CSE-310 : Data Structures and Algorithms

Homework 6 - Due Tuesday April 5th by 1:30pm

This is an individual programming assignment. The goal is to implement a working version of the red-black tree data structure with the insert, delete and search capabilities assuming that the key values of the nodes are distinct. The input file will contain a *valid RB-Tree* as well as a list of operations that need to be performed on the tree. The output file will be the resulting tree after the operations are performed in the same format. Your program should be compilable on the general.asu.edu server and must include a README file describing how to compile and test your program.

- You can assume there will be enough memory space available to keep the RB-Tree in memory.
- Your program will be tested on the general.asu.edu server so make sure that it compiles and executes correctly on that server.
- Your program must accept the name of the input file as an input parameter.
- The output file should be named "output.txt".
- You *must* include a documentation file named README that includes your name and student ID and explains how to compile and test your program. You can use a "makefile" to simplify the compilation process.

1. Input file format
The Input file will only contain 32 bit-integer values. The first portion of the input file will contain the description of the starting tree while the second portion will contain a list of commands. These two parts are separated by a -1 in the input file.

1.1 Input Tree
The input tree will be given in pre-order form and will be in the format of a color code $(0 : red, 1 : black)$ followed by the key value of that node. You need to implement the Tree-Insert function of the BST and call that function on every node given as input and then color the node using the given color in $O(nlgn)$ (The running time for each insertion will be $O(lgn)$ because we are assuming the input tree is a valid RB-Tree. Alternatively you can try

to implement a faster version with a running time of $O(n)$ however this will not affect your grade.)

1.2 Input Commands
A non-negative number k which is followed by k commands. The command operations will be given in the following coded form:

- 0 x: Add a node with key value x into the RB-Tree. You can assume that the tree currently does not contain a node with this key value. A new node should be created with the given key value and inserted into the tree and then the appropriate fix-up needs to be performed.
- 1 x: Delete the node with key value x from the RB-Tree. You first need to search for the node with key value x inside the tree. If the node is not found then nothing needs to be changed. If the node was found then it should be removed and the appropriate fix-up needs to be performed.

2. Output file format
Same as the first portion of the input file followed by a -1.

3. Submission
Create a zip file named "lastname_firstname_hw3.zip" that consists of your program files and the README file and upload it to the hw3 section in the assignments page before the deadline. Turn in a hard-copy of your README file in class.

4. Examples

4.1 Example 1

Starting from an empty RB-Tree insert 2 and 4, then try to delete 3:
Input file:

```
-1
3
0 2
0 4
1 3
```

Output file:

```
1 2
0 4
-1
```

4.2 Example 2

This is the fifth example of RB-Insert from the class slides:
Input file:

```
1 5
1 3
0 17
1 15
0 13
0 16
1 20
-1
1
0 12
```

Output file:

```
1 15
0 5
1 3
1 13
0 12
0 17
1 16
1 20
-1
```

4.3 Example 3

This is the fifth example of RB-Delete from the class slides:
Input file:

```
1 9
1 5
0 2
1 1
1 3
0 4
1 8
1 15
1 13
1 16
-1
1
1 8
```

Output file:

```
1 9
1 2
1 1
0 4
1 3
1 5
1 15
1 13
1 16
-1
```