# Lab 5: ADT

## 1. Introduction

Last week we talked about some of Abstract Data Types (ADT) and discussed their implementations. In this lab, you will gain experience implementing and using an ADT. More specifically, you will implement a Stack using a Linked List and use it to check whether parentheses in the given string are balanced. Note that you will use your stack implementation in future lab(s), too.

The code skeleton is under: **/comp/15/files/l5**

1. Log in to the CS homework server.
2. Move to your **comp15** directory that you created in Lab 1.
3. Create a directory named **lab5** under the comp15 directory.
4. Move to the lab directory.
5. Copy the code skeleton to the current working directory.
6. Leverage the features of Git to manage your progress toward writing the program.

The code skeleton provides you with seven files: **test.cpp**, **Node.hpp**, **Node.cpp**, **LinkedList.hpp**, **LinkedList.cpp**, **Stack.hpp** and **Stack.cpp**. First, take a look at the files to find the relationships among the **Stack**, **LinkedList** and **Node** classes. The Node class has already been implemented for you, but if necessary, you can implement the default constructor, copy constructor, assignment operator and destructor. Note that the LinkedList class supports only selected operations, and the copy constructor and assignment operator are commented out so that this lab can be completed within our lab time, but you are encouraged to work on them later on. (Same for the Stack class). **test.cpp** provides you with the skeleton of the **isBalanced()** function and several tests with assert() that your implementation is supposed to pass.

## 2. Requirements

1. Implement the **LinkedList** class.
    a. The LinkedList class should be Singly Linked List with head.
    b. The at(int ith) method: "ith" is 0-based. You will define what should happen when the given "ith" is not appropriate.
    c. If you like, you can use your code from the lab2 and lab4.
2. Implement the **Stack** class.
    a. The Stack class should be singly-linked-list based stack.

   b. The push(), pop() and top() methods: Each of them should be implemented as a constant time operation.
   c. The pop() and top() methods: You will define what should happen when *this* stack does not have any items.
3. Implement the **isBalanced()** function in **test.cpp**
   a. The function examines whether the parentheses in the given string are balanced. In order for us to focus on usages of a stack, if the given string includes any characters other than '(' or ')' , then this function returns false.
   b. You may find the following two methods from the string class useful.
      i. size(): returns the size of *this* string.
      ii. at(position): returns the character at the given position in *this* string.
4. Pass the tests with assert() written in **test.cpp** with no memory leaks and no memory errors.

## 3. README

Create the README file that includes the following categories with appropriate section headers.

1. **Name**: Your name.
2. **Date**: The last updated date.
3. **Summary**: A brief summary of the lab.
4. **Files**: A list of files that are necessary to build and test the program.
5. **Instructions**: A sequence of commands to compile and test the program. Note that you are expected to report procedures without using the make command.
6. **References**: A list of citations to information used to complete the lab.

## 4. Submission

Submit your files listed below using Gradescope.
Files: **test.cpp Stack.cpp Stack.hpp LinkedList.cpp LinkedList.hpp Node.cpp Node.hpp README**