

NAME: \_\_\_\_\_

NETID: \_\_\_\_\_

CS2110 Spring 2015 Final

Write your name and Cornell NetID. There are 8 questions on 9 numbered pages. Check now that you have all the pages. Write your answers in the boxes provided. Use the back of the pages for workspace. Ambiguous answers will be considered incorrect. The exam is closed book and closed notes. Do not begin until instructed. You have 2 ½ hours.

Use the backs of the pages if you need more room to write. Be sure to make it clear where an answer is continued.

**Note:** We suggest that you read through the whole test, writing answers to questions that need little thinking and that you can answer quickly and correctly. As you do so, get a sense of the other questions, which require more thought and care. This will allow you to budget your time wisely.

Good luck!

Have a nice summer break!

	1	2	3	4	5	6	7	8	Total
Score	/20	/8	/12	/10	/10	/14	/12	/14	
Grader									

1. True-false questions
2. Asymptotic complexity
3. Data structures
4. Concurrency
5. Recursion
6. Graphs
7. Object-oriented programming
8. Loops and algorithms

**1. (20 points) True or false questions**

a	T	F	A public, non-nested class C must be defined in a file called C.java.
b	T	F	It is ok for an interface to declare two methods with the signature <code>void foo(int n)</code> .
c	T	F	Assume B is a subclass of A and C a subclass of B. It is impossible for any call on a method in an object of class C to execute code in a constructor of A.
d	T	F	A type is a set of values
e	T	F	Consider a variable v declared as <code>C v;</code> . A call <code>v.m(args)</code> is legal as long as every subclass of C declares method m.
f	T	F	Mergesort is a worst-case $O(n \log n)$ algorithm.
g	T	F	Quicksort is a worst-case $O(n \log n)$ algorithm.
h	T	F	Dijkstra's shortest path algorithm is a worst-case $O(n \log n)$ algorithm (where n is the number of vertices of the graph)
i	T	F	The proof technique of mathematical induction cannot be used if one explicitly proves more than one base case.
j	T	F	Suppose class B is a subclass of class C. Then <code>ArrayList&lt;B&gt;</code> is a subclass (or subtype) of <code>ArrayList&lt;C&gt;</code> .
k	T	F	Assume <code>foo(n)</code> is a recursive method whose body includes a call to <code>foo(n+1)</code> . Then the call <code>foo(100)</code> will induce infinite recursion and never terminate.
l	T	F	Suppose String objects b and c contain the same sequence of characters. Then <code>b == c</code> is true.
m	T	F	Assume b is a public field of object c. If two methods <code>foo()</code> and <code>bar()</code> , in completely different classes, each have a block declared as <code>"synchronized(c.b) { ... }"</code> , then these two blocks will be synchronized with each other.
n	T	F	A Java Swing component can have at most one listener for a particular type of event (e.g. <code>MouseEvent</code> ).
o	T	F	To make several instances of a class execute concurrently, it is sufficient to implement interface <code>Runnable</code> .
p	T	F	If one makes a set of Integers synchronized by creating it with the command <code>"Set&lt;Integer&gt; s = Collections.synchronizedSet(new HashSet&lt;Integer&gt;());"</code> , then all subsequent operations on it are threadsafe.
q	T	F	Two threads that do not access shared data or resources can never enter into a race condition.
r	T	F	A minheap of n elements requires $O(n)$ space.
s	T	F	A class cannot access a private field of an inner class directly.
t	T	F	If a program throws a <code>NullPointerException</code> , the first debugging step is usually to identify the location of the error from the stack trace.

**2. (8 points) Asymptotic complexity**

(a) (3 pts) Two algorithms A and B have running times given by the functions  $f(n)$  and  $g(n)$  respectively. Assume  $f(n)$  is  $O(g(n))$  and  $g(n)$  is  $O(f(n))$ . In this case, could there be a reason to choose one algorithm over another in a practical setting? Answer YES or NO and give a reason for your answer.

(b) (3 pts) Give the asymptotic complexity in terms of  $n$  (e.g  $O(n^3)$ ) and explain your answer. You should give the *tightest possible* big-Oh bound.

```
int sum= 0;
for (int i= 1; i <= n; i= i + 1) {
    for (int j= 1; j <= Math.min(1000000, i); j= j + 1) {
        sum= sum + 10;
    }
}
sum= sum + 9999;
```

(c) (2 pts) You have an  $O(n^2)$  algorithm for analyzing the interaction of  $n$  molecules. The algorithm loops over every pair of molecules and computes the force between them. On your laptop, the code runs extremely slowly even for  $n = 10$ . But by a sudden windfall, your research is allocated computing resources at a supercomputing facility, and you get to run your code on a 500,000-core monster machine. You split the set of pairs evenly across the cores so that each core processes only  $1/500,000$  of the set. Suddenly, you are able to simulate a system with  $n = 1000$  molecules in real time. How has the asymptotic (big-Oh) complexity of your algorithm changed in terms of  $n$ ? Explain briefly.

**3. (12 points) Data structures**

(a) (4 points) Describe two characteristics of a good hash function and explain why they are necessary.

(b) (2 points) Is this a good hash function? Why or why not?

```
public static int hash(int n) {
    return n % 2;
}
```

(c) (6 points) Consider a doubly linked list consisting of Node objects. Class Node has the following structure:

```
public class Node {
    public Node prev; // points to previous node in list
    public Node next; // points to next node in list
}
```

Complete the following (non-recursive) function:

```
/** Return the middle Node in a list from head to tail inclusive.
 * If the list has an even number of nodes, return the first
 * Node of the middle pair. Note: head and tail are not null. */
public static Node middle(Node head, Node tail) {

    while ( _____ ) {

        _____

        _____

    }

    return _____ ;
}
```

**4. (10 points) Concurrency**

You have written a multithreaded GUI library. Each object in the GUI monitors its events (such as mouse clicks) on a separate thread. In other words, there is one thread per class instance.

Your colleague decides to use your library to create a GUI program. She creates a central object of class GUI (there is only one such object in the program), with the code shown below. Her program has the following logic: there are several Buttons in the interface. When a Button is clicked, it calls the GUI's function *actionPerformed*, in the Button's thread. The GUI then looks up the address of a webpage associated with the Button in a HashMap and downloads the page content (HTML, images, videos, etc.). This download may take several seconds.

To ensure that there are no synchronization errors when several Button threads are simultaneously accessing the HashMap, your colleague synchronizes function *actionPerformed* as shown below.

```
public class GUI {
    private HashMap<Object, String> lookupTable= new HashMap();
    // Constructor etc. goes here ...

    /** Handle a button click. */
    public synchronized void actionPerformed(ActionEvent e) {
        // Look up the button's web address in the HashMap
        String address= lookupTable.get(e.getSource());

        // Fetch the webpage content (this may take a while...)
        downloadWebpage(address);
    }

    /** Download the webpage at address, if it is not null.
        Return only when the download is complete. */
    private void downloadWebpage(String address) {
        ...
    }
}
```

(a) (4 points) Is the synchronization **correct**? (That is, will it fully avoid race conditions?) Answer YES or NO. If it is not correct, show how to correct it. (You can indicate your changes inline in the code listing above, without rewriting the whole thing.)

(b) (6 points) Is the synchronization **efficient**? Why or why not? Explain briefly, but fully. If your answer is NO, then rewrite the function so that it is both correct and efficient. (You can indicate your changes inline in the code listing above, without rewriting the whole thing.)

**5. (10 points) Recursion**

(a) (3 points). Below, write the steps in executing a procedure call, for example,  $m(5, 2+3)$ ;

(b) (7 points) The University of Oz assigns each student a unique randomly chosen integer as a student ID, storing the already assigned IDs in a binary search tree (BST). Each node of the BST is an instance of `TreeNode`:

```
public class TreeNode {
    public TreeNode left;    // left child
    public TreeNode right;  // right child
    public int value;       // value (ID) stored at this node
}
```

When a new student is admitted, the university picks a random integer and checks if it is already in the BST. If it is not, it is set as the student's ID. If it is, the process is repeated with a new random integer.

Unfortunately, the Wicked Witch of the West has reportedly hacked the BST and changed one or more values. Fearing the worst, the university hires you to check the BST. As a first step, you decide to verify that the BST is correctly structured, i.e. no values are out of order. Below, write the recursive Java function to achieve this. The function is initially called on the BST with root node `root` as

```
misplaced(root, Integer.MIN_VALUE, Integer.MAX_VALUE)
```

For simplicity, assume no value in the tree is exactly `Integer.MIN_VALUE` or `Integer.MAX_VALUE`.

```
/** If the binary tree rooted at node t is a valid BST and its
    values are in the range m..n, return null.
    Otherwise, return a value that is out of order, i.e. violates
    the BST property. */
public static Integer misplaced(TreeNode t, int m, int n) {
```

```
}
```

**6. (14 points) Graphs**

(a) (2 points) If all edge weights are equal, to what algorithm is Dijkstra's shortest paths algorithm equivalent?

(b) (2 points) What property must a directed graph satisfy in order for topological sort to work on it?

(c) (2 points) Kruskal's algorithm for constructing a spanning tree constructs a set of edges  $E$  of the spanning tree. Starting with  $E$  empty, at each iteration, it adds to  $E$  an edge with minimum weight that does not create a cycle among the edges in  $E$ . What does Prim's algorithm do at each iteration?

(d) (2 points) Look at the specification of procedure `dfs` below, in part (e). Explain what is meant by "a node is REACHABLE from  $u$ ", as discussed in lecture.

(e) (6 points) Write the body of the following algorithm. You may make it recursive or iterative, but recursive is easier.

```
/** Node u is unvisited. Visit all nodes REACHABLE from u. */  
public static void dfs(Node u) {
```

```
}
```

**7. (12 points) Object-oriented programming**

(a) (3 points) Write down the three steps in evaluating a new-expression `new C(args)`

Peter, Paul, and Mary form a group to write a graphics program that processes shapes. Their code includes the following two classes:

```
/** An instance is a rectangular shape */
public class Rectangle {
    protected int width;
    protected int height;

    public Rectangle(int w, int h) {
        width= w; height= h;
    }

    public int getWidth() { return width; }
    public void setWidth(int w) { width= w; }
    public int getHeight() { return height; }
    public void setHeight(int h) { height= h; }
}

/** An instance is a square shape. */
public class Square extends Rectangle {
    public Square(int s) { super(s, s); }

    public void setSize(int s) {
        width= s; height= s;
    }
}
```

(b) (4 points) What problem(s) do you anticipate with the above design? Your answer should *not* talk about the structure of the classes and how they should be different. Instead, your answer should talk about possible behaviors —what a user could do that might cause errors, that might destroy what is expected.

(c) (5 points) Rewrite the classes, adding any other code (including new classes or interfaces if necessary) to improve the design and prevent the problem(s) of part (a). There is no single answer here. There are several ways to overcome the possible behavioral problems. Choose one.



**8. Loops and algorithms**

(a) (4 points) Write the loop invariants for insertion sort and selection sort to sort an array  $b$ .

The rest of question 8 concerns the Fibonacci numbers 0, 1, 1, 2, 3, 5, 8, 13, 21 ..., which are defined by:

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2} \quad \text{for } n > 1 \end{aligned}$$

Here is a function to calculate  $F_n$ :

```
/** Precondition: n > 0. Return Fibonacci number F_n */
public static int fib(int n) {
    int b= 0, c= 1;

    // invariant: (you write this below)
    for (int k= 1; k < n; k= k + 1) {
        c= b + c;
        b= c - b;
    }
    return c;
}
```

(b) (4 points) Write a brief, meaningful, and complete invariant for the for-loop, in terms of  $b$ ,  $c$ , and  $k$ .

(c) (6 points) We point out that mathematical induction is actually used to prove that the invariant is maintained. Consider the theorem:

*Theorem:*  $P(k)$  holds for  $k \geq 1$ , where  $P(k)$  is: after  $k-1$  iterations, the loop invariant holds

Prove this theorem by induction over the positive integers.