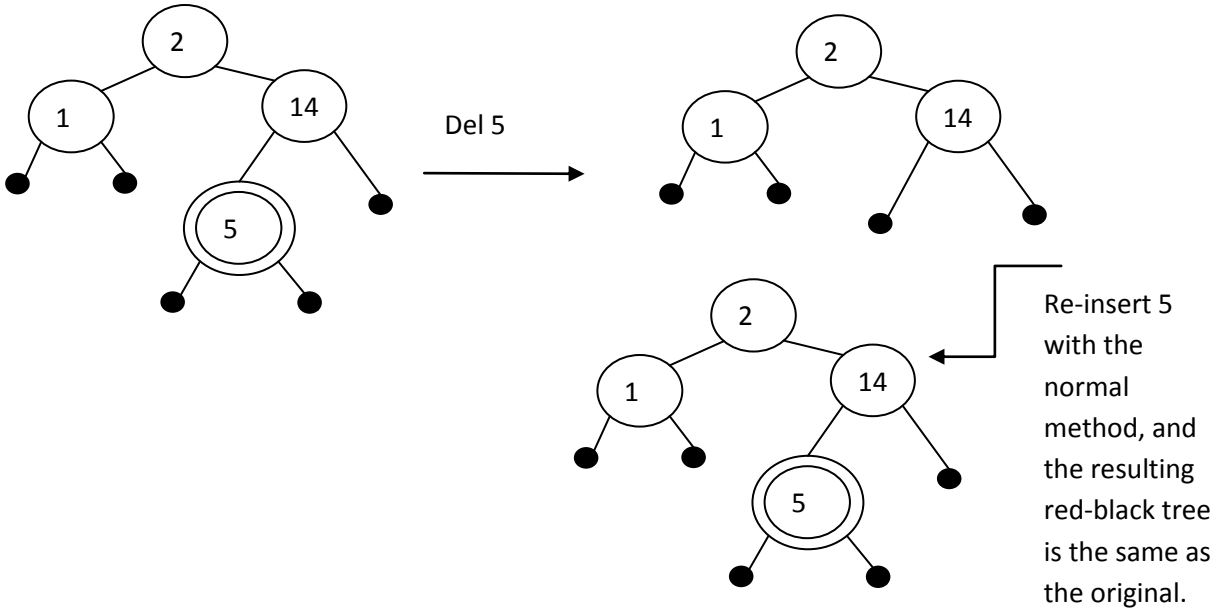
**CSCD320 Homework3,Spring 2014, Eastern Washington University. Cheney, Washington.**

**Name:** Brandon Fowler **EWU ID: 00639348** **Due:** 11:59pm, April 27, 2014 (Sunday)

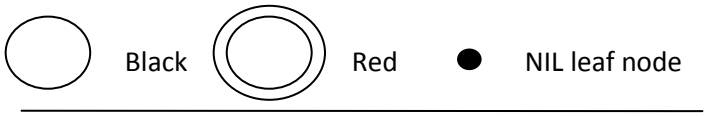
**Problem 4** (25 points)**.** *Suppose that a node* x *is deleted from a red-black tree and then is immediately inserted back. Is the resulting red-black tree the same as the initial red-black tree ? If yes, prove it; If no, give an example to justify your answer.*

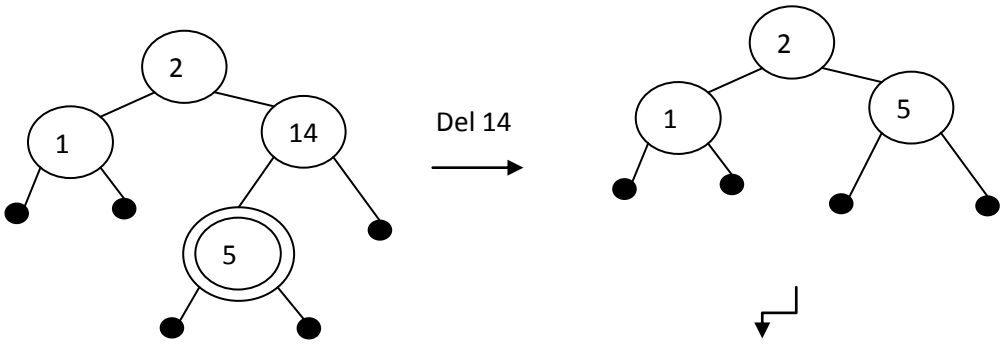
Positive example:




Negative Example:

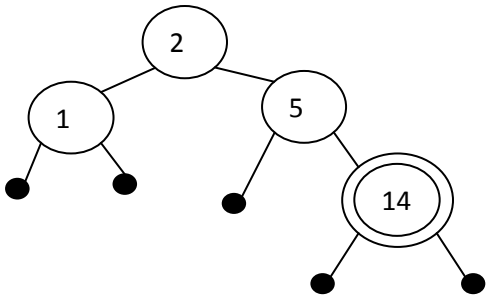


**CSCD320 Homework3,Spring 2014, Eastern Washington University. Cheney, Washington.**

**Name:** Brandon Fowler **EWU ID: 00639348** **Due:** 11:59pm, April 27, 2014 (Sunday)

Re-insert 14 with the normal method, and the resulting red-black tree is not the same as the original.




Shown by the above first example, it is possible when a node(such as an end red node) is deleted, and then it is immediately re-inserted, the resulting red-black tree will be the same as it was before the deletion. However, by the second example, it is possible if a node is deleted, and then it is immediately re-inserted, the resulting red-black tree will be different. Therefore, in regard to the initial question, it is not guaranteed or even likely that when a node x is deleted, and then immediately re-inserted, that the resulting red-black tree will be the same as the initial tree, but it can be possible.