Homework 3

Question 1

Let the body of the if statements be $p_{2.2}$ and $q_{2.2}$.
After executing the sequence $q_1p_1p_2q_2q_{2.2}q_3q_4p_{2.2}p_3p_4$. The Process p and q will be in the critical section at the same time. Because while executing q_3, the value for $wantp$ is 0 and $wantq$ is then -1. So, we have $0 \neq 1$. Afterwards, $wantp$ will be set to 1 which is apparently not equal to $wantq = -1$. This would results in passing the line p_3.

Question 2

Can you find any reorderings that break mutual exclusion and/or eventual entry?
Yes. For example, if we reorder the code for p_3: $\boldsymbol{await}\ \forall j, flag[j] < 3$ into:

```
flag[1] < 3;
flag[0] < 3;
flag[2] < 3;
```

The mutual exclusion requirement would be broken.

Are there any awaits that don't seem sensitive to reordering at all?
Yes. For example, the **await** for hold door: p_7: $\boldsymbol{await}\ \exists j, flag[j] = 4$ is not sensitive to the order.

What if you reorder the tests for all the awaits in the exact same way?
If the tests are reordered in the same way, mutual exclusion can still be broken.
Example: replace ordering from $0 \rightarrow 1 \rightarrow 2$ to $1 \rightarrow 0 \rightarrow 2$.

And finally, based on any error trails you obtain, can you form an educated guess about why the test order matters?
According to the examples I obtained so far, I guess that the reason why ordering matters is relevant to p_9 or p_{11}, which are:

$$p_9: \boldsymbol{await}\ \forall j < i. flag[j] < 2$$

$$p_{11}: \boldsymbol{await}\ \forall j > i. flag[j] < 2\ or\ flag[j] > 3$$

Because some of them only check the subsets of all processes, the expanded atomic code for each process is slightly different. That may be the reason why the test order matters.

Question 3

a) If either A or B satisfies mutual exclusion, then C satisfies mutual exclusion.

Correct. Because only one process can enter B at the same time if A satisfies mutual exclusion, and critical section is contained in B which means that the critical section can only be executed by one process. On the other hand, if B satisfies mutual exclusion, the case is not worse than only running algorithm B, so mutual exclusion is also satisfied.

b) If A has no unnecessary delay and B satisfies mutual exclusion then C has no unnecessary delay.

Correct. Because only one process can enter B at the same time, and because A has no unnecessary delay, the process in B can never be prevented from entering critical section by the other process. And for only one process, it can never cause unnecessary delay it self.

c) If A satisfies mutual exclusion and B has no unnecessary delay then C has no unnecessary delay.

Incorrect. Because A has unnecessary delay, the process that is waiting to enter critical section can be prevented by the other in noncritical section in A.

d) If A is deadlock free and B guarantees eventual entry then C guarantees eventual entry.

Incorrect. Because A doesn't guarantee eventual entry, so it's possible that one process is hung up and the other process is executed forever.