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

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

CSCI 201L Written Exam #2
Fall 2017
15% of course grade

1. **Inheritance** – When Java was developed, it was scrutinized since it only has single inheritance. Give two responses the creators of Java gave to the scrutiny. **(0.5% + 0.5%)**

   Response #1
   Multiple inheritance is not commonly used in other languages.

   Response #2
   Java has interfaces to support something similar to multiple inheritance.

2. **Serialization –** As you may have seen throughout the course, serialization provides an easier way to save data because the parsing is already done for you (among other reasons). With networking, serialization is not always the best choice though. Give one reason why a programmer would not use serialization in a networking application. **(0.5%)**

   If the two programs communicating with each other are not both written in Java, serialization won't work.

   Also, if the protocol is already defined as something other than serialization, serialization won't work.

3. **JDBC –** SQL injection is a type of attack where a hacker tries to pass SQL code into an input field with the hopes that the program takes the value of the field and passes it directly into a SQL statement. Give two ways JDBC programmers can avoid SQL injection attacks. **(0.5% + 0.5%)**

Reason #1
Parse the submitted text and escape all SQL control characters

Reason #2
Use prepared statements

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 to a program that couldn't otherwise have happened with single-threaded code. Specifically, two things happening at what appears to be the same time.

Parallel
Make the program faster by taking advantage of redundant hardware on the computer.

Distributed
Make the program faster by taking advantage of redundant hardware, typically different computers.

5. **Networking –** Multi-threading is often used with networking. However, if the main thread services the request directly after accepting the connection, we wouldn't need to use multi-threading. What would be the implication or negative consequences of this? In other words, explain the differences with the following two snippets of code. **(1.0%)**

```
Single-Threaded
while(true) {
  accept connection
  process request
}
```

```
Multi-Threaded
while(true) {
      accept connection
      spawn new thread to process request
}
```

With the single-threaded code above, we would not accept subsequent connections until we have finished processing the previous request. If that process takes a significant amount of time, no other connections would be allowed until the process finishes. We might have some connection timeouts occur.

With the multi-threaded code above, we will continue accepting new connections constantly and processing the requests in separate threads.

6. **Garbage Collection** – The garbage collector in Java helps with memory management. Java provides a method called `System.gc()` that programmers can invoke.
   a. Explain what the `System.gc()` method does. **(0.5%)**
   This method increases the priority of the garbage collector thread.

   b. Why does `System.gc()` not cause the garbage collector to be run immediately? **(0.5%)**
   The garbage collector is a thread, and threads do not preempt other threads. With the priority increased, the garbage collector will have a higher probability of being switched into the CPU during the next context switch.

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

```
IP –          213.87.146.27
IP –          1101 0101  0101 0111  1001 0010  0001 1011
Subnet Mask – 255.255.255.240
Subnet Mask – 1111 1111  1111 1111  1111 1111  1111 0000
```

a. What class IP address is given? What is the network address? **(0.25% + 0.25%)**
Class C
213.87.146.0

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

c. How many hosts can be on the network? **(0.5%)**
$2^8 - 2 = 254$ (the host with all 0s is the network address, and the host with all 1s is the broadcast address)

If 256 is the answer, give 0.25%.

d. How many hosts can be on the subnetwork? **(0.5%)**
$2^4 - 2 = 14$

If 16 is the answer, give 0.25%

e. What is the network and subnetwork combination? **(0.25% + 0.25%)**
Binary
1101 0101 0101 0111 1001 0010 0001 0000

Decimal
213.87.146.16

f. What are the first and last IP addresses that could be assigned to hosts in the subnetwork? **(0.25% + 0.25%)**
First Assignable IP Address (in decimal)
213.87.146.17

Last Assignable IP Address (in decimal)
213.87.146.30

8. Locks – Look at the following code, then answer the questions that follow the code.

```
1  import java.util.concurrent.locks.Lock;
2  import java.util.concurrent.locks.ReentrantLock;
3  public class Question8 {
4      private Lock lock = new ReentrantLock();
5      public void foo() {
6             try {
7                    lock.lock();
8                    System.out.println("foo");
9             } finally {
10                   lock.unlock();
11            }
12     }
13
14     public void bar() {
15            try {
16                   lock.lock();
17                   System.out.println("bar 1");
18                   foo();
19                   System.out.println("bar 2");
20            } finally {
21                   lock.unlock();
22            }
23     }
24
25     public static void main(String [] args) {
26            Question8 q8 = new Question8();
27            q8.bar();
28     }
29 }
```

a. The type of lock we used in Java was called a `ReentrantLock`. Explain what a `ReentrantLock` is and how it is different from a lock that is not reentrant. **(0.5%)**
With a reentrant lock, a thread will check to see if it already has the lock needed to enter a critical section. If it does, it will not try to reacquire the lock.

b. What is the output of the above code? **(0.5%)**
bar 1
foo
bar 2

c. If the variable `lock` was not reentrant, what would the output of the code be? **(0.5%)**
We would see "bar 1", but then the code would hang, waiting on the thread to give up the lock, but the current thread has the lock.

9. Locks and Monitors – Look at the following code then answer the questions below.

```
1  import java.util.concurrent.locks.Lock;
2  import java.util.concurrent.locks.ReentrantLock;
3
4  public class Question9 extends Thread {
5      private static Lock lock = new ReentrantLock();
6      public void bar() {
7              synchronized(Question9.class) {
8                      System.out.println("bar 1");
9                      System.out.println("bar 2");
10             }
11     }
12     public void run() {
13             try {
14                     lock.lock();
15                     System.out.println("foo");
16                     bar();
17             } finally {
18                     lock.unlock();
19             }
20     }
21     public static void main(String [] args) {
22             for (int i=0; i < 100; i++) {
23                     Question9 q9 = new Question9();
24                     q9.start();
25             }
26     }
27 }
```

a. Give two rules concerning the output of the above program. In other words, provide two statements that will always be true about the output. **(0.5% + 0.5%)**

Rule #1

"foo" will always be followed by "bar 1" and
"bar 1" will always be followed by "bar 2"

Rule #2

There will be 100 "foo", "bar 1", "bar 2" lines printed, giving a total of 300 output lines.

b. If lines 14 and 18 are commented out, give two rules concerning the output of the program. **(0.5% + 0.5%)**

Rule #1

"foo" will always print before "bar 1" in the same thread

Rule #2

There will never be two "bar 1" statements printed before a "bar 2" statement.
(NOTE: there could be a "foo" printed between "bar 1" and "bar 2")

10. **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%)**
When we create a lock, we can specify that we want the lock to be fair. In that case, calling `signal()` would move the thread that has been waiting the longest on the condition to be moved to the Ready state.

It may also be more efficient to only move one thread back into the Ready state instead of moving a lot of threads if there are a lot waiting on a condition.

11. **Multi-Threading and Parallel Programming** – Explain why a program written using parallel computing could run slower than a program written using multi-threading. **(0.5%)**
There is overhead that needs to be taken into consideration when spawning a new thread and moving it into a different core. The amount of time it takes to execute the functionality of the thread should outweigh the amount of time to spawn the new thread if you want to experience a decrease in runtime.

12. **Distributed Programming –** We don't typically have too many security concerns when writing multi-threaded or parallel code, but security is a major concern with distributed programming. Explain why. **(0.5%)**
With distributed programming, code is being sent from one memory space to another to execute. This typically involves sending code from one computer to another. If the client computer is not trusted, security needs to be taken into consideration.

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

Here is the `University` table.

universityID	longname	shortname	mascot
1	University of Southern California	USC	Trojans
2	University of California Los Angeles	UCLA	Bruins
3	Stanford	Stanford	Trees
4	University of California Berekely	Cal	Bears

Here is the `Department` table.

departmentID	deptname	deptabbr
1	Computer Science	CSCI
2	Law	LAW
3	Medicine	MED

Here is the `Ranking` table.

rankingID	universityID	departmentID	rank
1	1	1	1
2	2	1	4
3	3	1	2
4	4	1	3
5	1	2	3
6	2	2	3
7	3	2	1
8	4	2	4
9	1	3	2
10	2	3	3
11	3	3	1

a. Write the SQL code to get the rankings of all of the Computer Science departments sorted in ascending order by rank. In other words, the following table should be returned by the `SELECT` statement. **(0.5%)**

shortname	rank
USC	1
Stanford	2
Cal	3
UCLA	4

```
SELECT u.shortname, r.rank
     FROM University u, Department d, Ranking r
     WHERE     d.deptname='Computer Science'
          AND  u.universityID=r.universityID
        AND    d.departmentID=r.departmentID
     ORDER BY r.rank ASC;
```

b. Draw the table that is returned from the following query. **(0.5%)**

```
SELECT      u.shortname, d.deptabbr, r.rank
  FROM      University u, Department d, Ranking r
  WHERE     u.universityID=r.universityID
        AND d.departmentID=r.departmentID;
```

shortname	deptabbr	rank
USC	CSCI	1
USC	LAW	3
USC	MED	2
UCLA	CSCI	4
UCLA	LAW	3
UCLA	MED	3
Stanford	CSCI	2
Stanford	LAW	1
Stanford	MED	1
Cal	CSCI	3
Cal	LAW	4

c. Write the SQL code to create the `Ranking` table. **(0.5%)**

```
CREATE TABLE Ranking (
    rankingID INT(11) PRIMARY KEY AUTO_INCREMENT,
    universityID INT(11) NOT NULL,
    departmentID INT(11) NOT NULL,
    rank INT(2) NOT NULL,
    FOREIGN KEY fk1(universityID) REFERENCES
University(universityID),
    FOREIGN KEY fk2(departmentID) REFERENCES
Department(departmentID)
);
```

Extra Credit Questions

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

Extra Credit (0.25%) – I have been questioning whether programming exams are an accurate form of assessment for CSCI 201, and I would like your input on it. I realize the exam this semester was long, but do you think programming exams provide a useful measure for determining whether you learned the material covered in the class and through the assignments?

Yes **No** **Maybe**

Please explain.

Extra Credit (0.25%) – Fill in the following table with one checkmark in each row answering the following question. Do you think the amount of coverage of each topic in the class was too little, just right, or too much?

Topic	**Too little coverage**	**Perfect amount of coverage**	**Too much coverage**
Java basics			
HTML, CSS, JavaScript			
JSP, Servlets			
Networking			
Databases, JDBC			
Multi-threading			
Concurrency issues (monitors, locks, conditions, semaphores)			

Please explain.