Name: _____ Andrew Id: _____

15-121 Fall 2023 Assessment 3

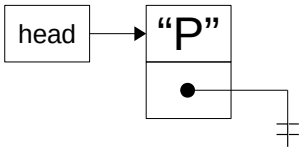
Up to 50 minutes. No calculators, no notes, no books, no computers. Show your work!

1. 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  public class A3CT {
2      public static void main(String[] args) {
3          ListNode<String> head = null;
4          ListNode<String> a = new ListNode<String>("Z");
5          ListNode<String> b = null;
6          ListNode<String> c = new ListNode<String>("A");
7
8          head = new ListNode<String>("P");
9          a.next = head;
10         b = head;
11         head = a;
12         head.next = new ListNode<String>("T");
13         head.next.next = b;
14         head.next.next.next = c;
15         b.next = head;
16         head = b;
17         head.next.next = null;
18     }
19 }
```

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 8



(b) (4 points) After Line 11

(c) (4 points) After Line 13

(d) (4 points) After Line 14

(e) (4 points) After Line 17

2. *Photo Tweaker*

Imagine that you are working on a picture editing program called *Photo Tweaker* and you are in charge of handling the undo functionality.

As the user of the program works and makes changes to their picture, `EditAction` objects are created. Each `EditAction` contains the details of a single edit the user makes to the picture. `EditAction` objects are added to an `UndoStack` in the order that the actions happen. When a user wants to undo their most recent action, they click the "Undo" option in the interface, and the most recent `EditAction` is popped off of the `UndoStack` and the program uses the information in it to undo that action. *Photo Tweaker* only allows you to undo the last 50 actions.

You have been given the following specifications for the `UndoStack` data type:

1. It should have a method that adds an `EditAction` to the top of the stack. This method should also ensure that there are, at most, 50 items on the `UndoStack` at a time. (If there are more than 50 items on the `UndoStack` after adding the new one, then the oldest item should be removed.)
2. It should have a method that pops the most recent `EditAction` from the stack and returns it.
3. It should have a method that returns how many `EditAction`s are currently on the `UndoStack`.

(a) (6 points) Write an interface, `UndoStack`, based on the description above. (It will have three methods.) 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 `EditAction` already exists and stores information about a single edit to the photo. (Hint: You don't care what is inside `EditAction`, you just need to push and pop them from your stack.)

(b) (18 points) Write a class (name it whatever you want) that implements the `UndoStack` interface and methods as described above. Your implementation must use a doubly linked list as the way it stores the data. For full credit, the methods should all be $O(1)$.

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

```
public class DoubleNode {
    public EditAction data;
    public DoubleNode next;
    public DoubleNode prev;

    public DoubleNode(EditAction data) {
        this.data = data;
        this.next = null;
        this.prev = null;
    }
}
```

Hints:

- We are not specifying how you should handle error cases. Handle them in any suitable way.
- Make a plan for how you will store your data before you start writing code.
- You should probably keep both a head and tail pointer if you are going to meet your efficiency requirements.
- You are writing an entire class, so make sure to include a proper prototype.

Additional space.