Hello sir
I have submitted the red black tree again. Could you please check?
For your kind consideration, I have added much comments on the lines of my code.
There are some basic differences between the code I have submitted now and the code I have submitted at the first time.

Sir,
I have wrote down the code by myself. As I have learned the red black tree from the Wikipedia there may be some structural similarity between my code and code written on Wikipedia. But it is not the copy because code in Wikipedia is written in C programming language.

Now I am describing my code:

I have three classes in the package named “redblacktreesol”

1. NodeCreation
2. RBTree
3. Task(contains the main class)

1 class NodeCreation

This class has been created as the basic entity for Red Black tree. Its every object is a node of the tree. Each of the object contains five class fields

1. key
2. rightchild
3. leftchild
4. parent
5. color

This class also contains following methods

1. NodeCreation(constructor)→ assigns children of the new node if any given
2. FindGrandparent()→ returns parent of parent of a given node
3. FindSiblings()→returns another child of parent of a given node
4. FindUncle()→returns another child of grandparent of a given node

2 class RBTree

This class organizes to create the tree. It contains the following class fields

1. ANSI_RESET= "\u001B[0m"→ ANSI reset code to reset the to the default color. It is used in printer method
2. ANSI_RED= "\u001B[31m" → ANSI red color code used in printer method to print using red color

3. Spacing= integer type variable used in printer method for spacing

The class also contains the following methods

1. Insert()→ creates an object of NodeCreation class which is actually a node of the tree and assigns key of the new node, place the new node as like as node placement in Binary search tree. And call the inFact1() method to ensure that the properties of red black tree are preserved.
2. inFact1()→ make initiative to preserve the properties of Red Black tree if it is the insert case 1 of the red black tree insertion. It also calls the inFact2() method if the case of the new node does not fulfill the requirements of insert case 1 .
3. inFact2()→ make initiative to preserve the properties of Red Black tree if it is the insert case 2 of the red black tree insertion. It also calls the inFact3() method if the case of the new node does not fulfill the requirements of insert case 2 .
4. inFact3()→ make initiative to preserve the properties of Red Black tree if it is the insert case 3 of the red black tree insertion. It also calls the inFact4() method if the case of the new node does not fulfill the requirements of insert case 3 .
5. inFact4()→ make initiative to preserve the properties of Red Black tree if it is the insert case 4 of the red black tree insertion. It also calls the inFact5() method if the case of the new node does not fulfill the requirements of insert case 4 .
6. inFact5()→ make initiative to preserve the properties of Red Black tree if it is the insert case 5 of the red black tree insertion.
7. delete()→ deletes a node using its key. It calls the NodeSearch() method to search a node using its key. Then it calls the inOrderPredecessor() method to search the in-order predecessor of the node to whose key is to be deleted. Then it copies the key of the in order predecessor node to the node whose key is to be deleted. After that it moves the pointer to the in order predecessor node and make it current node (its key to be deleted, denoted by P). It also finds the child(N) of the current node. At last it calls delFact1() to ensure that the properties of the Red Black tree are preserved.
8. delFact1()→make initiative to preserve the properties of Red Black tree if it is the delete case 1 of the red black tree node deletion. It also calls the delFact2() method if the case of the node whose key to be deleted does not fulfill the requirements of delete case 1 .
9. delFact2()→make initiative to preserve the properties of Red Black tree if it is the delete case 2 of the red black tree node deletion. It also calls the delFact3() method if the case of the node whose key to be deleted does not fulfill the requirements of delete case 2 .
10. delFact3()→make initiative to preserve the properties of Red Black tree if it is the delete case 3 of the red black tree node deletion. It also calls the delFact4() method if the case of the node whose key to be deleted does not fulfill the requirements of delete case 3 .
11. delFact4()→make initiative to preserve the properties of Red Black tree if it is the delete case 4 of the red black tree node deletion. It also calls the delFact5() method if the case of the node whose key to be deleted does not fulfill the requirements of delete case 4 .
12. delFact5()→make initiative to preserve the properties of Red Black tree if it is the delete case 5 of the red black tree node deletion. It also calls the delFact6()
13. delFact6()→make initiative to preserve the properties of Red Black tree if it is the delete case 1 of the red black tree node deletion.
14. RotatingRight()→ rotates to the right of a given node
15. RotatingLeft()→rotates to the left of a given node
16. NodeReplacing()→ switch the position of two nodes in the tree

17. getNodeColor()→ returns the color of the given node. If node is null it returns the “BLACK” color assuming that the given node is a leaf node
18. inOrderPredecessor()→ It is used to find the in order predecessor of a node. Actually it returns the right most successor of the given argument. If we want to have in order predecessor of a given node we need to pass the left child of the given node as argument of this method.
19. NodeSearch()→ returns the node after searching according to the given key. It uses the standard way binary search tree searching procedure.
20. Printer()→ this method prints the whole red black tree as the last most situation of the tree.

3 class Task(contains the main class)

This class contains the main method. It creates the object of the RBTree class. It reads the key runtime. It calls the insert(), delete() and printer() method of the RBTree class to do respectively insert , delete and print the tree. This class uses a switch-case system to offer the options to the user.

4 extra code: It is another copy of the class Task. But it is designed to print out the RBTree100, RBTree50 and RBTree075. If anybody wants to use this code he needs to delete all code from the class Task and copy the codes from the file extra code. He also needs to paste the address of the input key file in the respective place of the code.