Quiz 1 (February 21, 2014)

Your name:_________________________________________________________

Your Athena username:_______________________________________________

You have 50 minutes to complete this quiz. It contains 6 pages (including this page) for a total of 100 points.

The quiz is closed-book and closed-notes, but you are allowed one two-sided page of notes.

Please check your copy to make sure that it is complete before you start. Turn in all pages, together, when you finish. Before you begin, write your name on the top of every page.

Please write neatly. **No credit will be given if we cannot read what you write.**

For questions which require you to choose your answer(s) from a list, do so clearly and unambiguously by circling the number(s) or entire answer(s). Do not use check marks, underlines, or other annotations – they will not be graded.

Good luck!

<table>
<thead>
<tr>
<th>Problem</th>
<th>Points</th>
<th>Grade</th>
<th>Grader</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Short Answer</td>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2: Code Review</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3: Recursion</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4: Static Checking</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5: Testing</td>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 1 (Short Answer) (23 points).
Circle all correct answers for the following questions.

(a) When we implement a function recursively:

A. the postcondition of the function’s spec only restricts the base case of the recursion
B. we cannot use println debugging because recursive calls happen in arbitrary order
C. we cannot include tests that cause OutOfMemoryErrors if they violate the spec
D. changing the implementation to an iterative one will require us to change the spec
E. in Java, recursive calls will increase the depth of the call stack

(b) Which of the following are part of a function’s specification:

A. return type
B. restrictions on return value
C. number of arguments
D. argument types
E. restrictions on argument values

(e) Which of the following can be true about a pair of specifications $A$ and $B$:

A. $A$ can be stronger than $B$ and have a weaker precondition
B. $A$ can be stronger than $B$ and have the same precondition
C. $A$ can be stronger than $B$ and have a stronger precondition
D. $A$ can be stronger than $B$ and have an incomparable precondition
E. $A$ can be incomparable to $B$

(d) Which of the following are signs of an excellent specification:

A. the specification is declarative
B. the specification is operational
C. the implementation is allowed to ignore invalid arguments
D. the implementation is allowed to use different algorithms depending on the arguments
E. the specification utilizes the reader’s knowledge of the implementation

(e) In 6.005, your software should be:

A. easy to understand
B. ready for change
C. devoid of test cases
D. safe from bugs
E. documented in Klingon
Problem 2 (Code Review) (20 points).
Consider the following method:

```java
public static final String f(int a) {
    int MAJOR = 6;
    if (a == MAJOR) {
        return "OK";
    } else {
        return "." + f(a-1);
    }
}
```

This code was code-reviewed, producing the comments below. Circle AGREE or DISAGREE depending on whether the comment is correct or incorrect, and add your own one-sentence comment explaining your answer. The right explanation is worth more than the right circle.

(a) “Since the MAJOR variable is used only once in this method, you should get rid of it and just say a == 6 instead.”

AGREE DISAGREE

(b) “This method needs to state a precondition, because it doesn’t work for all inputs.”

AGREE DISAGREE

(c) “But adding a precondition would mean that if you call f(66), the function would check whether to throw an exception about 60 times, not just once.”

AGREE DISAGREE

(d) “This method is not safe from bugs because it uses global variables.”

AGREE DISAGREE
Problem 3 (Recursion) (19 points).
Consider the printSubsequences method below, implemented using a private helper.

Fill in the missing code to complete the recursive function. In doing so, the following String methods may be useful:

- int length() – returns the length of the String
- char charAt(int x) – returns the character at position x in the String
- String substring(int x) – returns a substring by omitting the first x characters

/** * Print all the subsequences of String s of length n, for n >= 0. * For example: printSubsequences("wxyz", 3) prints "wxy", "wxz", "wyz", "xyz" */
public static void printSubsequences(String s, int n) {
    findAndPrintSubseqs("", s, n);
}

private static void findAndPrintSubseqs(String subseq, String rest, int n) {
    // shift characters from rest into subseq while decreasing n
    if (n == 0) {
        // when n == 0, subseq has accumulated a subsequence of length n
        System.out.println(subseq);
        return;
    }

    if (___________) return;

    findAndPrintSubseqs(___________, ________________, __________);
    findAndPrintSubseqs(___________, ________________, __________);
}
Problem 4 (Static Checking) (12 points).
For each of the following pieces of code, determine the result of writing the code in a Java program.

- Circle OK if the code will compile and run without any exceptions.
- Circle Compile-time Error if the code will fail to compile.
- Circle Runtime Error if the code will compile but will throw an exception when run.

(a) `int[] arr = new int[] { 1, 2 };`  
   `arr[2] = 3;`
   OK    Compile-time Error    Runtime Error

(b) `int[] arr = new int[] { 1, 2 };`  
   `arr[0] = "2";`
   OK    Compile-time Error    Runtime Error

(c) `int[] arr = new int[] { 1, 2 };`  
   `arr[0] = 2;`
   OK    Compile-time Error    Runtime Error

(d) `String s = null;`  
    `System.out.println(s == null);`
   OK    Compile-time Error    Runtime Error

(e) `String s = null;`  
    `System.out.println(s.length());`
   OK    Compile-time Error    Runtime Error

(f) `String s = null;`  
    `System.out.println(s);`
   OK    Compile-time Error    Runtime Error
**Problem 5 (Testing) (26 points).**
Consider the following specification:

```java
/**
 * Reverses the end of a string.
 * 012345 012345
 * For example: reverseEnd("Hello, world", 5) returns "Hello
dlrow ,"
 * <-----> <----->
 * With start == 0, reverses the entire text.
 * With start == text.length(), reverses nothing.
 * @param text non-null String that will have its end reversed
 * @param start the index at which the remainder of the input is reversed,
 * requires 0 <= start <= text.length()
 * @return input with the substring from start to the end of the string reversed
 */
public static String reverseEnd(String text, int start)
```

(a) Which of the following inputs should we include in our test suite? **Circle all the valid test cases.**
1. null, 0
2. "abba", 1
3. "abracadabra", 12
4. ",", 0
5. "a", 0

(b) We want to test `reverseEnd` systematically. We will partition the spaces of the inputs, then select test inputs from the Cartesian product of these partitions.

Fill in **3 partitions for text** down the side of the table below.

Fill in **4 partitions for start** across the top. You may define partitions for `start` in terms of `text.length()`.

Then, choose test inputs, placing them at the appropriate intersection of partitions.

- If a box can be correctly filled with a valid test case from part (a) above, you must use that test.
- You may also devise additional test cases. **In those tests, use only the letters w, x, y, and z in text.**
- You may write the same test inputs in more than one box if appropriate.
- You may leave boxes blank if appropriate.

<table>
<thead>
<tr>
<th>partitions for text</th>
<th>partitions for start</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>