1. (3 points) True/False (**T** or **F**) Based on course policies (https://csse.msoe.us/taylor/policy)

_____ It is acceptable to discuss ideas and approaches to solve lab assignments with others.

_____ It is acceptable to show your code to another student and ask for help debugging it.

_____ It is acceptable to take a photo of another student's code as long as you understand how it works.

2. (7 points)

```
public static void fileLineLengths(Path path) throws IOException {
```

1. (5 points) True/False (**T** or **F**)

_____ It is possible to have multiple catch blocks associated with the same try block.

_____ It is possible to have multiple finally blocks associated with the same try block.

_____ NullPointerExceptions should not be caught.

_____ IOExceptions should not be caught.

_____ Both `catch` and `catches` are keywords used for exception handling.

2. (5 points) List three classes used in JavaFX and explain the purpose of each.

1. (3 points) Add an event handler to the button below that displays "full points please" to the console when the button is fired.

```
Button button = new Button("Grade■Question");
```

2. (3 points) Add an event handler to the text field below that clears the text in the text field when the user hits enter.

```
public class QuizW4 extends Application {
  TextField textField = new TextField();

  public void start(Stage stage) {
    textField = new TextField();









  }
}
```

3. (4 points) Implement the controller class corresponding to the FXML file below. Include everything necessary for the Java source file except `import` statements.

```
<FlowPane fx:controller="wk4.QuizController" prefHeight="50" prefWidth="200"
          xmlns="http://javafx.com/..." xmlns:fx="http://javafx.com/fxml/1">
  <children>
    <Label fx:id="displayText" text="This is a label that contains text" />
    <Button fx:id="toggleButton" text="Blur" />
  </children>
</FlowPane>
```

```
public static void populateList(List<String> list) {
  list.add("I");
  list.add("E");
  list.add(1, null);
  list.add("o");
}
```

1. (5 points) Draw a memory diagram (similar to the ones drawn in lecture) showing the contents of the list after the method above is called passing in an empty wk5.ArrayList object.

2. (5 points) Implement the get(int index) method for the wk5.ArrayList. You may not use any other methods in the wk5.ArrayList class.

True/False (**T** or **F**)

_____ The Iterator.hasNext() method will throw a NoSuchElementException called on a completely empty collection.

_____ The List.remove() method is more efficient than the Iterator.remove() method since the list knows more about the underlying data structure than the iterator.

_____ The List.listIterator(index) returns an iterator that begins just before the position index.

_____ A ListIterator can be used to navigate both forward and backward over a Collection.

_____ The ListIterator is a subinterface of the Iterator interface.

_____ The enhanced for loop makes use of an iterator to navigate the collection.

_____ Iterator objects throw an IllegalStateException if they are asked to retrieve the next element after all elements have been processed.

_____ If a call to java.util.Iterator.remove() is not preceded by a call to next(), an IllegalStateException will be thrown.

_____ The Iterator interface declares the iterator() method.

_____ The Collection.forEach() method relies on an iterator to navigate the collection.

1. (2 points) If a method is annotated with @BeforeEach, what does that imply?

2. (3 points) Why did we create our own PureQueue interface instead of using the java.util.Queue interface?

3. (5 points) Consider the following test method for the AutoCompleter.add(String word) method. Implement the UnorderedList.add(String word) method such that the test method passes but the implementation is incorrect.

```
@Test
public void addTest() {
    AutoCompleter test = new UnorderedList(new ArrayList<>());
    Assertions.assertThrows(IllegalArgumentException.class, () -> test.add(null));
    Assertions.assertThrows(IllegalArgumentException.class, () -> test.add(""));
    Assertions.assertEquals(0, test.size());
    Assertions.assertTrue(test.add("word"));
    Assertions.assertEquals(1, test.size());
}
```

1. (4 points) Recall that the BST<E> class had one attribute, Node<E> root, that is a reference to an object defined by an inner class with three attributes: E value, Node<E> left, and Node<E> right. Implement the recursive size() method that is called by the method below:

```
public int size() {
  return size(root);
}
```

2. (4 points) Identify any errors in the following method and suggest how the errors, if any, could be corrected.

```
public static boolean binarySearchRec(List<String> list, String target) {
  boolean found = false;
  if (!list.isEmpty()) {
    int middle = list.size() / 2;
    int compare = target.compareTo(list.get(middle));
    if (compare == 0) {
      found = true;
    } else if (compare < 0) {
      found = binarySearchRec(list.subList(middle + 1, list.size() - 1), target);
    } else {
      found = binarySearchRec(list.subList(0, middle - 1), target);
    }
  }
  return found;
}
```

3. (2 points) Describe the difference between a *perfect* tree and a *complete* tree.

1. (3 points) Explain what an expression tree is and give an example.

2. (4 points) Describe how the Map and Set interfaces differ.

3. (3 points) How is the *load factor* for a hash table calculated? How is it used?