Lab 1

Due Feb 6 by 11:59pm **Points** 50 **Submitting** a text entry box or a file upload
Available Jan 23 at 12am - May 21 at 11:59pm 4 months

CIS 27: Data Structures and Algorithms

Lab 1

Due Thursday, February 6 at 11:59 PM

To get started, see **Using Eclipse**.

All programs must be:

- **properly documented** (TODO)
  - contracts? precondition/postcondition?
- **follow Java coding practices (as explained in Chapter 1.1 of the textbook)**
- **and include unit tests that show how you verified that your program works (expected input/output).**
  - Your unit tests should be in the main() method. See p. 26 of the textbook.

1. **Linked Lists – Deque (Double-ended queue) (15 points)**

Implement a class Deque for building doubly-linked lists. The class should have a **nested** class inside it, DoubleNode, where each node contains a reference to the item preceding it and the item following it in the list (null if there is no such item). Then implement methods for the following tasks:

- Insert at the beginning
- Insert at the end
- Remove from the beginning
- Remove from the end
- Insert before a given node
- Insert after a given node
- Remove a given node
- Move to front (move an object to the front)
- Move to end (move an object to the end)

2. **Stacks – Evaluating Arithmetic Expressions (20 points)**

   Write a class ArithmeticExpressionEvaluator that evaluates an infix arithmetic expression. Do this is two steps. First, convert the infix expression to a postfix expression: create a filter InfixToPostfix that converts an arithmetic expression from infix to postfix. Second, evaluate the postfix expression: write a postfix evaluator EvaluatePostfix that takes a postfix expression, evaluates it and print the value.

   Your program should print the infix expression, postfix expression, and the final result.

3. **Union-Find - Maze (15 points)**

Write a program that generates mazes of arbitrary size using the union-find algorithm. A simple algorithm to generate the maze is to start by creating an N x M grid of cells separated by walls on all sides, except for entrance and exit. Then continually choose a wall randomly, and knock it down if the cells are not already connected to each other. If we repeat the process until the starting and ending cells are connected, we have a maze. It is better to continue knocking down the walls until every cell is reachable from every cell as this would generate more false leads in the maze.

Test your algorithm by creating a 10 x 10 grid, and print all the walls that have been knocked down. Draw the resulting maze (hand-drawing is acceptable)

How to submit your solutions

In the text box, answer the following questions, all of which are required:

- Which partner(s) did you work with? (I expect most students will work in pairs, but there may be groups of 3.)
- Rate your lab partner(s) in the context of this lab, on a scale of 1 to 5: 1 is the least helpful, 5 is the most helpful. Explain your rating.
- Rate yourself in the context of this lab, also on a scale of 1 to 5. Explain your rating.

Also, attach all your Java files to your submission as separate attachments. You and your partner should submit the same code, but you will each submit your own assignment on Canvas so you can rate each other.

Depending on ratings, you and your partner may receive different grades on the lab.

Rubric:

TBD

File Upload | Text Entry | Google Doc | Dropbox | Google Drive | Office 365

Upload a file, or choose a file you've already uploaded.

File: Choose File no file selected

+ Add Another File

Click here to find a file you've already uploaded

Comments...

Cancel Submit Assignment