

Prepare your answers to the following questions either in a plain text file or in a Microsoft Word file. Answer each question clearly and concisely, and state conclusions using complete sentences. Explanatory tables and/or diagrams are acceptable, but there must always be a written discussion as well.

Submit your file to the Curator system by the posted deadline for this assignment. No late submissions will be accepted.

1. [20 points] Consider the following proposed solution to the infinite-buffer producer-consumer problem (taken from Stallings and the notes on concurrency):

```
int n;
binary_semaphore s = 1;
binary_semaphore delay = 0;
```

```
void producer() {
    while (true) {
        produce();
        semWaitB(s);
        append();
        n++;
        if ( n == 1 )
            semSignalB(delay);
        semSignalB(s);
    }
}
```

```
void consumer() {
    semWaitB(delay);
    while (true) {
        semWaitB(s);
        take();
        n--;
        if ( n == 0 )
            semWaitB(delay);
        semSignalB(s);
        consume();
    }
}
```

```
int main() {
    n = 0;
    parbegin(producer, consumer);
}
```

According to Stallings, this solution is incorrect because it is possible for the producer and consumer to reach a deadlocked state. Demonstrate that deadlock is possible by creating a hypothetical trace (as shown on slides 21-22 of the concurrency notes) that results in deadlock.

| | Producer | Consumer | s | n | delay |
|-----|----------|----------|---|---|-------|
| 1 | | | 1 | 0 | 0 |
| ... | | | | | |

2. [10 points] Comment on the following solution to the Dining Philosophers problem:

A hungry philosopher waits until his right fork is available, and then picks it up. Once he has acquired his right for, if his left fork is available, he picks it up and eats; otherwise, he puts down his right fork and repeats the cycle.

Consider both deadlock and starvation.

3. [10 points] Consider a system with a total of 200 units of memory, allocated to three processes as shown:

| Process | Max | Hold |
|---------|-----|------|
| 1 | 90 | 60 |
| 2 | 150 | 70 |
| 3 | 40 | 15 |

Apply Dijkstra's Banker's Algorithm to determine whether it would be safe to grant each of the following requests. If yes, indicate a possible sequence of terminations that could be guaranteed possible. If no, show the reduction of the resulting allocation table.

- A fourth process arrives, with a maximum memory need of 60 and an initial need of 20 units.
- A fourth process arrives, with a maximum memory need of 60 and an initial need of 25 units.