# Homework 2

### James Albert, 16004325

1.

- 000111 is consistent with the following possible sequence of operations

  ```
  z <- 1
  print(x, y)
  y <- 1
  print(x, z)
  x <- 1
  print(y, z)
  ```

- 001100 is inconsistent because there are no possible sequences of operations that satisfy this print statement. We can prove this with logical reasoning. The first print statement called shows 2 things: 1) only 1 write operation has been called before this print statement, 2) the variables in the print statement must not be the variable that was just written to. Some of the possible sequences that get close to forming a solution are:

  ```
  x <- 1
  print(y, z)
  z <- 1
  print(x, z)
  ```

  ...

  ```
  y <- 1
  print(x, z)
  x <- 1
  print(x, y)
  ```

  ...

  ```
  z <- 1
  print(x, y)
  x <- 1
  print(x, z)
  ```

  but these all fall short when the last print statement requires 2 variables to not be set which is an impossibility given that we've satisfied the other print statements' constraints.

2.
   ```
   L1      w(1)        R(1)               R(1)
   -------------------------------------------
   L2            PW(1)       W(2)  R(2)
   ```

- This diagram **does not** show the provision of monotonic-read consistency; there must be a PW in order for L1 to read the most up-to-date version of data item X. A correct diagram would be something like this:

  ```
  L1      w(1)        R(1)      PW(2)      R(1)
  -------------------------------------------
  L2            PW(1)      W(2)      R(2)
  ```

- This diagram **does** provide monotonic-write consistency because there is a PW before any successive write operations. The fact that L1's last read operation returns a stale value is irrelevant.

3.

- read-your-writes consistency **is guaranteed** when using a blocking primary-backup protocol because the propagation of the update is waited upon before anything else can happen by the same process. Therefore, after a write, the read will take place after the propagation is completed.

- read-your-writes consistency **is not guaranteed** when using a non-blocking primary-backup protocol because the propagation of the update will not be waited upon before anything else can happen by the same process. Therefore, after a write, the read may take place before the propagation is completed.

4.

- The overhead comes from the fact that each process must acquire and release a variable first before performing any operation on it and having that update be propagated respectively. This entails making sure that the process requesting the variable will use the most updated version of that variable and that processes that request the updated variable will be able to see the changes.
- I would expect the advantages of using a critical section method to be that you can explicitly request an up-to-date copy of the data item and continue through the process knowing you have it. The disadvantages would be having that extra overhead in the application layer.

5.

| C1 | C2 |
|---|---|
| invalid->exclusive, from memory | invalid |
| exclusive->modified, from memory | invalid |
| modified->shared, from cache | invalid->shared, from cache |
| shared->invalid, from cache | shared->modified, from memory |
| invalid, from cache | modified->shared, from cache |