Name: _____ Andrew Id: _____

15-121 Fall 2025 Assessment 3

Up to 60 minutes. No calculators, no notes, no books, and no computers. Show your work!

1. **Short Answer**

   Answer the following in 1 sentence; longer answers will be marked incorrect.

   (a) (2 points) How does a Java interface relate to the concept of an ADT?

   (b) (2 points) Assuming an initially empty queue, what value is returned by the `peek()` operation after performing the following operations are all executed in order:

```
enqueue("Alice")
enqueue("Bob")
enqueue("Carol")
dequeue()
enqueue("David")
dequeue()
peek() // Your answer should be the string returned by this peek operation
```

   (c) (2 points) What is the advantage of a doubly linked list over a singly linked list? Give a specific operation that is more efficient with a doubly linked list and explain why it is more efficient.

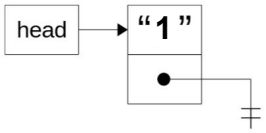
2. **Linked List Code Tracing**

Consider the following program that creates a singly linked list. You may assume that the `ListNode` class exists and was defined as in class.

```
1   ListNode<Integer> head = null;
2   ListNode<Integer> a = new ListNode<Integer>(1);
3   ListNode<Integer> b = new ListNode<Integer>(8);
4   ListNode<Integer> c = new ListNode<Integer>(7);
5   ListNode<Integer> d = new ListNode<Integer>(4);
6   head = a;
7   a.next = b;
8   b.next = c;
9   c.next = d;
10  head.next.next = d;
11  head.next.next.next = new ListNode<Integer>(9);
12  head.next = c;
13  head.next.next.next = a;
14  c.next = null;
15  head = b;
```

Draw the state of the linked list after the execution of each specified line of code. (The linked list is defined as starting at `head`.) The first one is drawn for you.

(a) After Line 6



(b) (3 points) After Line 9

(c) (3 points) After Line 10

(d) (3 points) After Line 11

(e) (3 points) After Line 12

(f) (3 points) After Line 14

(g) (3 points) After Line 15

3. **PlayList Queue**

Imagine that you are working on a music streaming application and you are in charge of implementing the playlist queue functionality. When users add songs to their queue, Song objects are created and added to the PlaylistQueue. Each Song contains information like title and artist. The queue should process songs in FIFO order (first song added plays first). However, users often accidentally add the same song multiple times in a row, and the system needs to be able to remove these consecutive duplicates to improve the listening experience.

The `PlaylistQueue` abstract data type has the following functionality:

1. Returns whether or not the queue is empty.
2. Returns the total number of Songs currently in the queue.
3. Adds a Song to the end of the queue.
4. Returns the next Song *without* removing it from the queue.
5. Removes and returns the next Song from the front of the queue.
6. Remove all consecutive duplicate songs from the queue. For example, if the queue contains: A $\rightarrow$ A $\rightarrow$ B $\rightarrow$ C $\rightarrow$ C $\rightarrow$ C $\rightarrow$ D $\rightarrow$ A, after removing consecutive duplicates, it should contain: A $\rightarrow$ B $\rightarrow$ C $\rightarrow$ D $\rightarrow$ A. Note that only consecutive duplicates are removed.

(a) (8 points) Write an interface, `PlaylistQueue`, based on the description above. You can choose the method names yourself, but you should make sensible choices. In this part, you are only writing an interface. You can assume that the class Song already exists, has a proper equals() method implemented, and stores information about a single song.

(b) (18 points) Write a class (name it whatever you want) that implements the `PlaylistQueue` interface. Your implementation must use a singly linked list as the way it stores the data.

All methods except the one to remove consecutive duplicates must be $O(1)$. The method to remove consecutive duplicates may be $O(N)$.

To save you time, you may assume that a Node class, designed for this problem, already exists:

```
public class Node {
        public Song data;
        public Node next;

        Node(Song data) {
            this.data = data;
            this.next = null;
        }
    }
}
```

Hints:

- If a method should return a Song, but there is no Song to return, then throw a `NoSuchElementException`.
- Make a plan for how you will store your data before you start writing code.
- You should keep both a head (front) and tail (back) pointer for efficient queue operations.
- You are writing an entire class, so make sure to include a proper class declaration.

Additional Space for writing the class

Additional Space for writing the class