Final Score ________________

Name ______________________________ ID ____________________________ 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%)**

   <u>Response #1</u>

   <u>Response #2</u>

□

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%)**

□ **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

Reason #2

□ **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

Parallel

Distributed

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. **(0.5%)**

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

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%)**

    b. Why does `System.gc()` not cause the garbage collector to be run immediately? **(0.5%)**

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%)**

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

c. How many hosts can be on the network? **(0.5%)**

d. How many hosts can be on the subnetwork? **(0.5%)**

e. What is the network and subnetwork combination? **(0.25% + 0.25%)**
Binary

Decimal

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)

Last Assignable IP Address (in decimal)

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

```
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;
public class Question8 {
    private Lock lock = new ReentrantLock();
    public void foo() {
          try {
                lock.lock();
                System.out.println("foo");
          } finally {
                lock.unlock();
          }
    }

    public void bar() {
          try {
                lock.lock();
                System.out.println("bar 1");
                foo();
                System.out.println("bar 2");
          } finally {
                lock.unlock();
          }
    }

    public static void main(String [] args) {
          Question8 q8 = new Question8();
          q8.bar();
    }
}
```

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%)**

b. What is the output of the above code? **(0.5%)**

c. If the variable `lock` was not reentrant, what would the output of the code be? **(0.5%)**

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

Rule #2

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

Rule #1

Rule #2

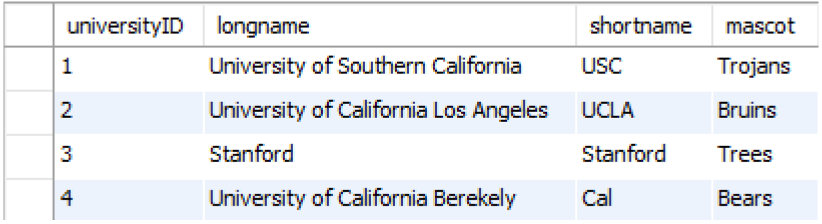
□ **10. Conditions –** There are two ways to wake up a thread that is waiting on a condition – `signal()` and `signalAll()`. Since we don’t have control over the order threads are switched into the CPU from the Ready state, why would we ever call `signal()` instead of `signalAll()` on a Condition? **(0.5%)**

□ **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%)**

□ **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%)**

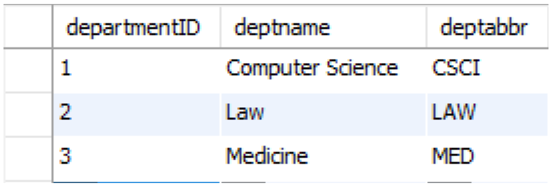
13. Databases and SQL – Answer the following questions concerning the database below.

Here is the `University` table.



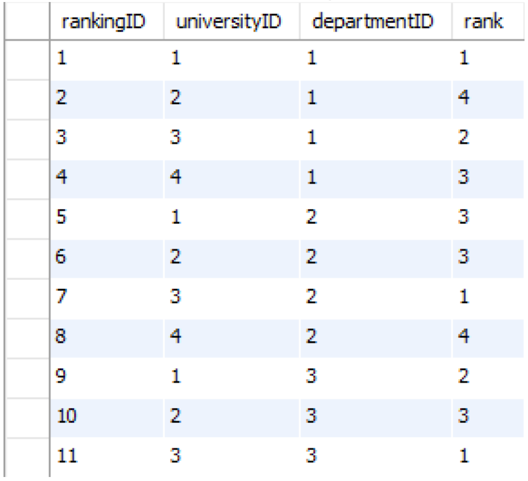
	universityID	longname	shortname	mascot
	1	University of Southern California	USC	Trojans
	2	University of California Los Angeles	UCLA	Bruins
	3	Stanford	Stanford	Trees
	4	University of California Berekely	Cal	Bears

Here is the `Department` table.



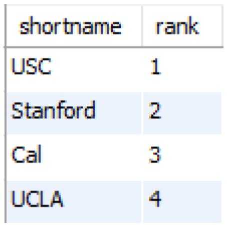
	departmentID	deptname	deptabbr
	1	Computer Science	CSCI
	2	Law	LAW
	3	Medicine	MED

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

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;
```

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

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.