

2110 Lab 1
Instructor: Travis Gagie

Posted January 21st, 2022
Due January 27th, 2022 at 23:59 AST

This lab is broken into 3 questions that are meant to help enhance your knowledge of the data structures we covered in class and their implementations. Question 1 and 2 are worth 25% and question 3 is worth 50% of the lab grade. We will give you files L1Q1.java, L1Q2.java, and L1Q3.java which contain a completed main method that you can use to test your code. You will be responsible for implementing the required methods and classes for the data structure. This main function will not be changed nor will it need to be changed in order to complete this lab. Read the entire lab carefully because there will be information and requirements you may not be familiar with.

Pseudo-code: A pseudo-code implementation can be found on Brightspace in lecture 6 of Alex's Lectures section.

Style: We will follow a relaxed style for labs. We expect proper whitespace, variable names, and simple comments to describe anything complex. This will be a single mark for each lab question and will not be given out partially.


Submission and Mimir: The Mimir IDE will be **disabled**, and submissions will be done by uploading files from your computer.

Efficiency: You will learn more about algorithmic time complexity (efficiency) as the course progresses, however, for this lab the efficiency requirement will be simpler until this topic is covered in class. Therefore, for each question, ***The Student Cannot Use Nested Loops***. This will ensure no time complexity beyond $O(n^2)$; a concept you will learn about later.

Academic Integrity: If the markers suspect any part of any students' labs that is not given, copied, or unoriginal content, the academic integrity office ***must*** be notified. We do not want to do this but we have comprehensive tools that are developed specifically for catching plagiarism (i.e., Moss), so you ***will*** be caught if you try. Therefore, please do your own work; it is worth taking a 0 on several labs instead of failing the course or getting expelled!

Question 1

Using Java, implement a **Stack** data structure with the following functions:

```
C: > Users > hyper > Downloads >  2110_assignment_1.tex
1  ✓ private static class Stack<T> {
2      // Instantiate variables
3
4      // Constructor
5      private Stack() {
6      }
7
8      // Inserts an item at the top of the stack
9      private void push(T item) {
10     }
11
12     // Returns an item to LOOK at but not remove from the stack
13     private T peek() {
14     }
15
16     // Removes the first item on the top of the stack
17     private T pop() {
18     }
19
20     // Checks if a stack is empty and returns true or false
21     private Boolean isEmpty() {
22     }
23 }
24
```

As discussed in class, a stack operates on FILO (First In, Last Out) which means the first item put into the stack will be on the bottom similar to stacking books on top of each other.

The submitted file should be L1Q1.java

Question 2

Using Java, implement a **Queue** data structure with the following functions:

```

C: > Users > hyper > Downloads > TeX 2110_assignment_1.tex
1  private static class Queue<T> {
2      // Instantiate variables
3
4
5      // Constructor
6      private Queue() {
7
8      }
9
10     // Inserts an item at the last position of the Queue
11     private void enqueue(T item) {
12
13     }
14
15     // Removes and returns the first item in the Queue
16     private T dequeue() {
17
18     }
19
20     // Returns the first item in the Queue but does not remove
21     private T peek() {
22
23     }
24
25     // Checks if a stack is empty and returns true or false
26     private Boolean isEmpty() {
27
28     }
29 }

```


As discussed in class, a Queue operates on FIFO (First In, First Out) which means the first item put into the stack will be first in line to be taken out – like waiting in line at a checkout.

The submitted file should be L1Q2.java

Question 3

Using Java, implement a ***Doubly Linked List*** data structure with the following functions:

```
C: > Users > hyper > Downloads > TeX 2110_assignment_1.tex
1  private static class DoublyLinkedList<T> {
2      private class Node<T> {
3          private T item;
4          Node prev;
5          Node next;
6
7          private Node(T item, Node prev, Node next) {
8              this.item = item;
9              this.prev = prev;
10             this.next = next;
11         }
12
13         public void setPrev(Node prev) { this.prev = prev; }
14         public void setNext(Node next) { this.next = next; }
15
16         public Node getPrev() { return prev; }
17         public Node getNext() { return next; }
18         public T getItem() { return item; }
19     }
20
21     // Instantiate variables
22
23
24     // Constructor
25     private DoublyLinkedList() {
26
27     }
28
29     // Returns the first item at the head of the doubly linked list
30     private T getFirst() {
31
32     }
33
34     // Returns the last item at the tail of the doubly linked list
35     private T getLast() {
36
```

C: > Users > hyper > Downloads >  2110_assignment_1.tex

```

36
37     }
38
39     // Inserts an item before the first item (or head) of the doubly linked list
40     private void addFirst(T item) {
41
42     }
43
44     // Inserts an item after the last item (or tail) of the doubly linked list
45     private void addLast(T item) {
46
47     }
48
49     // Removes the first item (head) of the doubly linked list
50     private T removeFirst() {
51
52     }
53
54     // Removes the last item (tail) of the doubly linked list
55     private T removeLast() {
56
57     }
58
59     // Checks if a stack is empty and returns true or false
60     private Boolean isEmpty() {
61
62     }
63
64     // Returns the count of the current number of nodes in the doubly linked list
65     private int count() {
66
67     }
68 }
69

```

As discussed in class, a doubly linked list is identical to a linked list but instead of each node pointing only to its next node, each node points in both directions – both to its next node and to its previous node. A doubly linked list is almost always used to implement a Deque and so, if you have implemented a doubly linked list, you have nearly everything you need to make a Deque. You are given the Node class to make this assignment easier and more specific to doubly linked list implementation but you should be familiar or familiarize yourself with creating a Node class because you will use them very frequently.

The submitted file should be L1Q3.java

Marking

Question 1 & 2 will be worth 6 points each and Question 3 will be worth 11 points. 1 point of each question will be allocated to style and the others are all functionality. When you upload your file to Mimir, it will automatically grade you against pre-made test cases and award points for every test passed. From those functionality points, you will lose 100% if you did not implement and use the intended data structures that were required for the lab and you will lose 20% if your program is not meeting the efficiency requirement (no nested loops) and takes longer than intended to complete. Here is a breakdown of marks for each component:

1. Functionality (10 points) **All percentages are taken from the total net functionality points**

- Using intended data structures

0% if implemented

-100% if not implemented

- Efficiency

0% The student didn't use nested loops

-20% The student used nested loops

2. Style (1 points) **There has to be a reasonable attempt to solve the problem to gain style points.** Provide appropriate whitespace (spacing and blank lines), good variable names, and comments in areas where code is unclear or complex.

1/1 Readable based on whitespace and naming with needed comments

0/1 Did not make a reasonable attempt at question or style