CprE 308 Quiz 3

Department of Electrical and Computer Engineering
Iowa State University

You can write on the back of the sheet.

Student Name:

1. (3 points) For dining philosophers problem, we showed a solution with random delays, why do we need random delays?

```
Philosopher i:
    Think();
    while (unsuccessful) {
        wait(random());
        lock(left_chopstick);
        if (trylock(right_chopstick))
            unsuccessful = 0;
        else
            unlock(left_chopstick);
    }
    Eat();
    unlock(left_chopstick);
    unlock(right_chopstick);
```

Answer: The reason why we want to introduce the random delay is to avoid **starvation** problem. We can ensure this by adding a random delay before retrying. It is unlikely that our random delays will be in sync too many times

2. (3 points) A computer system has a total of r resources. Each process needs 5 ($5 <= r$) resources to finish. The number of processes is n. What can be the maximum n for which the system is deadlock free? Why? (Give equation)

Answer: If a process has 5 resources, it can finish and cannot be involved in a deadlock. Therefore, if we assume the worst case scenario wherein every process has 4 resources and needs another one and there is one resource left over, one process can finish and release all its resources, letting the rest finish too.
Therefore the minimum value of r to avoid deadlock is $r >= 4n + 1$. Thus, the maximum value of n for which the system is deadlock free is $n=(r-1)/4$.

3. (3 points) A system has 4 processes and 5 allocatable resources. The current resource allocation and maximum needs are as follows:

	Allocated	Maximum
Process A	1 0 2 1 1	1 1 2 1 3
Process B	2 0 1 1 0	2 2 2 1 0
Process C	1 1 0 1 0	2 1 3 1 0
Process D	1 1 1 1 0	1 1 2 2 1

Available resources: 0 0 x 1 1

What is the smallest value of x for which this is a safe state?

Answer:
The needs matrix is as follows:
0 1 0 0 2
0 2 1 0 0
1 0 3 0 0
0 0 1 1 1

If x is 0, we have a deadlock immediately. If x is 1, process D can run to completion.
When it is finished, the available vector is 1 1 2 2 1. Unfortunately, we are now deadlocked. If x is 2, after D runs, the available vector is 1 1 3 2 1 and C can run. After it finishes and returns its resources the available vector is 2 2 3 3 1, which will allow B to run and complete. Now the available vector becomes 4 2 4 4 1. But we can never satisfy A's request. But the system is not in a deadlock situation (not satisfy the four conditions). Therefore, the smallest value of x that avoids a deadlock is 2.