Cairo University
Faculty of Computers & Artificial Intelligent

CS214 – Data Structures – 2020
Assignment (3)

[130 Points]

Delivered Files Name:

- Assignment number with teams member IDs in zip file name A2_ID1_ID2_ID3_ID4.zip
- **Don't** create folder inside the zip put code files directly
- **Don't** create zip files inside the main zip file.
- Only include the .h files, .cpp files, .doc files and .txt files. **Don't** include any other project files or exe files
- Problem code files name should be section number with problem number S#_P#.cpp or .doc if it wasn't code problem or just section number for first section problems
- **Any submission that doesn't follow the above rules will be ignored**
- **Example**
    - A2_2018203_2018204_2018205.zip
    - Inside zip file
        - BSTFCI.cpp (it will be used in section 1 problems)
        - S1.cpp
        - S2_P1.cpp
        - S2_P2.cpp
        - S3_P1.cpp
        - S3_P2_P3.doc

Section #1 (80 points):

Create template binary search tree class with this name BSTFCI and create node class with name BSTNode (10 points)

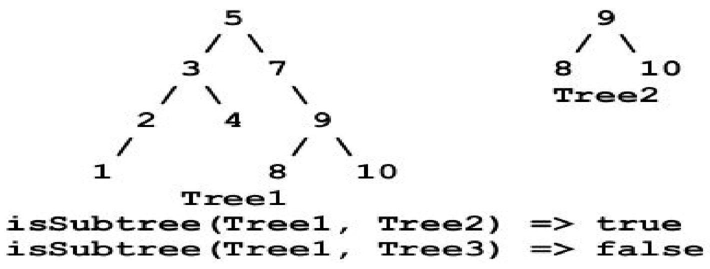
1. Add Checking Tree Balance (10 points):

- A Balanced Binary Tree is a tree where the heights of the two child sub-trees of any node differ by at most one and the left subtree is balanaced and the right subtree is balanced.
- Add method called **'isBalance'** to BSTFCI this method will check in the BST is balance or not.

2. Tree Comparison (10 points):

-Write a function that decides if a BSTFCI T2 is a sub-tree of another BSTFCI T1. **Prototype: bool isSubTree(BSTFCI* t1, BSTFCI* t2);**

Note: You may need to write another function that takes 2 BSTNodes and compares their sub-trees.



```
            5                           9                  3
           / \                         / \                / \
          3   7                       8   10             2   4
         / \   \                      Tree2             /
        2   4   9                                      1
       /       / \                                       Tree3
      1       8   10
          Tree1
isSubtree(Tree1, Tree2) => true
isSubtree(Tree1, Tree3) => false
```

3. Print Range (10 points):

Add a recursive function named **printRange** in the class BSTFCI that stores integers and given a low key value and a high key value, it prints in sorted order all records whose key values fall between the two given keys. Function **printRange** should visit as few nodes in the BST as possible.

You should NOT traverse all the tree in-order and print the ones in the range. This will not be considered a correct solution. You should do smart traversal to only traverse the related parts.

```
           5
          / \
        3    7
       / \    \
     2    4    9
    /         / \
   1         8   10

printRange(3, 6)    => [3,4,5]
printRange(8, 15)   => [8,9,10]
printRange(6,6)     => []
```

4. Tree Flipping (10 points):

Write a **flip** method that takes the node which the mirror image of the tree will start from if no parameter is sent to the function the default value will be the root node.
void flip(Node* node = root)
Output: print the flipped tree

```
Example original tree:           Example new tree:
         1                                1
        / \                              / \
       2   3                            3   2
      / \                                  / \
     4   5                                5   4
```

5. Largest Values (10 points):

Find the largest value in each row of this tree.

```
      1
    /   \
  4       7
         / \
        5   3
```

The output should be: [1, 7, 5]
In the first row, there is only one vertex, the root with value 1;
In the second row, there are two vertices with values 4 and 7, so the largest value is 7;
In the third row, there are two vertices with values 5 and 3, so the largest value is 3.

[input] tree values type integer

A tree of integers. It's guaranteed that the sum of encoded integers won't exceed 2^52.
Guaranteed constraints: 0 ≤ tree size ≤ 5, -1000 ≤ node value ≤ 1000.

[output] array, an array of the largest values in each row of the tree.
Function Signature is => int* largestValues(BSTFCI<int> t)

6. Branches Sum (10 points):

Sum encoded values in each path from root to leaf.

```
      1
    /   \
  0      4
        / \
       3   5
```

The output should be: <u>298</u> (10 + 143 + 145)
Path 1 → 0 encodes 10
Path 1 → 4 → 3 encodes 143
Path 1 → 4 → 5 encodes 145

[input] tree values type integer

A tree of integers. It's guaranteed that the sum of encoded integers won't exceed 2^52.
Guaranteed constraints: 1 ≤ tree size ≤ 2000, 1 ≤ tree depth ≤ 9, 0 ≤ node value ≤ 9.

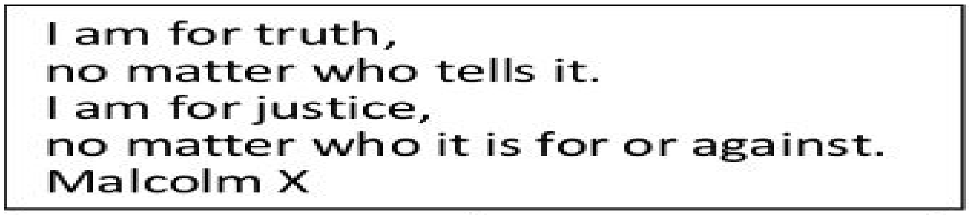
[output] integer64, the sum of the integers encoded in t, as described above.
Function Signature is => long digitSum(BSTFCI<int> t)

7. Tree Application (10 points):

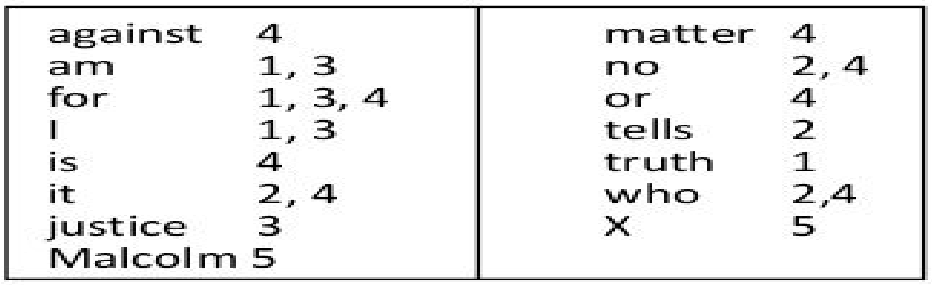
- Write an index builder application that takes text consisting of lines and prints a list of the words of the text and the lines they appear on are printed next to them.
- The application works by building a binary search tree using BSTFCI and each node contains a word and a vector of that contains the list of lines where this word exists. For

each new word, the program finds it and adds the line number to the vector. If word is not found, it is added to the tree. Then traverse the tree in-order to print the nodes.

- You need to remove punctuation marks like **.** and **,** from the text before processing it.
- For example, the text below produces the given index, Test Your code on the given text and 1 more examples.



I am for truth,
no matter who tells it.
I am for justice,
no matter who it is for or against.
Malcolm X



against	4	matter	4
am	1, 3	no	2, 4
for	1, 3, 4	or	4
I	1, 3	tells	2
is	4	truth	1
it	2, 4	who	2,4
justice	3	X	5
Malcolm	5		

Section #2 (25 points):

1. Expression Tree Evaluation (10 points)

● Write an algorithm that accepts an arithmetic expression written in prefix notation and builds an expression tree for it. And then traverse to evaluate the expression. The evaluation should start after the whole expression has been entered.

● Implement your algorithm as a function.

● Write five test cases for your application and run them.

```
expression: + 3 * 4 / 8 2
tree:
         +
        / \
       3   *
          / \
         4   /
            / \
           8   2
Evaluation: 19
```

2. Tree Traversal (15 points)

Assume a binary tree of non-repeated characters. Write a function printPostOrder that takes two strings representing the preorder traversal and the in-order traversal of the tree. Then the function prints the post-order traversal. The function prototype is:

void printPostOrder (string preorder, string inorder);

- Write a program that implements five test cases to test your function.
- A sample function call and the corresponding output is shown below:

printPostOrder ("ABFGC", "FBGAC") => FGBCA
// Tree is

```
      A
     / \
    B   C
   / \
  F   G
```

<u>**Section #3 (35 points):**</u>

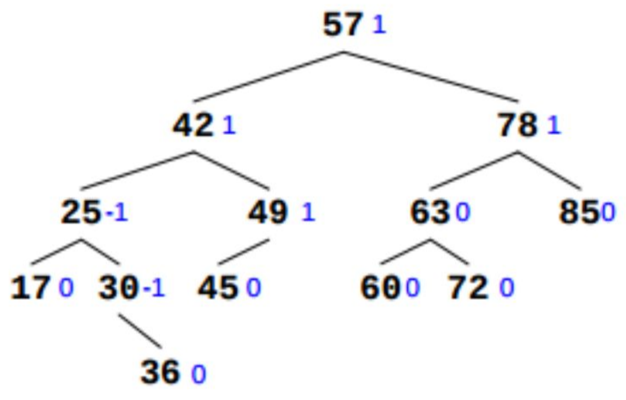
Problem#1 Implement BST AVL tree (25 points)

1. Insert Element into the tree (6 points)
2. Delete Element from tree (14 points)
3. Display Balanced AVL Tree: (5 points)
    a. InOrder traversal
    b. PreOrder traversal
    c. PostOrder traversal

Problem#2 (2 points)

Explain on the **BST AVL** what will happen when you add two elements with value **41, 50**
Note: for solve this problem you should view what happen after adding first element then after adding second element and show the changes in balance factors

Balance factors shown next to each node in blue

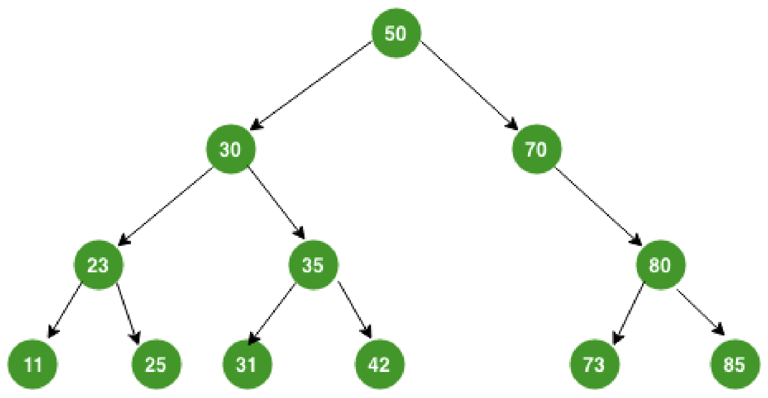


Problem#3 (8 points)

Apply the following operations on the **BST**
Note: The questions are independent on each other for solving any question you should do two steps

Step1: redraw the tree after applying the operation (50%)
Step2: explanation for your step (50%)

1. Delete the node with value 42 (2 points)
2. Delete the node with value 70 (3 points)
3. Delete the node with value 30 (3 points)


50
30
70
23
35
80
11
25
31
42
73
85