Informatics II, Spring 2024, Solution Exercise 9

Publication of exercise: April 29, 2024
Publication of solution: May 6, 2024
Exercise classes: May 6 - 10, 2024

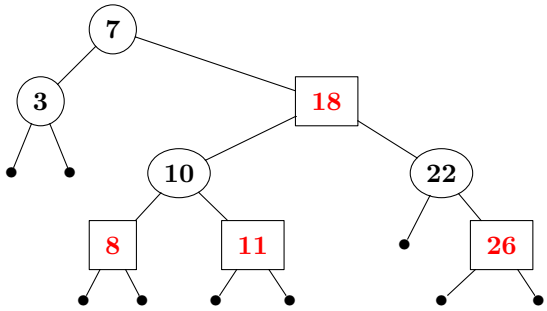
Learning Goal

- Understanding Red-black Trees and Insertion, Deletion rules on it.
- Knowing how to implement LeftRotate, RightRotate on Binary Trees and how it makes trees balanced in Red-black Trees.

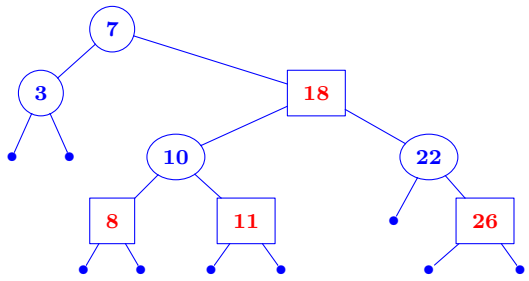
Task 1 [Medium]

Consider the red-black tree shown in following figure. State the operations and stepwise changes in tree when 15 is inserted in the tree (You have to show the state of tree after color change and rotation transformations).

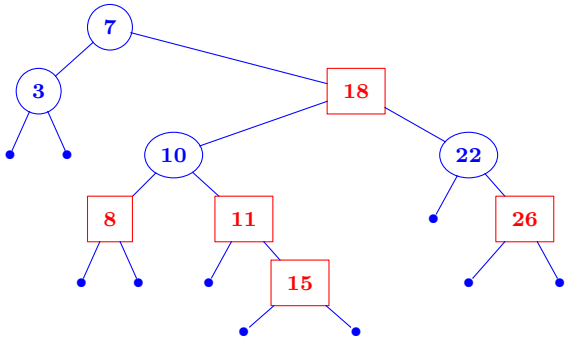
Initial tree



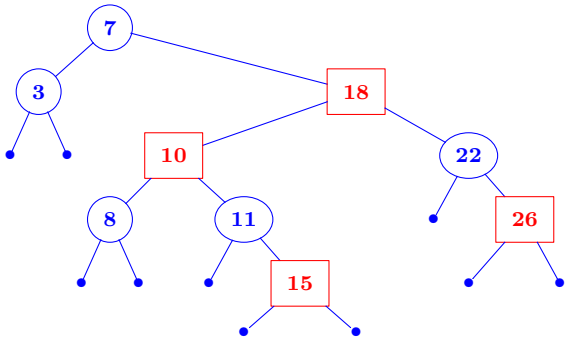
Initial tree



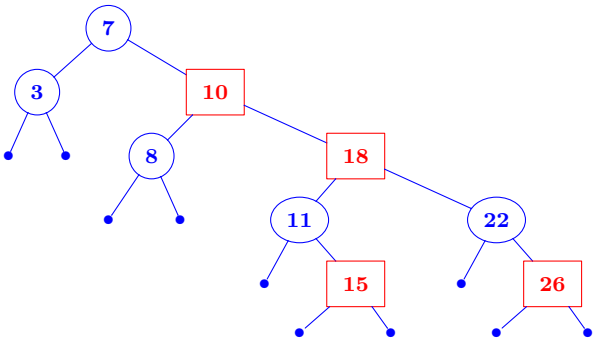
Step:1 Insert red node 15 by a rule of Binary search tree
Operations: `create(15)`, `11->right = 15`



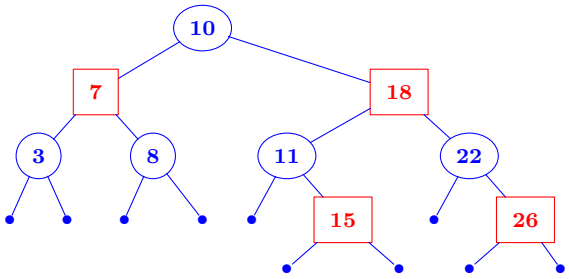
Step 2: Identify the insertion case for t=15:
Condition: 15's uncle 8 is red (Case 1)
Operations: `11->color = 8->color = black, 10->color = red`
set t = 10 and continue



Step 3: Identify the insertion case for t=10:
Condition: 10's uncle 3 is black, 10's parent 18 is a right child and 10 is a left child (Case 2 Mirrored)
Operations: `RightRotate(18)`
set t = 18 and continue



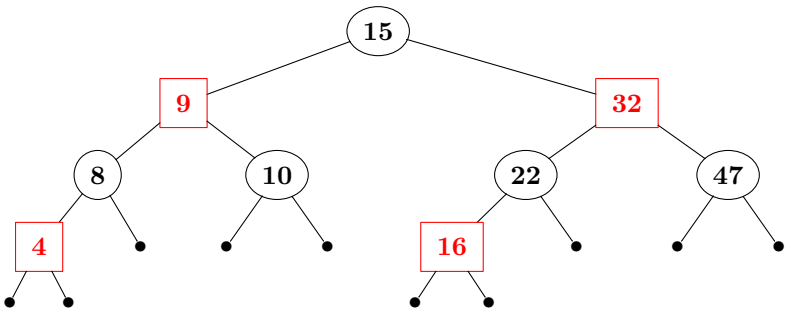
Step 4: Identify the insertion case for t=18:
Condition: 18's uncle 3 is black, 18's parent 10 is a right child and 18 is a right child (Case 3 Mirrored)
Operations: `10->color=black, 7->color=red, LeftRotate(7)`



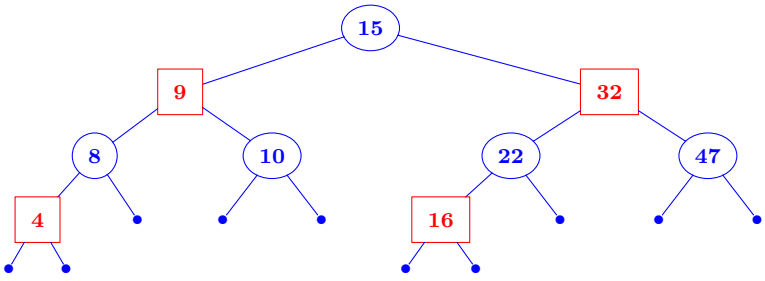
Task 2 [Hard]

Consider the red-black tree shown in Figure below. Perform the following operations on it and draw the resulting tree after each of them: Delete 10, Insert 5, Delete 9, Delete 15.

Initial tree



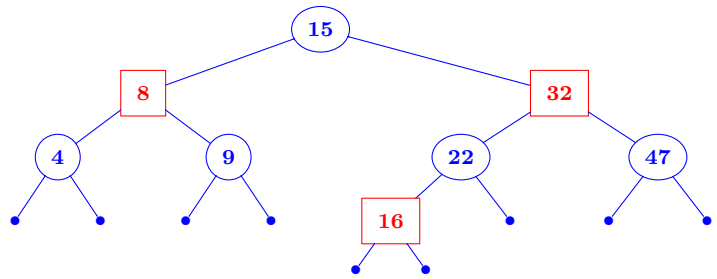
Initial tree



Delete 10

Condition: 10 is a leaf node and right child. 10's brother 8 is black. 8's left child 4 is red whereas its right child is black. (Case 4 mirrored)

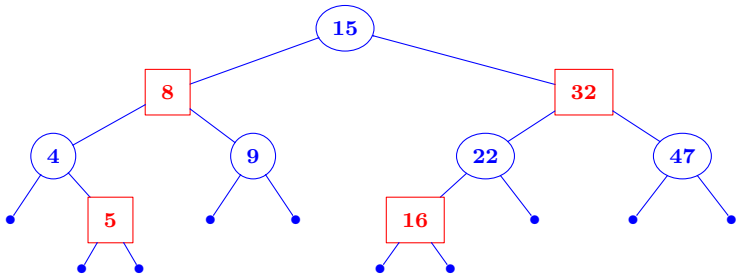
Operations: `Delete(10), 8->color=9->color (red), 9->color=black, 4->color=black, RightRotate(9)`



Insert 5:

Insert red node 5 by a rule of Binary search tree

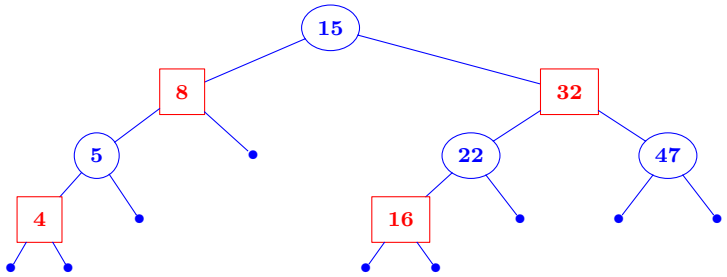
Operations: `create(5), 4->right=5`



Delete 9

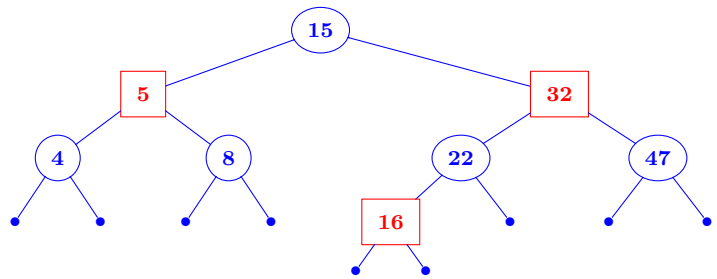
Condition: 9 is a leaf node and right child. 9's brother 4 is black. 4's right child 4 is red whereas its left child is black. (Case 3 mirrored)

Operations: `Delete(9), 5->color=black, 4->color=red, LeftRotate(4)`
setting brother b=5

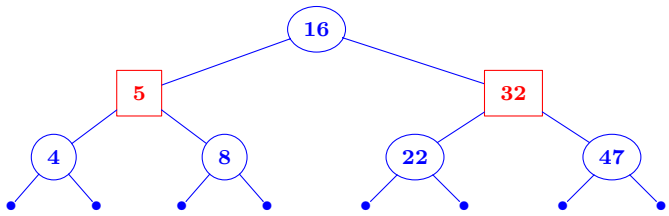


Condition: Deleted node 9's brother 5 is black. 5's left child 4 is red whereas its right child is black. (Case 4 mirrored)

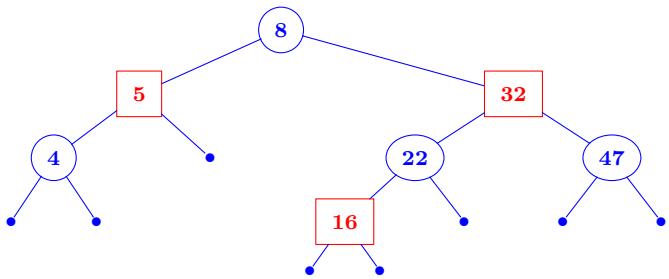
Operations: `5->color=8->color (red), 8->color=black, 4->color=black, RightRotate(8)`



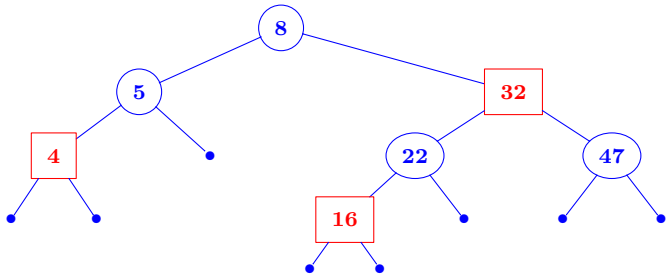
Delete 15:
Solution 1: Replace 15 with the smallest child from right subtree, i.e. 16
Operations: `Delete(16), 15->key = 16`
The color of delete node 16 is red we do not need to do anything else.



Solution 2: Replace 15 with the smallest child from right subtree, i.e. 8
Operations: `Delete(8), 15->key = 8`
The color of deleted node 8 is black, so we continue.



Condition: Deleted node 8's brother 4 is black and both its children are also black. (Case 2)
Operations: `4->color=red, 5->color=black`



Task 3 [Medium]

You are given an implementation of binary search tree in the file `task3_framework.c`. A **binary search tree** is of the following type:

```
1  struct TreeNode {
2    int val;
3    struct TreeNode* left;
4    struct TreeNode* right;
5    struct TreeNode* parent;
6  };
```

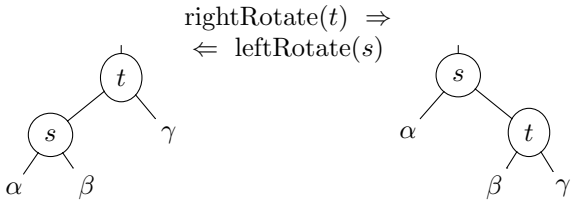
The implementation also includes the following functions:

- *struct TreeNode* insert(struct TreeNode* root, struct TreeNode* parent, int val)* - inserts an integer `val` into the binary search tree.
- *struct TreeNode* search(struct TreeNode* root, int val)* - search and returns a binary search tree node with value *val*. Return *NULL* if value is not found.

Your task is to implement two functions `leftRotate` and `rightRotate` in C:

1. *struct TreeNode* leftRotate(struct TreeNode* root, int val)* - that left rotates the binary search tree node with value *val*.
2. *struct TreeNode* rightRotate(struct TreeNode* root, int val)* - that right rotates the binary search tree node with value *val*.

The left and right rotation operations on nodes s and t respectively have been illustrated in Figure below where α, β and γ represents the subtrees.



Solution: see code task3.c