CSL 201 Data Structures
Lab 3 Due on September 10, 11.55pm

Instructions

- You are to use Java as the programming language. Use Eclipse as the IDE.
- You have to work individually for this lab.
- You are not allowed to share code with your classmates nor allowed to use code from the internet. You are encouraged engage in high level discussions with your classmates; however ensure to include their names in the report. If you refer to any source on the Internet, include the corresponding citation in the report. If we find that you have copied code from your classmate or from the Internet, you will get a straight F grade in the course.
- The submission must be a zip file with the following naming convention - rollnumber.zip. The Java files should be contained in a folder named after the question number.
- Include appropriate comments to document the code. Include a `read me` file containing the instructions on for executing the code. The code should run on institute linux machines.
- Upload your submission to moodle by the due date and time. Do not email the submission to the instructor or the TA.

This lab will improve your understanding of tree structures, and traversals. This is a lengthy lab, so get started early to complete it on time.

1 Binary Tree Re-Construction (25 points)

Given two traversals of a binary tree, your program must reconstruct the unique binary tree that results in these traversals. If a unique tree cannot be constructed, your program must output that the binary tree cannot be constructed. We have discussed extensively how the unique trees can be constructed using traversals in class. Refer to the lecture slides for understanding the approaches. Every node is displayed in the terminal as a tuple (node type, node value). The values of node type are `root`, `left` and `right`. The contents of the node (in this case strings) corresponds to node value. The nodes are displayed left justified, one node per line. The tab space between the left margin and the tuple is proportional to the depth of the node. So the left and right child of the root node will be each one tab space away from the margin. Their children will be 2 tab spaces away from the left margin and so on. Nodes at the same depth of the tree will have same amount of tab space from the left margin. The preorder traversal of the tree should be used while displaying the tree.
For example consider the tree in Figure 1. This will be displayed at the console in the following format

```
(root, a)
    (left, b)
        (left, d)
        (right, e)
            (left, h)
            (right, i)
    (right, c)
        (left, f)
        (right, g)
```

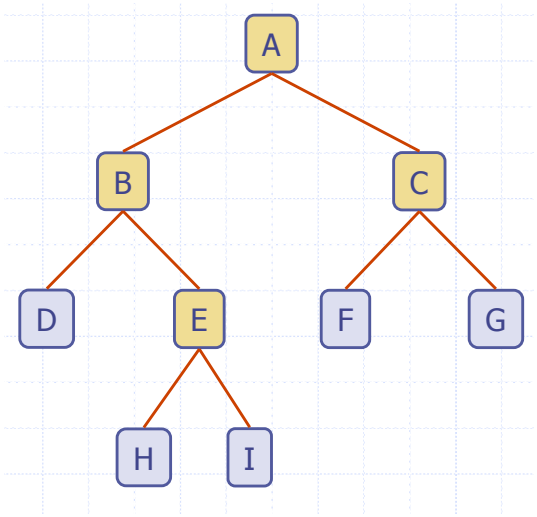

Figure 1: Sample Binary Tree for Problem 1

The input to the program will be defined as `traversal format number of nodes node sequence`. We will considering preorder, ignored and postorder traversals. Thus `traversal format` can be `preorder`, `inorder`, or `postorder`. This is followed by the number of nodes in the tree and finally the sequence of nodes visited according the specified traversal. For example

Input

preorder 9 a b d e h i c f g

inorder 9 d b h e i f c g

Output

```
(root, a)
    (left, b)
        (left, d)
        (right, e)
            (left, h)
            (right, i)
    (right, c)
        (left, f)
        (right, g)
```

Another sample input where the tree cannot be constructed

Input

preorder 9 a b d e h i c f g

inorder 9 h d b e i a f c g

Output

error: Tree cannot be constructed using these two traversals

Please stick to this output format. We will be automatically checking if the output of your program matches with the correct output.

2 Documentation of Code (5 points)

Documentation of code will fetch 5 points.