(a) Before starting the lab, go to GitHub by using your account and create your private repositories called CS102_lab07. Whenever you are done with one of the following parts of the lab you will need to add all of your changes, commit them and push them to the remote repository. Make sure that your repositories are indeed private, because if they are not then anyone on the web will be able to copy your homework, which will get you into trouble. For every part of the lab you should have at least one commit that has a clear message what was implemented in that commit and for which part.

(b) Implement a *Node* class that you will use in the following parts of the lab that will contain attributes: *data* and *next*. For this lab you can assume that data will be of type String. Next will obviously be a node as well.

(c) Implement the *SimpleLinkedList* class as specified below. Write a test class that demonstrates its use, e.g. create a list, add a number of strings to it, print the entire list, get items at various index locations within the list, and remove items from it one-by-one, each time printing it out again. *Note: get(index)* should return *null* if no such *index* exists. Don't forget that your linked list will consist of the nodes you implemented in part a.

public class SimpleLinkedList

public SimpleLinkedList()
public void addToTail(String data)
public String removeFromHead()
public boolean isEmpty()

public Node get(String data)

public String toString()

(d) Using the *SimpleLinkedList* class, create a *SimpleQueue* class, which has *only* the standard *enqueue(String data)* that add data into queue *dequeue()* that remove data from queue and *isEmpty()* methods, and a constructor to create an empty queue.

(e) Now that you have implemented Queue we can use it to make a stack by using 2 Queues. Create *SimpleStackWithQueue* class having only *push(String data) , pop() , toString()* methods.

Hint. For example you can use following method

```
push(x) // x is the element to be pushed
 1) Enqueue x to q2
 2) One by one dequeue everything from q1 and enqueue to q2.
 3) Swap the names of q1 and q2
    // Swapping of names is done to avoid one more movement of all elements
    // from q2 to q1.

pop(s)
 1) Dequeue an item from q1 and return it.
```

Optional Extras...

(a) Try adding more functionality to the SimpleLinkedList class, for example, insert/delete anywhere, append to the end of the list, deleteAll elements in the list, compare with another list, etc.

(b) Try to use the *SimpleStackWithQueue* to make stack structure and demonstrate your stack by using it to evaluate simple postfix expressions. The user should be able to enter the expression as a *String* in which each character is either an operator (+ or -) or an operand (a value between 0 and 9.) A postfix expression has the form "firstNumber secondNumber operator", e.g. "48+" is equivalent to the more conventional infix expression 4 + 8, while "85-" is 8 - 5, and "48+5-" is 4 + 8 - 5. To evaluate such an expression, create an empty stack, then for each token (character) in the expression, if it is a number push it on the stack, else pop the top two numbers from the stack, apply the operator to them and push the result back onto the stack. After all tokens have been processed, pop the result from the stack.

(c) Implement a binary search tree class. Include preOrder, inOrder, postOrder traversals, as well as methods to perform insert and search.

(d) Check out the Java Collections Framework tutorial on Oracle's website.
