CSE 15L
Midterm
Summer 2011

Page 1 ___________ (17 points)
Page 2 ___________ (6 points)
Page 3 ___________ (11 points)
Page 4 ___________ (8 points)
Page 5 ___________ (15 points)
Page 6 ___________ (12 points)
Total ___________ (69 points = 65 base points + 4 points EC [>5%])

(65 points = 100%)

This exam is to be taken by yourself with closed books, closed notes, no electronic devices.
You are allowed one side of an 8.5"x11" sheet of paper handwritten by you.
Write a for loop statement to complete the code below. The body of the loop is given, just write the for loop statement.

```java
int[] intArray = new int[readInt()]; // method readInt() reads an int from somewhere
// Write the for loop statement here to index each element of the array

//
{
    System.out.println( intArray[i] ); // Prints each element in the array at index i
}
```

If you were to get an ArrayIndexOutOfBoundsException in the above code, what common type of programming logic error does this usually indicate?

Name a type of software defect that will NOT show up as incorrect or unexpected results/output. No code.

Assume variable str is properly defined as a String. Why might one prefer to code a string comparison as

```java
if ("Match".equals(str)) versus if (str.equals("Match"))
```

Which combination of coupling and cohesion is a good "best practices" software engineers should strive for which will ultimately help with debugging?

- A) low coupling and low cohesion
- B) low coupling and high cohesion
- C) high coupling and low cohesion
- D) high coupling and high cohesion

What is a common type of defect/bug that can seem to disappear (defect does not occur) when a program is run in a debugger or debugging mode?

Identify when various errors can occur in a compiled language like Java:

- missing semicolon  [A) Compile time]
- cannot find symbol  [B) Run time]
- NullPointerException
- stack backtrace displayed
- variable might not have been initialized
- infinite loop
You are debugging a Java program with a loop and code like this:

```
16: int x = readInt(); // Read an int from somewhere (not important)
17: x = Math.abs(x);  // If x is negative, make it positive
18: while (x > 0);
19: {
20:   System.out.println(x);
21:   x--;
22: }
23: System.out.println("DONE");
```

Note the semicolon at the end of the `while` statement on line 21. When you compile and run this program, what two details would you observe with the output that would make you focus your debugging eye on a possible stray semicolon at the end of this `while` statement? For this part, assume `x` does not have the value 0.

1)
2)

Say you did not notice the semicolon at the end of the `while` statement. Based on the observed output you described above, at which line number would be the best place to add a logging/println statement to help debug this piece of code?

What output would you expect if `readInt()` returned the `int` value 0?

Suppose you had some code like the following:

```
final int MAX = 100; // This particular value is not important
int[] intArray = new int[MAX];
int a, b;

a = Math.abs(...); // Some positive value read from somewhere
b = Math.abs(...); // Some positive value read from somewhere

if ((a + b) < MAX)
   System.out.println(intArray[a + b]);
```

When you run this code you get an `ArrayIndexOutOfBoundsException`. Explain how this could happen.
In Java, what type must `var1` and `var2` be for the following piece of code to compile?

```java
if ( var1 = var2 )
    doSomething();
```

A) the absence of bugs  
B) how the program is supposed to work  
C) the presence of bugs  
D) proof that the program is correct  
E) all of the above

Program testing can be used to show ____

You should always put a space around all binary operators. Give an example and explain where it does make a difference whether you use a space around a binary operator or not.

You are given the task of testing a method in Java. The method takes a String as its only argument. Before even looking at what this method is supposed to do with the String being passed in, what two argument values should be used in your test suit to check a couple corner cases? Assume the name of the method is `foo`:

```java
foo( __________ )   foo( __________ )
```

What is a Heisenberg? ____

A) buggy print statements printing the wrong thing  
B) reassembling more than necessary will need to be disassembled again  
C) repeatedly split up the search space into a good half and a bad half  
D) test instrumentation affects the system under test  
E) all other things being equal, the simplest explanation is the best

Logging is ______ (off/on) by default.

Assertions are ______ (off/on) by default.

Assertions should not be used for ____

A) checking precondition of private methods/ctors  
B) checking precondition of public methods/ctors  
C) checking postcondition of private methods/ctors  
D) checking postcondition of public methods/ctors  
E) checking class invariants  
F) they should be used for any and all of the above

What should you use/do instead of using assertions for this particular instance?

The structure of an assert statement is

```java
assert Expression1 : Expression2 ;
```

Give/state an example of what cannot be used in `Expression2`. 
A directory has 3 Java source files: A.java, B.java, C.java

$ ls
A.java  B.java  C.java

class A defines an instance variable of type B.
class B defines an instance variable of type C.
class C defines an instance variable of type String.

The user issues a Java compile command with only one command line argument and then another `ls`.

$ javac ____________

$ ls
A.java  B.class  B.java  C.class  C.java

What single command line argument must the user have used to get the files listed in the second `ls`? Fill in the line after `javac` above.

In general, why is it usually better to put a special case check (like the special case of checking if the first char started a group of repeated chars in `countRepeatedCharacters()` in Lab 5 WordAnalyzer) above a looping construct rather than building additional logic in the body of the loop?

In Lab 4 Uniq, we had a test run that looked like:

$ java Uniq xxx > /dev/null

What is `/dev/null` known as? ______________________________

Assuming there is no file named `xxx` in the current directory, why would we want this test run as opposed to the same but without the `> /dev/null` redirection?

What would you type in at the command line immediately after you run the above test run to check the exit status of the program? The $ is the shell prompt.

$ ________________

In computing, the exclamation mark (!) is commonly known as __________

Given the test run from Lab 3 Sort

$ java Sort -r - - > xxx < yyy

Where is input coming from? ______
Where is output going? ______

A) keyboard/stdin
B) terminal window/stdout
C) file xxx
D) file yyy
In Lab 1 HTML Debugging, explain the problem/bug with the hypertext link in the code

<p>This butterfly, <u>Karner's Blue</u>, named by Nabokov, <a href="www.libraries.psu.edu/nabokov/endan2.htm" title="Nabokov's Endangered Blues">More here</a>.</p>

You type in java Bar2 at the command/shell prompt and you get the following:

```
Exception in thread "main" java.io.FileNotFoundException: line23 (No such file or directory)
  at java.io.FileInputStream.open(Native Method)
  at java.io.FileInputStream.<init>(FileInputStream.java:135)
  at java.io.FileInputStream.<init>(FileInputStream.java:95)
  at java.io.FileReader.<init>(FileReader.java:64)
  at Helper.readFile(Foo5.java:74)
  at Bar2.main(Bar2.java:32)
```

What is the name of the file this program tried to read from? _____________________

What method in what class in what file and line number in your code did this occur?

Method _______________________________
Class _______________________________
File ________________________________
Line # ______________________________

What do all the <init> symbols indicate?

What does (Native Method) tell us about the java.io.FileStream.open method?

In the Java run time environment, where are the following stored

_____ objects
_____ static variables
_____ formal parameters
_____ instance variables
_____ method/constructor code
_____ local variables

You execute the ls command at the Linux bash shell prompt with several command line arguments. Without using the arrow keys to edit previous commands and without typing in all the same command line arguments again, what do you type in at the shell prompt to execute the rm command with all the same command line arguments as the previous command. The $ is the shell prompt.

$ ______________________
Change one thing at a time. This is part of the scientific method.

Indeed, Agans states use a ____________________, not a ____________________. We want to avoid using the latter approach to debugging (making lots of [relatively undirected] changes hoping that one of them fixes the bug).

In Lab5 WordAnalyzer, what was the class invariant condition (the condition in the hasValidState() method)?

In the Agans book, what does he mean by grab the brass bar with both hands?

When keeping your interactive debug log in the labs (or any time you are debugging), why is it a bad idea to write down only the line number of the code you changed vs. the writing down in your log book the code/expression on that line and what you changed.

What type of testing allows you to perform code coverage testing - white box or black box testing? __________

What boundary-value analysis / corner case values should you use to test a program that does one thing if the input value is less than 25, another thing if the input value is greater than 25, and does nothing if the input value is equal to 25?

What is the name of the type of testing that attempts to uncover new errors/bugs in previously working functionality/code after a change has been made to the system?

What is the name of the process of improving the design of existing code (to make it clearer or better structured which often makes it easier to spot and fix a bug) without changing its behavior?

From Lab 5 WordAnalyzer, how do you fix the bug in the constructor below without changing the name of the formal parameter (word)?

```java
public WordAnalyzer(String word)  // Cannot change anything on this line!
{
    word = word;
}
```

Write the corrected line of the body of the constructor in the space below here.