CSE 331 19su Midterm Exam 7/22/19

Name _____ UW ID # _____

There are 11 questions worth a total of 100 points. Please budget your time so you get to all of the questions. Keep your answers brief and to the point.

The exam is closed book, closed notes, closed electronics, closed mouth, open mind.

Please do NOT remove any pages from the middle of the exam. There are extra copies at the end of the exam of the full pages of code that you can detach and reference during the exam if you want.

There is an additional blank page with extra space for your answers if you need more room. It is after all the questions but before the detachable pages.

Many of the questions have short solutions, even if the question is somewhat long. Don't be alarmed.

For all of the questions involving proofs, assertions, invariants, and so forth, you should assume that all numeric quantities are unbounded integers (i.e., overflow cannot happen) and that integer division is truncating division as in Java, i.e., 5/3 evaluates to 1.

If you don't remember the exact syntax of some command or the format of a command's output, make the best attempt you can. We will make allowances when grading.

Relax, you are here to learn.

Score _____ / 100

1. _____ / 5
2. _____ / 8
3. _____ / 16
4. _____ / 10
5. _____ / 12
6. _____ / 6
7. _____ / 9
8. _____ / 14
9. _____ / 4
10. _____ / 14
11. _____ / 2

Remember: For all of the questions involving proofs, assertions, invariants, and so forth, you should assume that all numeric quantities are unbounded integers (i.e., overflow cannot happen and there are no fractional parts to numbers) and integer division is truncating division as in Java, i.e., 5/3 => 1.

Question 1. (5 points) (Forward reasoning) Starting with the given assertion, insert appropriate assertions in each blank line. You should simplify your final answers if possible.

```
{ x > 3 }
x = x + 1;

{ ____________________________________ }
y = x * 10;

{ ____________________________________ }
x = x - 2;

{ ____________________________________ }
```

Question 2. (8 points) (Backward reasoning). Find the weakest precondition for the sequence of statements below to establish the given postcondition. Write appropriate assertions in each line and simplify your final answer if possible.

```
{ ____________________________________________________ }
if (x < y) {

   { ____________________________________________________ }
   x = x + 1;

   { ____________________________________________________ }
} else {

   { ____________________________________________________ }
   y = y - 1;

   { ____________________________________________________ }
}
{ x != y && x > 0 }
```

Question 3. (16 points) Loops. The following loop finds the two largest distinct values in an array `a` that has `n` elements. Your job is to prove that it works correctly and establishes the postcondition that is given at the end. According to the precondition, the array length `n` is at least 2, there are no duplicate values in the array. Write a suitable loop invariant and add the necessary assertions to complete the proof. To save writing, use the notation `a[i..j]` to refer to the array elements starting with `a[i]` and ending with `a[j]`. Hint: the Java expression `b?x:y` used below evaluates to `x` if `b` is true and to `y` if `b` is false. (It works correctly – you don't need to prove it.)

The code is spread out over this page and the next to provide lots of space for writing.

{ pre: n ≥ 2 && a has no duplicate elements }

```
// initialize max and max2nd (you can assume this is correct)
int max    = (a[0]<a[1]) ? a[1] : a[0];
int max2nd = (a[0]<a[1]) ? a[0] : a[1];
int k = 2;
```

{ inv: _____ }

```
while(k != n) {
```

{ _____ }

```
   if (a[k] > max) {
```

{ _____ }

```
      max2nd = max;
```

{ _____ }

```
      max = a[k];
```

{ _____ }

```
      …
```

(loop body continued on next page)

Question 3. (cont.) Body of while loop and proof continues below.

```
   …
   } else if (a[k] > max2nd) {
      { _____ }
      max2nd = a[k];
      { _____ }
   } else {
      // nothing needed
      { _____ }
   } // end if-elseif-else
   { _____ }
   k = k + 1;
   { _____ }
} // end loop
{ _____ }
```

{ post: max = largest in a[0..n-1] && max2nd = 2nd largest in a[0..n-1] }

The next several questions concern the following code, which contains a class that represents a shopping cart and a second class that represents items that can be added to the cart. There may be some logic bugs in the code (to be explored later), but it does compile and execute without any reported errors. There is a second copy of this page at the end of the exam that you can remove for convenience.

```
/** A ShoppingCart holds a list of n Items i1, i2, ..., in */
public class ShoppingCart {
  private List<Item> items;
  public ShoppingCart() { items = new ArrayList<Item>(); }
  public void addItem(Item item) {
    items.add(item);
  }
  public void applyDiscount(double scaleFactor) {
    for(Item item : items) {
      item.setPrice(scaleFactor * item.getPrice());
    }
  }

/** An item in a shopping cart */
public class Item {
  private String name;
  private double price;

  public Item(String name, double price) {
    this.name = name;
    this.price = price;
  }

  public String getName() { return this.name; }
  public double getPrice() { return price; }
  public void setPrice(double price) { this.price = price; }

  /** Items are considered equal if they have the same name.   */
  @Override
  public boolean equals(Object o) {
    if(!(o instanceof Item)) return false;
    Item other = (Item) o;
    return this.name.equals(other.name);
  }

  /** an attempt at a hashCode for an Item */
  @Override
  public int hashCode() {
    return (31 * Double.hashCode(price)) + name.hashCode();
  }
}
```

Do not remove this page from the exam, but feel free to tear off the copy of this page at the end of the exam. Continue with questions about this code on the next page.

Question 4. (10 points) As usual, whoever writes these exams doesn't provide proper specifications for things. Below, fill in a correct CSE 331-style specification for the constructor and the `applyDiscount` method in `ShoppingCart`. For CSE 331-specific custom tags, you can write `@spec.xyz` or just `@xyz` – whichever you prefer.

```
/** Construct a new, empty ShoppingCart
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
public ShoppingCart() { ... }

/** Apply a discount to the Item prices in this ShoppingCart
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
public void applyDiscount(double scaleFactor) { ... }
```

Question 5. (12 points, 3 each) Testing. Describe four distinct black-box tests that could be used to verify that the `applyDiscount` method specified in the previous problem works properly. Each test description should describe the test input and expected output. For full credit each test should be different in some significant way from the other tests (think about boundary conditions and subdomains, etc.). You **should not** provide JUnit or other code, just a clear, precise description of each test, and your descriptions should be a few lines each, at most. If you want to describe a specific `Item` as part of a test you can write it with the notation (name, price), i.e., (magicwand, 17.42).

(a) Input or test setup:

Expected output:

(b) Input or test setup:

Expected output:

(continued on next page)

Question 5. (cont.)

(c) Input or test setup:

Expected output:

(d) Input or test setup:

Expected output:

Question 6. (6 points, 2 each) We would like to add a `getTotal()` method to `ShoppingCart` that returns the total price of the items in the `ShoppingCart`. Here are three possible specifications for this new method.

Spec. A

```
/**
 * @return The total price of all items in the shopping cart.
 * @throws IllegalStateException if the shopping cart is empty
 */
```

Spec. B

```
/**
 * @return The total price of all items in the shopping cart.
 * @requires The shopping cart is not empty.
 */
```

Spec. C

```
/**
 * @return The total price of all items in the shopping cart, or $0 if there are none.
 */
```

Describe the relationship between each pair of specifications by circling the correct answer below:

a) **Spec A is** weaker than stronger than incomparable to **Spec B**

b) **Spec A is** weaker than stronger than incomparable to **Spec C**

c) **Spec B is** weaker than stronger than incomparable to **Spec C**

Question 7. (9 points, 3 each method) Here are three possible implementations of `getTotal()` for `ShoppingCart`. For each implementation, circle *all* of the names of the specifications from the previous page that it satisfies (if any).

```
public double getTotal1() {
    if(items.isEmpty()) {
        throw new IllegalStateException();
    }
    double total = 0.0;
    for(Item item : items) {
        total += item.getPrice();
    }
    return total;
}
```

Satisfies Spec A

Satisfies Spec B

Satisfies Spec C

```
public double getTotal2() {
    if(items.isEmpty()) {
        return -1.0;
    }
    double total = 0.0;
    for(Item item : items) {
        total += item.getPrice();
    }
    return total;
}
```

Satisfies Spec A

Satisfies Spec B

Satisfies Spec C

```
public double getTotal3() {
    double total = 0.0;
    for(Item item : items) {
        total += item.getPrice();
    }
    return total;
}
```

Satisfies Spec A

Satisfies Spec B

Satisfies Spec C

Question 8. (14 points). We've decided to add method `getTotal3` from the previous page to the existing `ShoppingCart` code, and we've written a small program to try it out. One of our lucky customers is given a discount. Here is the code:

```
  public static void main(String[] args) {
    ShoppingCart harrysCart = new ShoppingCart();
    ShoppingCart hermoniesCart = new ShoppingCart();
    Item apple = new Item("apple", 5.0);
    Item banana = new Item("banana", 2.5);
    harrysCart.addItem(apple);
    hermoniesCart.addItem(apple);
    hermoniesCart.addItem(banana);
    harrysCart.applyDiscount(0.8);
    /**** HERE!!! ****/
    System.out.println("Harry: " + harrysCart.getTotal3());
    System.out.println("Hermione: " + hermoniesCart.getTotal3());
}
```

(a) (6 points) Draw a diagram showing the variables and objects as they exist when execution of `main` reaches the first `println` statement (i.e., where the `HERE!!!` comment is, right after `applyDiscount` returns).

(continued on next page)

Question 8. (cont) (b) (2 points) There's a bug somewhere. In method `main`, the discount should only apply to `harrysCart` (that's what the client expects), but it appears to have also affected `hermoniesCart` too! What output is printed when the program is executed?

(c) (3 points) What is the bug? You should assume that the problem is *not* in the client code – `main` should work as written. Describe what went wrong in a couple of sentences. If this bug has a specific name be sure to include that in your description.

(d) (3 points) Describe an appropriate way to fix this bug, also briefly.

Question 9. (4 points) Class `Item` contains a `hashCode` method, but, as the comment in the code implies, it might not be correct. Does the given method satisfy the specification for `hashCode`? If so, give a brief justification for why it does; if not, describe what's wrong and how to fix it.

Question 10. (14 points, 2 each) Overloading, overriding, and `equals`. We've found the following code, which defines classes for 2-D and 3-D points, but doesn't quite get equality right – notice that the parameter types in the `equals` methods look suspicious. But the code does compile and execute without reporting any errors. Answer questions about this code on the next page. There is a second copy of this page at the end of the exam that you can remove for convenience.

```
/** Point on a 2-D plane with x,y coordinates */
public class Point {
  private int x, y;
  public Point(int x, int y) {
    this.x = x; this.y = y;
  }
  public boolean equals(Point o) {
    if (! (o instanceof Point)) {
      return false;
    }
    Point p = (Point) o;
    return this.x == p.x && this.y == p.y;
  }
}
/** Point on 3-D plane with x,y,z coordinates */
public class Point3d extends Point {
  private int z;
  public Point3d(int x, int y, int z) {
    super(x,y);
    this.z = z;
  }
  public boolean equals(Object o) {
    if (! (o instanceof Point)) {
      return false;
    }
    if (! (o instanceof Point3d)) {
      return super.equals(o);
    }
    Point3d p3 = (Point3d) o;
    return super.equals(p3) && this.z == p3.z;
  }
  public static void main(String[] args) {
    Point pta = new Point(1,2);
    Point ptb = new Point(5,6);
    Point3d p3a = new Point3d(1,2,3);
    Point3d p3b = new Point3d(1,2,4);
    Object o2a = pta;
    Object o3a = p3a;
    Object o3b = p3b;
    ______________________ ;  // insert code from questions here
  }
}
```

Do not remove this page from the exam, but feel free to tear off the copy of this page at the end of the exam. Continue with questions about this code on the next page.

Question 10. (cont.) For each line of code below, indicate what happens if it is inserted by itself at the end of the `main` method on the previous page and executed. For each one, indicate which method is called during execution (`Object.equals`, `Point.equals`, or `Point3d.equals`) and whether the method call returns true or false. Circle the correct answers.

(a) `pta.equals(ptb);`

Class whose `equals` method is executed: `Object` `Point` `Point3d`

Result: `true` `false`

(b) `pta.equals(p3a);`

Class whose `equals` method is executed: `Object` `Point` `Point3d`

Result: `true` `false`

(c) `p3a.equals(pta);`

Class whose `equals` method is executed: `Object` `Point` `Point3d`

Result: `true` `false`

(d) `p3a.equals(p3b);`

Class whose `equals` method is executed: `Object` `Point` `Point3d`

Result: `true` `false`

(e) `o3a.equals(p3a);`

Class whose `equals` method is executed: `Object` `Point` `Point3d`

Result: `true` `false`

(f) `o2a.equals(p3a);`

Class whose `equals` method is executed: `Object` `Point` `Point3d`

Result: `true` `false`

(g) `o3a.equals(o3b);`

Class whose `equals` method is executed: `Object` `Point` `Point3d`

Result: `true` `false`

Question 11. (2 free points) (All reasonable answers receive the points. All answers are reasonable as long as there is an answer. ☺)

(a) (1 point) What question were you expecting to appear on this exam that wasn't included?

(b) (1 point) Should we include that question on the final exam? (circle or fill in)

Yes

No

Heck No!!

$!@$^*% No !!!!!

No opinion / don't care

None of the above. My answer is _____.

Additional space for answers if needed. Please indicate clearly which questions you are answering here, and also be sure to indicate on the original page that the rest of the answer can be found here.