

Your Name: \_\_\_\_\_

page: 1

## **6.170 Final Quiz Solutions**

**April 15, 2003**

Circle the name of your original (problem set) recitation instructor,  
and underline the name of your new (project) recitation instructor:

Yuriy Brun

Connie Cheng

Naveen Goela

Greg Harfst

Lee Lin

Jennifer Louie

Matthew Notowidigdo

Eugene Shih

Zeehsan Syed

Stefanie Tellex

You have 50 minutes to complete this quiz.

There are 9 questions, 1 per page.

Answer all questions in the space provided.

Please write your name on the top of each page.

Good luck.

1) Provide an appropriate loop invariant and decrementing function for: (10 points)

```
assert i > 0 && j == 0 && i == x
    while (i > 0) {
        j = j + x;
        i = i - 1;
    }
assert j == x2
```

Decrementing function: i

5 points for this.

3 points for other functions that decrement but are not as helpful in proofs, such as  $i-1$ ,  $j-x^2$ ,  $x+i$ , or  $xi$ .

0 points for  $i=i-1$ .

Invariant:  $(j = x^2 - (i*x)) \ \& \ 0 \leq i$

or  $(j = x(x-i)) \ \& \ 0 \leq i$

Or  $(x^2 = j + i*x) \ \& \ 0 \leq i$

or  $(j=kx) \ \& \ 0 \leq i$  where k is the number of turns run so far

5 points for this. Full credit if " $0 \leq i$ " is omitted.

Otherwise no partial credit.

Your Name: \_\_\_\_\_

page: 3

2) What is the value of  $\text{wp}(x := f(y); y := 3, x = x + 1)$ ? Note that “:=” is a programming language assignment operator, and “=” is mathematical equality. (10 points)

Answer: false

Full credit for  $f(y) = f(y) + 1$

9 points for  $f(3) = f(3) + 1$

8 points for “undefined” (if they show work concluding the formula cannot be satisfied)

3 points for  $\text{wp}(x := f(y), x = x + 1)$

1 point for  $\text{wp}(y := 3, \text{wp}(x := f(y), x = x + 1))$

3) Below is part of a specification of an abstract type:

Overview: IntSets are finite sets of ints

```
public IntSet ( )
    effects: creates a fresh, empty IntSet
public void insert (int x);
    modifies: this
    effects: this' = this U {x}
public void delete (int x);
    modifies: this
    effects: this' = this - {x}
public boolean member (int x);
    returns: (x ∈ this)
```

Given the following excerpt from a purported implementation of the abstraction, provide an abstraction function and a rep invariant that are sufficient to make a rigorous argument that the implementation of **member** satisfies the specification of the method. You do not need to provide the argument itself – just the abstraction function and rep invariant.

(16 points)

```
public class IntSet {
    private ArrayList a;
    ...
    private int getElt(int i) {
        return ((Integer) a.get(i)).intValue( );
    }
    public boolean member (int x) {
        for (int i = 0; i < a.size( ); i++) {
            if (getElt(i) == x) return true;
            if (getElt(i) > x) return false;
        }
        return false;
    }
}
```

8 Points for: Inv:  $((0 \leq i, j < a.size()) \ \& \ i < j) \Rightarrow \text{getElt}(i) \leq \text{getElt}(j)$

full credit if they write  $0 \leq i, j < a.size() \ \& \ i < j) \Rightarrow a.get(i) < a.get(j)$

full credit if they don't bother to give the bounds for  $i, j$

full credit if they write “elements of a are in ascending order”

4 points if they write “elements of a are sorted”

no credit if they don't mention ordering of a at all (e.g., only mention that  $a \neq \text{null}$ )

Don't worry about type checking, casts, or Java syntax (e.g., “()” after methods)

8 Points for: Abst fnc:  $\{x \mid (a.get(i) = x) \ \& \ (0 \leq i < a.size)\}$

or:  $\{a.get(i) \mid 0 \leq i < a.size\}$

Full credit if second conjunct is omitted

Your Name: \_\_\_\_\_

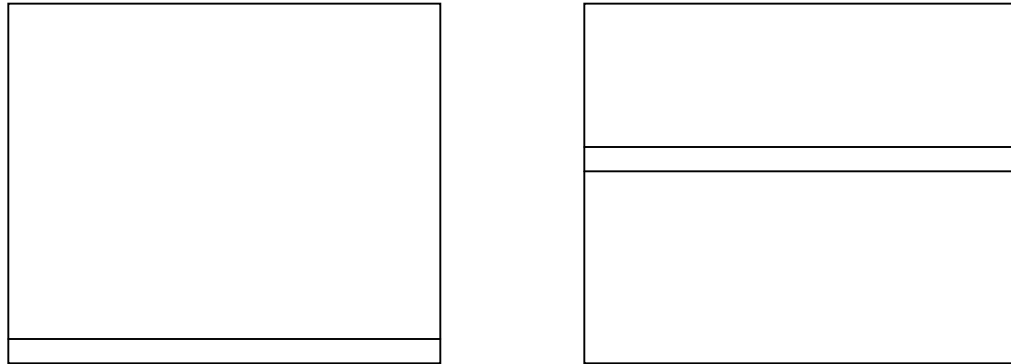
page: 5

4 points if they include irrelevant information that belongs in RI

4 points if they have a function from concrete state, but don't mention abstract state

3 points if they just give an example mapping, but don't generalize to a function

4) Consider the following two placements of menus to be selected using a mouse. In the design on the left, the menu is at the bottom of the screen. In the design on the right it is near the middle. Using Fitt's law,  $T = a + b \log (D/S + 1)$ , argue, in 2-3 sentences, that the placement on the left is superior. (8 points)



Solution: The term  $D/S$  stands for distance/size. Fitt's Law states that the time required to reach a target is proportional to the distance from the cursor to the target and inversely proportional to the size of the target. Even though the interface on the right side is likely to have a smaller distance to the target, the interface on the left has a much larger target size. When a user attempts to click on a button aligned with the bottom edge of a screen, the size of the target includes all of the space below the button because the edge of the screen prevents overshooting the target. If  $S$  is considered very large, then Fitt's Law reduces to  $T = a + b \log (D / S + 1) = a$ . The large  $S$  easily compensates for slightly longer distances.

#### Grading Guidelines:

4 points for : mentioning that the size of the target is very large or infinite for a button aligned at the edge of the screen.

OR: mentioning that a user can just "whip the cursor" as far down as possible without worrying about overshooting.

HALF-PTS: Mentioning that the button is easier to click when at an extreme section of the screen.

NO-PTS : Arguing that the left interface gives more usable work space or is less confusing.

4 points for: demonstrating a clear understanding of Fitt's Law, by clearly stating what items are  $T$ ,  $S$ , and  $D$  and arguing  $S$  is much larger on the left interface.

Your Name: \_\_\_\_\_

page: 7

OR: For simplifying the Fitt's Law equation for the left interface by eventually saying  $T = a$ .

OR: For saying "Although the right interface might have a smaller  $D$ , the left interface has a much higher  $S$  which easily compensates for the higher distances."

NO-PTS: For confusing  $D$  and  $S$  (lots of students thought  $D$  changed,  $S$  same).

5) If one overrides the `equals` method inherited from `Object` for a type being stored in a `HashSet`, one should also override `hashCode`. Explain why in 2-4 sentences. (10 points).

The `HashSet` specification depends on the property “`a.equals(b) ⇒ a.hashCode()==b.hashCode()`”; overriding `equals` (changing the equivalence relation it defines) can violate this property, so one must then override `hashCode` to re-establish it (and ensure that equal values in the new equivalence relation have the same hash key).

Failing to override `hashCode` may cause lookup to give the wrong answer. In particular, it is possible to insert an element `a`, then for a lookup with an element `b` such that `a.equals(b)` but `a.hashCode()!=b.hashCode()` not to find element `a`. In other words, looking up `a` gives a different answer than looking up `b`.

Full credit if they only give the second sentence of my answer.

3 points if they give only the first clause of my last sentence.

0 performance argument alone

“`HashSet` fails”

calling lookup on same object will return false

2 Calling `HashSet.contains()` fails or returns false (without explanation)

4 Calling `HashSet.contains()` fails with the following explanation: 2 different objects that are `.equals()` will return false, but should return true

7 `a.equals(b) ⇒ a.hashCode() == b.hashCode()`

+1 If you don't override `hashCode`, then `Object.hashCode` will be used. `Object.hashCode()` relies on `==`, which is a reference to memory. Thus, two objects (even if they have the same abstract value) will return different `hashCodes`.

+3 Calling `HashSet.contains()` fails with the following explanation: 2 different objects that are `.equals()` will return false, but should return true, thus override `hashCode()`

-2 incorrect information is added

-3 `a.equals(b) ⇒ a == b`

6) Consider a project design that breaks the project up into hundreds of components. Assume that two different teams have been asked to construct implementations of the design. Assume that for each project on average each step of the process takes exactly as long as predicted, but that some steps take longer than expected and some less. Assume that the only difference in the two projects is that:

For team A, the variation in the time taken to implement and unit test individual components is evenly distributed between  $\pm 10\%$  of the predicted time.

For team B, the variation in the time taken to implement and unit test individual components is evenly distributed between  $\pm 20\%$  of the predicted time.

Which team would you expect to finish first? In 1-3 sentences justify your answer. (10 points)

Team A is expected to finish first.

In the development process, some tasks depend on others and cannot begin (or complete) until they complete. When a component depends on multiple other components, its start delay is the *maximum* of all their completion delays; this is likely to be near the worst-case delay, so the expected case is the worst case. Overall completion time is the sum (accumulation) of all completion time per "level". Since all of these are greater for B (because B has a worse worst-case), B finishes later.

+3 : discussing dependencies

+5 : saying delay at each level is the max delay over all the components at that level

+2 : saying overall completion time is sum of all completion times

No points for just saying A.

1/2 credit (round up) for each part if student gives example and covers the points above but does not generalize. Give full credit if they manage to generalize.

1/2 credit (round up) if they gave worse case scenario, then concludes B takes longer

1/2 credit (round down) if they give vague answers the cover main points above

5 points: if they assume the dependence structure is serial, with no fan-in at all ([ ] -> [ ] -> [ ] -> [ ]), and say A and B are expected to take the same time and mention accumulation of completion times is why.

7) The items in the left column are mechanisms or techniques. The items in the right column denote uses of these mechanisms. Next to each item in the left column, write the number of the item in the right column that most closely describes a use of the mechanism. Items in the right column should not be used more than once. (2 points each).

(We have filled in the first answer for you, and will even give you credit for getting it right.)

|                                |                          |
|--------------------------------|--------------------------|
| <u>1</u> javac                 | 1. compile java programs |
| <u>7</u> Subclassing           | 2. bottom-up testing     |
| <u>6</u> Delegation            | 3. top-down testing      |
| <u>2</u> Driver                | 4. information hiding    |
| <u>3</u> Stub                  | 5. constant time look up |
| <u>4</u> Abstract data type    | 6. dynamic adaptation    |
| <u>9</u> Decrementing function | 7. static adaptation     |
| <u>5</u> Hashing               | 8. partial correctness   |
|                                | 9. total correctness     |

2 points for each correct answer.

No partial credit for giving 8 (partial correctness) for decrementing function.

8) Explain how having access to old versions of the source code can be useful in fixing recently discovered problems. Your answer should be no longer than five sentences. (10 points)

The fact that a problem was just discovered doesn't mean that it wasn't there before. If one knows when a bug was introduced, this greatly limits the amount of code that one must look at to try and find the bug. Often, one only has to look at the changes that were made just before the bug was introduced. One can run old versions of the program to try and discover when the problem shows up in the results of test, since this often corresponds to when the bug was introduced.

4 points for first sentence (problem may have been there before).

4 points for 2nd / 3rd sentences (knowing when bug was introduced => localizing bug to differences in code).

2 points for last sentence (Run old versions of code on new test suite).

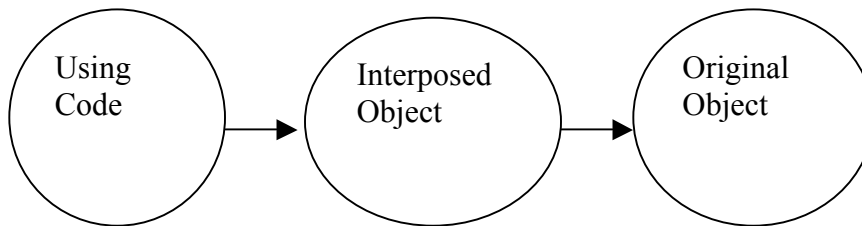
Additionally:

4 points for something else that is sensible (but incomplete)

-2 if you don't talk about running new tests on old code (cannot assume the test suite has never changed)

+2 for: We can revert to old code that used to work if we are at a loss. (capped at 10)

9) The *adapter* and the *proxy* design patterns each provide a kind of indirection by interposing an object between the original object and the using code:



In an *adapter* design pattern the interposed object has different behavior than the original object. In a *proxy* design pattern the interposed object has the same behavior as the original object. Assume that in the picture above:

The original object is an abstraction that returns all routes between two points.

The using code returns either one route between two points or all routes between two points, depending upon how it is invoked.

Describe in 2-4 sentences how you would use either an adapter or a proxy design pattern to achieve this. (10 points)

I would use the adapter design pattern to interpose an object with a method that returned all routes and a different method that returned one route. The using code would never call the methods of the original object, but would instead indirect through one of these two new methods.

5 points for each of the two sentences

- 10 Correct answer for how to use an adapter and AND/OR (1 is enough) proxy
- 7 "Fishing" (correct answer to adapter or proxy and incorrect answer to the other)
- 1 Don't mention how to separate a way of returning all paths vs. 1 path (for example, 2 methods, or pass in a parameter)
- 2 Talk about using a proxy but add a method (making it an adapter really)
- 2 Only return one of {1 path, all paths}
- 2 Describe a proxy correctly and thus not allow for 1 path to be returned
- 2 (total) Describe adapter and proxy correctly but talk about other problem
- 2 Talking about subclassing for no reason