Practice Problems 4/4/2016

Part 1: Thoughtful Testing

1. Complete the testToString method with tests that will find the bug in the following toString method.

```
public String toString() {
      String returnString = "[";
      while(!isEmpty()) {
            returnString += pop().toString();
      }
      return returnString + "]";
}

public void testToString() {
      assertTrue(stack.toString().equals("[]"));
      stack.push("a");
      assertTrue(stack.toString().equals("[a]"));
      //TODO: add additional testing to find the bug in toString().
}
```

2. How would you fix the bug in the toString method? (Hint: Use the methods **size** and **get**)

```
public String toString() {

}
```

3. Assume we have a null variable **foo.** Which is the correct way to use assert statements?

a. assertEquals(null, foo);
b. assertTrue(foo == null);
c. assertNull(foo);
d. All of the above

Part 2: Sorted Lists

1. What method is required when implementing Comparable<T>?

2. The following are possible return values for **var1.compareTo(var2).** What are possible values for var1 and var2 for each return value?

a. positive integer
b. zero
c. negative integer

```
/**
  * Use the following selection sort algorithm for questions 3-5
  */
private void sort(int[] array) {
     for (int lh = 0; lh < array.length; lh++) {
          int rh = findSmallest(array, lh, array.length);
          swapElements(array, lh, rh);
     }
}
```

3. How many passes of selection sort will go through a list of 10 elements?

4. For the list, [20, 40, 10, 30], what will the array look like after each pass of selection sort?

5. For a list of n elements that is already sorted (i.e. [1, 2, 3, 4, 5]), what is the efficiency?

6. Which sorting algorithm(s) is/are the best for the average case?

7. Which sorting algorithm(s) is/are the best for the best case (the list is already sorted)?

8. Which sorting algorithm(s) is/are the best for the worst case (none of the list is already sorted)?

Part 3: Generics

1. For the generic constructor **public Foo(T data){}**, which are valid initializations?

```
a. Foo<Integer> intFoo = new Foo<>(1);
b. Foo<Integer> intFoo = new Foo(1);
c. Foo<Double> doubleFoo = new Foo<>(1.0);
d. Foo<Double> doubleFoo = new Foo(1.0);
e. Foo<String> stringFoo = new Foo<>("BAR");
f. Foo<String> stringFoo = new Foo("BAR");
g. Foo<Object> objectFoo = new Foo<>();
h. Foo<Object> objectFoo = new Foo();
```

2. What does ? represent in relation to generics?