CPSC 319

Assignment#3

Alex Yeap

Alex.yeap@ucalgary.ca

Tutorial: T10

TA: Mobin Vahdati

1. Build a binary search tree using the data from the input file. Both insertion into and deletion from the tree will be done. The tree should be ordered by *student last name* (use a case-insensitive comparison). There are only unique records in the input file. Each node must contain the student data (exclude the *operation code*), a left child pointer, and a right child pointer. A parent pointer is optional, but might prove useful for some operations.

```
BinaryTree tree = new BinaryTree();
FileInputStream br = new FileInputStream(inputFile);
Scanner sc = new Scanner(br);
studentNameRecords data = new studentNameRecords();
while (sc.hasNextLine()){
    String line = sc.nextLine();

    if (line.charAt(0) == 'I'){
        data.studentRecords(line);
        tree.insertion(line.substring(8,17).trim());

    }
    else if (line.charAt(0) == 'D'){
        tree.deletion(line.substring(8,17).trim());
    }

}
sc.close();
```

```
[Last Name: McKay
, ID: 853453
, Home Department: 025
, ProgramCT
, Year: 1
,
, Last Name: LaPorte
, ID: 840034
, Home Department: 004
, ProgramJA
, Year: 1
,
, Last Name: Black
, ID: 849912
, Home Department: 034
, ProgramRS
, Year: 1
,
, Last Name: Green
, ID: 840091
, Home Department: 004
, ProgramRF
, Year: 1
,
, Last Name: Johnston
, ID: 842291
, Home Department: 034
, ProgramRS
, Year: 1
,
, Last Name: Schafer
, ID: 821239
, Home Department: 025
, ProgramES
, Year: 1
,
, Last Name: White
, ID: 836709
, Home Department: 034
, ProgramTX
, Year: 1
,
, Last Name: Phillips
, ID: 841209
, Home Department: 025
, ProgramES
, Year: 1
,
, Last Name: Smith
, ID: 840000
, Home Department: 025
, ProgramCT
, Year: 2
,
, Last Name: Sykes
, ID: 825599
, Home Department: 045
, ProgramWE
, Year: 2
,
, Last Name: Walker
, ID: 830000
, Home Department: 034
, ProgramTX
, Year: 1
,
, Last Name: Hall
, ID: 848843
, Home Department: 034
, ProgramRS
, Year: 1
,
, Last Name: Jones
, ID: 830670
, Home Department: 025
, ProgramCT
, Year: 2
,
, Last Name: Bannister
, ID: 830116
, Home Department: 025
, ProgramES
, Year: 1
,
, Last Name: Banks
, ID: 850045
, Home Department: 025
, ProgramES
, Year: 1
,
, Last Name: Woods
, ID: 838823
, Home Department: 025
, ProgramCT
, Year: 1
,
, Last Name: Appleford
, ID: 850388
, Home Department: 025
, ProgramES
, Year: 1
,
, Last Name: Hopper
, ID: 832218
, Home Department: 025
, ProgramCT
, Year: 1
,
, Last Name: Morrison
, ID: 832288
, Home Department: 004
, ProgramJA
, Year: 3
,
, Last Name: Card
, ID: 842299
, Home Department: 004
, ProgramJA
, Year: 2
,
, Last Name: Jordan
, ID: 839912
, Home Department: 033
, ProgramEN
, Year: 1
,
, Last Name: Last
, ID: 830050
, Home Department: 025
, ProgramCT
, Year: 1
,
, Last Name: West
, ID: 850000
, Home Department: 033
, ProgramEN
, Year: 1
,
, Last Name: Weston
, ID: 830123
, Home Department: 033
, ProgramEN
, Year: 2
,
, Last Name: Fisher
, ID: 850888
, Home Department: 034
, ProgramTX
, Year: 1
,
, Last Name: Watson
, ID: 840000
, Home Department: 033
, ProgramEN
, Year: 2
,
, Last Name: Zot
, ID: 859999
, Home Department: 045
, ProgramWE
, Year: 1
,
, Last Name: Waters
, ID: 849934
, Home Department: 034
, ProgramTX
, Year: 1
,
, Last Name: Watts
, ID: 850006
, Home Department: 034
, ProgramRS
, Year: 1
,
, Last Name: Newman
, ID: 832218
, Home Department: 025
, ProgramCT
, Year: 2
,
, Last Name: Doyle
, ID: 839900
, Home Department: 033
, ProgramEN
, Year: 1
,
]
```

2. Traverse the binary search tree recursively, printing out the nodes in ascending logical order; i.e. do a *depth-first, in-order* tree traversal. Print the node data to a text file.
3. Traverse the binary search tree, starting at the top level (the root node), proceeding downwards level-by-level. At each level print out the nodes from left to right. In other words, do a *breadth-first* traversal. You may have to use a queue to implement this. Print the node data to a text file.

a3input1 in-order and Breadth-first

```
C:\Users\AlexY\OneDrive - University of Calgary\CPSC 319\CPSC Assignment3>java Assignment3 a3input1 output1 output2


Appleford >Banks >Bannister >Black >Card >Doyle >Fisher >Green >Hall >Hopper >Johnston >Jones >Jordan >LaPorte >Las
t >McKay >Morrison >Newman >Phillips >Schafer >Smith >Sykes >Walker >Waters >Watson >Watts >West >Weston >White >Wo
ods >Zot >

McKay > LaPorte > Schafer > Black > Last > Phillips > White > Bannister > Green > Morrison > Smith > Woods > Banks
> Card > Johnston > Newman > Sykes > Zot > Appleford > Fisher > Hall > Jones > Walker > Doyle > Hopper > Jordan > W
est > Watson > Weston > Waters > Watts >
```

Output 1 file: can't get it to work

Output 2 file:

output2.txt

```
1    McKay > LaPorte > Schafer > Black > Last > Phillips > White > Bannister > Green > Morris
```

a3input2 in-order and Breadth-first

```
C:\Users\AlexY\OneDrive - University of Calgary\CPSC 319\CPSC Assignment3>java Assignme
nt3 a3input2 output1 output2


Appleford >Banks >Bannister >Black >Card >Doyle >Fisher >Green >Hopper >Johnston >Jones
 >Jordan >LaPorte >Last >Morrison >Newman >Phillips >Smith >Sykes >Walker >Waters >Wats
on >Watts >West >White >Woods >

Morrison > LaPorte > Smith > Black > Last > Phillips > White > Bannister > Green > Newm
an > Sykes > Woods > Banks > Card > Johnston > Walker > Appleford > Fisher > Hopper > J
ones > West > Doyle > Jordan > Watson > Waters > Watts >
```

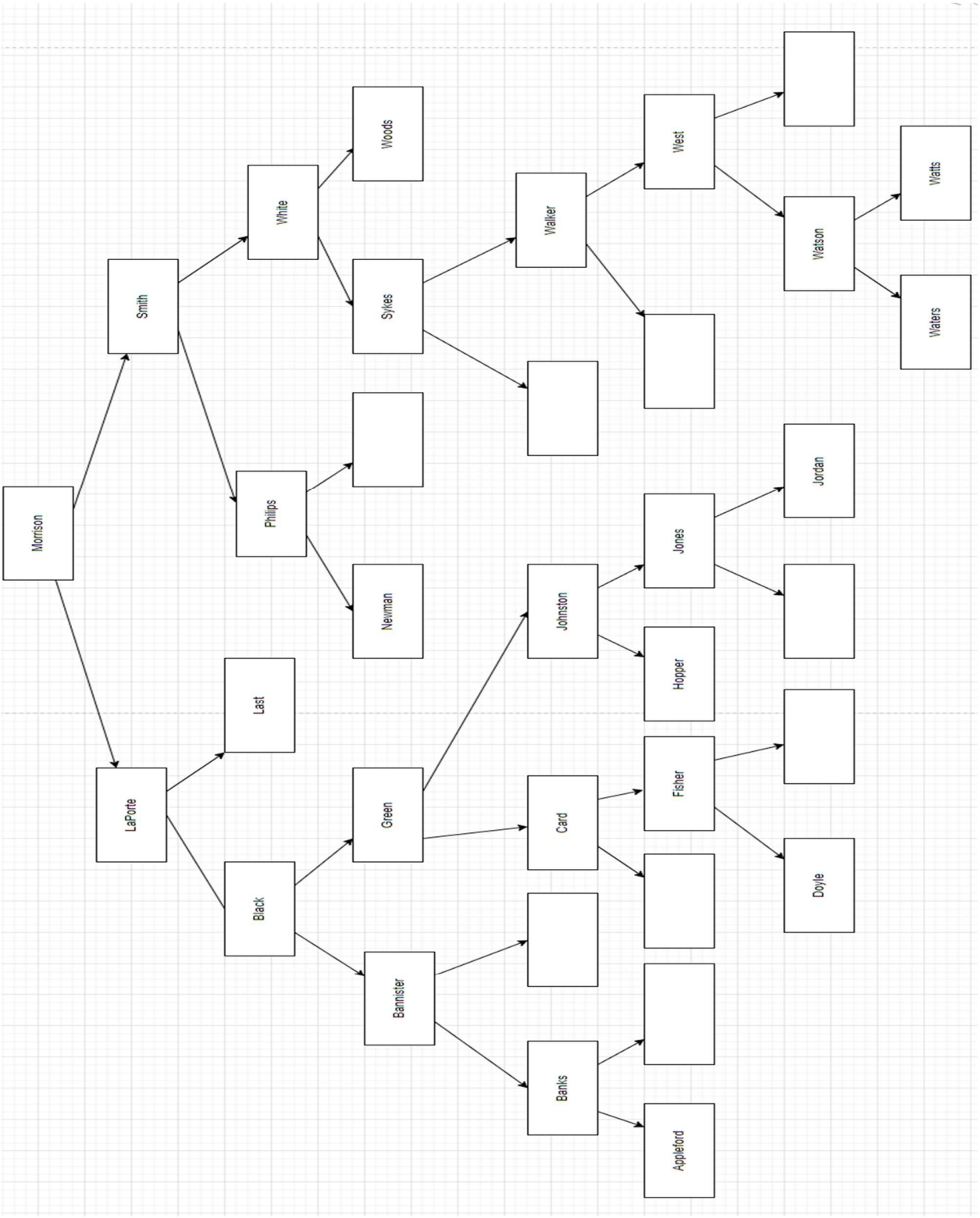
Output1 file: can't get it to work

Output 2 file:

output2.txt

```
1    Morrison > LaPorte > Smith > Black > Last > Phillips > White > Bannister > Gre
```

picture of the binary search tree, as it appears after processing all the insertions and deletions specified in the input file. Within each node, only show the data that is used to order the tree. The picture can be hand drawn, or you may use a graphics application to create it.



Complexity Analysis

Let n be the total number of records stored in the data structure.

1. Assuming that the records are inserted into the tree in random order, what is the height of your tree expressed using big-O notation?

If the records are inserted in random order as they are not balanced the tree we would have a total of 30 nodes + 1 root node which would represent n as n is the number of nodes which is also the number of records therefore the big-O notation would be O(n) = O(30)

2. What is the worst-case height of the tree? What input gives the worst case?

The worst case for the height of a Binary search tree will be the time complexity O(n) as n is the height of the tree. The height of the tree would be the distance from the root to the furthest node which is also known as the largest number of edges. The worst-case input would be a long narrow tree which is also known as a skew Binary search tree.

3. What is the worst-case space complexity of the depth-first, in-order traversal and breadth-first traversal? Compare your implementation of these two methods: is there one that will outperform another in terms of memory usage for a specific data set? Discuss.

The worst-case space complexity of a depth-first binary and in-order traversal will be O(n) where n is the longest path length which is also known as the maximum height/depth of the tree

The worst-case space complexity of a breadth-first traversal will be O(|v|) as v represents the number of nodes since we need to traverse all nodes. The worst case would also be holding all the vertices in the queue

In terms of memory usage, it will depend as the worst case for a depth-first binary is the best case for the Breadth first traversal and vice versa. Since they are opposite to another. It will use more memory usage on the worst-case scenario when traversing through the tree.

The result for the bonus question

Pre-Order