# 44-542 Lab Activities

**Lab Activity: Lab09StacksDequesAndLinkedList**

This lab contains **two** exercises which covers stacks, deques, and linkedlist.

**Note:** The name of the project will be your “*Lastname*\_ Lab09StacksDequesAndLinkedList”.

For example: “Smith\_ Lab09StacksDequesAndLinkedList”.

# Exercise 01: StacksDeques

All the classes related to **StacksDeques** must be placed in a package named “stacksanddeques”.

1. Classes: This package has one generic class named AStack<E>

Private instance variable

private ArrayDeque<E> myStack;

Constructors and methods:

* One no-arg constructor: initializes myStack to a new empty ArrayDeque of type E
* push(E element) – adds an element to the top of the stack; no return value
* pop() – removes and returns the top element
* peek() – returns, but does not remove, the top element
* size() – returns the current size of the stack
* isEmpty() – returns true if the stack is empty; false otherwise

1. Classes: This package has one main class named BalancedParens

* Create a new AStack containing objects of type Character, named parenStack.
* Read input from expressions.txt, which contains arithmetical expressions – one per line. A sample data file with corresponding sample output is included.
* For each expression that you read, determine whether the parentheses are correctly balanced.
  + Parse the expression, character by character.
  + If you encounter a left paren, push it onto the stack.
  + If you encounter a right paren, pop the stack. If the parentheses are balanced correctly, there will always be something to pop.
  + When the expressions has been completely parsed, the stack should be empty.
  + For each expression, write VALID if the parentheses are correctly balanced; otherwise, write INVALID, followed by an appropriate error message describing the specific problem, as shown in the sample output.

Sample Data for **expressions.txt**

**(a + b)**

**a + (((b -c) -3) \* (4 \* a) / 3)**

**(a + b**

**a + b)**

**(x + y / (c - d) - 3 + 14 \* (a + c) \* d)**

**(x + y / (c - d - 3 + 14 \* (a + c) \* d)**

**(((( x + y - z**

**(x + y / c - d - 3)) + 14 \* (a + c) \* d)**

**a + b - c + d**

**)a -b(**

Sample Output:

**(a + b): VALID**

**a + (((b -c) -3) \* (4 \* a) / 3): VALID**

**(a + b: INVALID:**

**Parsing complete, but the stack is not empty!**

**a + b): INVALID:**

**Trying to pop, but the stack is empty!**

**(x + y / (c - d) - 3 + 14 \* (a + c) \* d): VALID**

**(x + y / (c - d - 3 + 14 \* (a + c) \* d): INVALID:**

**Parsing complete, but the stack is not empty!**

**(((( x + y - z: INVALID:**

**Parsing complete, but the stack is not empty!**

**(x + y / c - d - 3)) + 14 \* (a + c) \* d): INVALID:**

**Trying to pop, but the stack is empty!**

**a + b - c + d: VALID**

**)a -b(: INVALID:**

**Trying to pop, but the stack is empty!**

**Exercise 02: Lab Activitiy: LinkedList Implementation**

All the classes related to **LinkedList Implementation** must be placed in a package named “**linkedlists**”.

1. Unzip the “LinkedList.zip” to work on Exercise 02.
2. Create a linked list class. To distinguish it from the built-in **LinkedList** class, we will name this class *Lastname***\_ALinkedList**.**java**

For example: Smith\_**ALinkedList.java**

1. Place the given **Node.java** in this package and will need to start the linked list implementation from this class.
2. Copy the stubs of **ALinkedList.java** to “*Lastname\_***ALinkedList.java”** and add the working code.
3. By end of this you must complete the linked list implementation.
4. Place the courses.txt outside “src” folder of the project.
5. Unzip the **testerclasses.zip** which contains two main classes “**Tester.java**” and “**CoursesTester.java**”.
6. Run the “**Tester.java**” and check the respective sample output.
7. Place the “**Course.java**” file in the respective package and test the “**CoursesTester.java**”.

**Sample Output for Tester.java:**

Contents of linked list

55

25

17

Deleting 55

Deleting 25

Deleting 17

Contents of linked list

list is empty

**Sample Output for CoursesTester.java**

17120 4 Calculus I

44543 3 Mobile Computing

44525 3 Theory & Implementation of Programming Languages

44542 3 Object-OrientedProgramming

44242 3 Data Structures

44241 3 Computer Programming II

44296 3 Professional Development Seminar

44141 3 Computer Programming I

8 courses in the list

deleting 17120 4 Calculus I

7 courses in the list