CSE 331 18su Midterm Exam

Name _____

Welcome to the 331 Midterm!

Please wait to turn the page until everybody is told to begin.

Friendly reminders:

1. **Attempt every problem.** You can receive partial credit.
2. **Write clearly!** We can't give you credit for answers that we can't read.
3. **If you make a mistake** and need to correct it, be sure that your final answer is very clearly indicated.
4. **If you run out of room** for any problem, you may continue on the last page. At the original location, be sure to indicate that your answer continues on the back.

The exam has 6 parts with the following point values:
Part 1 (26 points)
Part 2 (14 points)
Part 3 (20 points)
Part 4 (12 points)
Part 5 (16 points)
Part 6 (4 points)

This is an opportunity to show off what you have learned. Good luck and have fun!

Part 1: Hoare Logic (26 points)

Circle the correct answer in each row. (2 points each)

1.1	P: x > 5 Q: x >= 3	P is stronger than Q	Q is stronger than P	P and Q are incomparable
1.2	P: x % 10 = 0 Q: x % 5 = 0	P is stronger than Q	Q is stronger than P	P and Q are incomparable
1.3	P: | x - 1 | >= 2 Q: x >= 2	P is stronger than Q	Q is stronger than P	P and Q are incomparable
1.4	P: x > x Q: x = x	P is stronger than Q	Q is stronger than P	P and Q are incomparable

1.5 (18 points) Let a be an array of floats. Write code to compute the average value in a and use Hoare Logic to prove that your code is correct. Your code must use a non-trivial loop.

End of Part 1. I believe in you!!

Part 2: Specifications (14 points)

2.1 (4 points) Ada works for a small company, and Chandra is their customer. In January, Ada and Chandra had a meeting to discuss an application that the company would implement for Chandra. After the meeting, Ada wrote down a specification for the app. Over the next two months, Ada and teammates worked hard to implement the app. In March, Ada met with Chandra again to demonstrate the new software. Chandra was not pleased with the app. Here is a snippet of dialogue from the meeting:

Chandra: This software is wrong!
Ada: How is that possible? It's implemented exactly according to spec!

Assuming that the software does indeed match the spec, describe at least one idea for what the problem might be. Justify your answer(s).

Let **`IntStack`** be a class that represents a stack of integers. Consider the following specification for a method of the **`IntStack`** class:

```
/**
 * Look at the Integer that is n places from the top of the stack
 * without removing it from the stack. peek(0) looks at the top
 * Integer in the stack.
 * @param n - number of places from the top of the stack to peek
 * ...
 */
public Integer peek(int n)
```

The "..." in the spec above will be replaced with S1, S2, or S3.

(turn the page)

S1

```
@return the Integer n places from the top of the stack,
or null if n >= stack.size()
```

S2

```
@requires n < stack.size()
@return the Integer n places from the top of the stack
```

S3

```
@return the Integer n places from the top of the stack
@throws IllegalArgumentException if n >= stack.size()
```

Circle the correct answer in each row (2 points each):

2.2

S1 stronger than S2 S2 stronger than S1 S1 and S2 incomparable

2.3

S1 stronger than S3 S3 stronger than S1 S1 and S3 incomparable

2.4

S2 stronger than S3 S3 stronger than S2 S2 and S3 incomparable

2.5 (4 points) Here is another potential spec for **`peek`**. Is it valid? If not, explain why.

S4

```
@requires stack.size() >= n-1
@return the Integer n places from the top of the stack
@throws IllegalArgumentException if stack.size < n-1
```

You made it through Part 2. Are you having fun yet?

Part 3: Abstract Data Types (ADTs) (20 points)

3.1 (2 points) Complete the analogy.

Method Specification is to **Method Implementation** as **Abstract Data Type** is to

_____

3.2 (6 points) Which of the following are parts of an Abstract Data Type? For each thing, write "yes" if the thing is part of an ADT or "no" if it is not part of an ADT.

Class-level Javadocs _____

Public method Javadocs _____

Private method Javadocs _____

Abstraction Function _____

Representation Invariant _____

Method bodies _____

Public method signatures _____

Private method signatures _____

Private fields _____

Internal comments _____

Class declaration _____

3.3 (4 points) What is a representation invariant? Why is a representation invariant needed?

Consider the class **`AxisAlignedRectangle`**:

```
/**
* A quadrilateral with four right angles whose edges are parallel
* to the x and y axes.
*/
public class AxisAlignedRectangle {
  private double a, b, c, d;
  // Abstraction Function:
  // For example, if a=1, b=2, c=3, and d=4, it represents the
  // rectangle with vertices
  //   (1,2), (1,4), (3,2), and (3,4)
  ...
}
```

3.4 (4 points) Describe what is wrong with the abstraction function for AxisAlignedRectangle.

3.5 (4 points) Below, write a valid abstraction function for AxisAlignedRectangle.

Finished with Part 3. You're more than halfway through!

Part 4: Design (12 points)

4.1 (4 points) Consider the **`AxisAlignedRectangle`** class from Part 3. Identify four bad variable names from that class, and suggest better names for each one.

Consider the following code:

```
public String stringify(String a, String b, boolean flag) {
  StringBuilder ret = new StringBuilder();
  // strip characters in b from a
  if (flag) {
    for (int i = 0; i < a.length(); i++) {
      if (!b.contains(a.get(i))) {
        ret.append(a.get(i));
      }
    }
  }
  // concatenate b to the end of a
  else {
    ret.append(a);
    ret.append(b);
  }
  return ret.toString();
}
```

4.2 (8 points) Describe at least two design issues from **`stringify`** and write a specific and detailed description of how you would fix them. (You do not need to actually rewrite the code.)

End of Part 4. Great progress so far!

Part 5: Testing (16 points)

5.1 (4 points) What is wrong with this statement: "All the tests passed, therefore my program is correct!"

Consider the following method:

```
/** Find the maximum value in the array.
 * @param arr - an array of ints
 * @requires arr.length > 0
 * @return the minimum i such that arr[i] >= arr[j] for all j
 *    that are valid indices of the array
 */
public static int indexOfMax(int[] arr) {
  int max = Integer.MIN_VALUE;
  int idxOfMax = -1;
  for (int i = 0; i < arr.length-1; i++) {
    if (arr[i] > max) {
      max = arr[i];
      idxOfMax = i;
    }
  }
  return idxOfMax;
}
```

5.2 (12 points) Write at least three test cases for the **`indexOfMax`** method. In your test cases, you should identify inputs and expected outputs, but do not write actual test code. All of your tests must be for boundary cases. For each test, describe why it is a boundary case.

Part 6: Equals and Hashcode (4 points)

Consider the following code:

```
public class FluffyKitten extends Cat implements Fluffy {
  ...
  public boolean equals(FluffyKitten kitten) { ... }
}
```

6.1 (2 points) What is the problem with the signature for the equals method?

6.2 (2 points) Rewrite the signature for the equals method.

Congrats! You made it to the end! I hope you enjoyed this exam :)

Room for more answers

Be sure to clearly indicate which question(s) you are continuing here.