Name **SOLUTION** ID _____ Final Score _____
Extra Credit _____

Section (circle one): MW 8:30-9:50 TTh 9:30-10:50 TTh 11:00-12:20

# CSCI 201L Final – Written
# Spring 2017
# 12% of course grade

1. **Servlets** – Explain the difference between a GET and POST form submission. Give one reason why a programmer would choose to use a GET and one reason why a programmer would choose to use a POST. **(0.5% + 0.5% + 0.5%)**

   Difference
   **A GET will submit the data along the URL.**
   **(http://localhost/index.html?user=root&pass=yep)**

   **A POST will submit the data in the body of the request.**

   Why programmer would choose to use GET
   **Allow bookmarking of a page.**
   **Able to be read in JavaScript (client-side).**
   **Allows user to change the parameters in the URL.**

   Why programmer would choose to use POST
   **Hide the data from being seen by the user.**
   **Not able to be accessed on the client-side.**
   **If uploading a file, POST must be used.**

2. **Software Engineering –** The Mythical Man-Month by Fred Brooks claims that it takes 9 times as long to create a software product system compared to just creating a program. Give two reasons why it takes significantly longer to create a software product system. **(0.5% + 0.5%)**

   1. **Significant documentation is required for programmers.**
   2. **Documentation may be required for users.**
   3. **Extensive testing needed.**
   4. **More detailed requirements necessary.**
   5. **Better design of system.**
   6. **May need to meet government regulations.**
   7. ***Other answers may be correct.***

3. **AJAX and Web Sockets** – AJAX and Web Sockets both allow a web page to be updated without needing to refresh the page. Explain what AJAX is and what Web Sockets are. Why would a programmer choose to use one over the other? **(0.5% + 0.5% + 0.5% + 0.5%)**

Define AJAX

**Asynchronous JavaScript and XML (AJAX) allows calls to the server to take place asynchronously, which will then allow part of an HTML page to be updated without needing to refresh the page.**

Define Web Sockets

**Web Sockets allows persistent communication between the browser and the server. The communication can be initiated by the server, different from other HTTP communication.**

Why programmer would choose AJAX

**If only part of the page needs to be updated based on the user performing some action, AJAX would be used.**

Why programmer would choose Web Sockets

**If only part of the page needs to be updated based on the server initiating the action, Web Sockets would be used.**

4. **Concurrent Programming -** What is the primary goal of each of the three types of concurrent programming? **(0.5% + 0.5% + 0.5%)**

Multi-threading

**Add functionality since two threads are executing at what appears to be simultaneously to the user.**

Parallel

**Make the program execute faster because more than threads are actually executing simultaneously in different cores in a shared memory space.**

Distributed

**Make the program execute faster because threads are executing in different cores in different memory spaces, typically on different computers.**

5. **Networking Theory** – Assume you are given the following IP address and subnet mask.

```
IP –          120.192.168.50
IP –          0111 1000  1100 0000  1010 1000  0011 0010
Subnet Mask – 255.255.248.0
Subnet Mask – 1111 1111  1111 1111  1111 1000  0000 0000
```

**a.** What class IP address is given? What is the network address? **(0.5%)**
**Class A**
**120.0.0.0**

**b.** Is the IP address public or private? **(0.5%)**
**public**

**c.** How many hosts can be on the subnetwork? **(0.5%)**
**2046 ($2^{11}$ – 2 for the network and broadcast addresses)**

**d.** What is the network and subnetwork combination? **(0.5%)**
**120.192.168.0**

6. **Synchronization** – Explain how the following code can throw a `ConcurrentModificationException`. **(0.5%)**

```
1  class Question6 extends Thread {
2    private static Set<Integer> hashSet =
3            Collections.synchronizedSet(new HashSet<Integer>());
4    private static Lock lock = new ReentrantLock();
5    public Question6(int num) {
6      hashSet.add(num);
7    }
8    public void run() {
9      lock.lock();
10     try {
11       Iterator<Integer> iterator = hashSet.iterator();
12       while (iterator.hasNext()) {
13         System.out.print(iterator.next() + " ");
14       }
15     } finally {
16       lock.unlock();
17     }
18   }
19 }
```

**While iterating through the hashSet on lines 12-14, if another thread gets to line 6, a ConcurrentModificationException will be thrown. Iterating in Java is fail-fast, meaning that you cannot modify a data structure while another thread is iterating over it.**

7. **Locks –** In the following code, assume a thread calls the method `foo()`. Answer the questions that follow the code. **(0.25% + 0.25% + 0.25% + 0.25%)**

```
1  import java.util.concurrent.locks.Lock;
2  import java.util.concurrent.locks.ReentrantLock;
3  public class Question7 {
4      private Lock lock = new ReentrantLock();
5      public void foo() {
6              try {
7                      lock.lock();
8                      System.out.println("foo 1");
9                      bar();
10                     System.out.println("foo 2");
11             } finally {
12                     lock.unlock();
13             }
14     }
15
16     public void bar() {
17             try {
18                     lock.lock();
19                     System.out.println("bar");
20             } finally {
21                     lock.unlock();
22             }
23     }
24
25     public static void main(String [] args) {
26             Question7 q7 = new Question7();
27             q7.foo();
28     }
29 }
```

**a.** Does `foo 1` get printed? ☐ Yes ☐ No

**b.** Does `bar` get printed? ☐ Yes ☐ No

**c.** Does `foo 2` get printed? ☐ Yes ☐ No

d. If yes, explain why `foo 2` is printed. If not, what happens?

**The lock being used on line 4 is a ReentrantLock, meaning that you do not have to get the lock again to get into another critical section of code managed by that lock. That means that getting the lock on line 18 and releasing the lock on line 21 will not occur if the method bar() is called from within the method foo().**

**8. Multi-Threaded Programming –** Look at the following code then answer the questions below. **(0.5% + 0.5%)**

```
1  import java.util.concurrent.locks.Lock;
2  import java.util.concurrent.locks.ReentrantLock;
3
4  public class Question8 {
5      private Lock lock = new ReentrantLock();
6      public synchronized void foo() {
7              try {
8                      lock.lock();
9                      System.out.println("foo");
10             } finally {
11                     lock.unlock();
12             }
13     }
14     public void bar() {
15             try {
16                     lock.lock();
17                     synchronized(this) {
18                             System.out.println("bar");
19                     }
20             } finally {
21                     lock.unlock();
22             }
23     }
24 }
```

**a.** Is there a chance of deadlock? ☒ Yes ☐ No

b. If yes, explain how? If no, what are all the possible outputs?

**Deadlock can occur if one thread calls foo and gets the lock on the instance of Question8, then it gets switched out of the CPU.**
**Another thread comes in and gets the lock declared in line 5 on the same instance of Question8.**
**The second thread will not be able to get into the synchronized block of code on line 17, and the first thread will not be able to get the lock on line 8.**
**This causes circular waiting, which will be deadlock.**

**9. Conditions –** There are two ways to wake up a thread that is waiting on a condition – `signal()` and `signalAll()`. Since we don't have control over the order threads are switched into the CPU from the Ready state, why would we ever call `signal()` instead of `signalAll()`? **(0.5%)**

**The signal() method can be used to wake up just one thread that is waiting on a condition. We could make it wake up the thread that has been waiting on the condition the longest if the fairness policy is set to true.**
**With the fairness policy set to false, a random thread will be woken up, which will be the same as calling signalAll() and letting the JVM decide which thread to put into the CPU next.**

**10. Databases and SQL** – Answer the following questions concerning the database below.

Here is the `Book` table.

| bookID | title | author | isbn | numCopies |
|---|---|---|---|---|
| 1 | Tonight on the Titanic | Mary Pope Osborne | 978-0-606-16894-6 | 3 |
| 2 | Afternoon on the Amazon | Mary Pope Osborne | 978-0-679-86372-9 | 1 |
| 3 | Balto of the Blue Dawn | Mary Pope Osborne | 978-0-553-51085-0 | 2 |
| 4 | Happy Birthday, Bad Kitty | Nick Bruel | 978-0-545-29863-6 | 2 |
| 5 | Bad Kitty Does Not Like Candy | Nick Bruel | 978-1-62672-230-9 | 1 |
| 6 | Bad Kitty Drawn to Trouble | Nick Bruel | 978-1-62672-117-3 | 2 |

Here is the `User` table.

| userID | username |
|---|---|
| 1 | jimmy |
| 2 | joannie |
| 3 | johnny |
| 4 | jenny |

Here is the `CheckedOut` table.

| checkedOutID | bookID | userID | numCheckedOut |
|---|---|---|---|
| 1 | 3 | 4 | 1 |
| 2 | 3 | 3 | 1 |
| 3 | 1 | 1 | 2 |
| 4 | 2 | 2 | 1 |

**a.** Write the SQL code to represent `jenny` checking out the book Happy Birthday, Bad Kitty. **(0.5%)**

```
INSERT INTO CheckedOut (bookID, userID, numCheckedOut)
      VALUES (4, 4, 1);
```

**OR**

```
INSERT INTO CheckedOut (bookID, userID, numCheckedOut)
      VALUES (
SELECT bookID FROM Book WHERE title='Happy Birthday, Bad Kitty',
SELECT userID FROM User WHERE username='jenny',
1);
```

**b.** Draw the table that is returned from the following query **after** the SQL statement from part a has executed. **(0.5%)**

```
SELECT b.title, co.numCheckedOut
  FROM Book b, CheckedOut co
  WHERE b.bookID=co.bookID;
```

| **title** | **numCheckedOut** |
|---|---|
| Balto of the Blue Dawn | 1 |
| Balto of the Blue Dawn | 1 |
| Tonight on the Titanic | 2 |
| Afternoon on the Amazon | 1 |
| Happy Birthday, Bad Kitty | 1 |

**Extra Credit Questions**

*Extra credit is applied after the curve so does not affect other students.*

☐ **Extra Credit –** For the final project, we required weekly meetings with your assigned CP. **(0.25%)**

Did you think these meetings were beneficial? ☐ Yes ☐ No

Please explain.

**105 (81.4%) – yes**
**24 (18.6%) – no**

☐ **Extra Credit –** This is the first semester we have included web development in the course. **(0.25%)**

Do you think this was a positive change? ☐ Yes ☐ No

How do you think we can improve on this change?

**125 (96.9%) – yes**
**4 (3.1%) – no**