[**You may use a single side of an 8.5×11 in sheet of paper for reference.**] Show all of your work clearly in the space provided or on the additional page at the end of the exam. If the additional page is used, clearly identify to which exam question it is related. Be sure to **read each problem carefully**. Note that the exam is double sided. In this exam, wk5.ArrayList refers to the ArrayList implementation created in lecture, and wk6.LinkedList refers to the LinkedList implementation created in lecture.

1. (15 points) True/False (**T** or **F**)

______ Both Button and TextField are sources of action events.

______ Any of the following can have a fx:id attribute in an FXML file: Label, Button, and TextField.

______ Any of the following can have an onAction attribute in an FXML file: Label, Button, and TextField.

______ An @FXML variable in the controller class should be declared `static`.

______ One can convert from a File to a Path by casting. E.g., Path path = (Path) file.

______ If a method may encounter an IOException, the following must appear before the opening brace: `throws` IOException

______ Because the wk5.ArrayList is an indexed collection, you can access its elements using a subscript (the square brackets operator).

______ The inner Node class in the wk6.LinkedList implementation was declared as `static`.

______ The inner Node class in the wk6.LinkedList implementation was declared as `final`.

______ Primitive types cannot be used as a generic type.

______ A $O(2^n)$ algorithm always takes longer to complete than a $O(n)$ algorithm when run on the same hardware.

______ The add(E element) method for wk6.LinkedList class is $O(n)$.

______ The add(int index, E element) method for wk5.ArrayList class is $O(n)$.

______ The size() method for java.util.LinkedList class is $O(n)$.

______ The size() method for wk6.LinkedList class is $O(n)$.

2. (10 points) Without using any loops implement the following method that, given a list of integers, will return percentage of integers that are divisible by five.

```
public static double questionTwo(List<Integer> nums) {








}
```

3. Consider the following FXML file:

```
<VBox fx:controller="exam1.Controller" prefHeight="50" prefWidth="200"
          xmlns="http://javafx.com/..." xmlns:fx="http://javafx.com/fxml/1"
  <children>
    <HBox>
      <children>
        <TextField fx:id="addend1" />
        <Label text="+" />
        <TextField fx:id="addend2" />
      <children>
    </HBox>
    <Button onAction="#handleButton" text="Calculate" />
    <Label fx:id="result" />
  </children>
</VBox>
```

(a) (10 points) Sketch what the UI specified in the FXML file would look like.

(b) (15 points) Implement the controller class corresponding to the FXML file. Include everything necessary for the Java source file except `import` statements. When the button is activated, the program should add the two numbers together and display the result in the label. If anything other than numbers are present in the text fields, the program should display ”I can only add numbers” in the label.

4. (10 points) Implement the addMiddle(E element) method for the wk5.ArrayList that adds element to the size()/2th element. E.g., if the list is empty or has one element, element should be added at position 0; if the list has two or three elements, element should be added at position 1, etc... You may use size() from the wk5.ArrayList class, but no other methods.

5. Consider the following method:

```
public static void populateList(List<Integer> list) {
  list.add(null);
  list.add(3);
  list.add(1, 1);
}
```

(a) (10 points) Draw a memory diagram (similar to the ones drawn in lecture) showing the contents of the list after the method above is run when passed an empty wk6.LinkedList.

(b) (10 points) Draw a memory diagram (similar to the ones drawn in lecture) showing the contents of the list after the method above is run when passed an empty java.util.LinkedList.

6. (a) (10 points) Consider a method, addSecondToLast(E element), for the wk6.LinkedList class that adds an element to the second to last spot in the list. You will need a Node<E> reference to navigate the list. Draw a memory diagram of a wk6.LinkedList<Integer> that currently stores **1**, **2**, and **3**. In addition, show where the "walker" node is pointing immediately prior to inserting a value at the second to last position in the list.

(b) (10 points) Implement a method described in part **(a)**. The method must throw an IllegalStateException if the list is empty when called. You may not use any other methods in the LinkedList class.