- **Ex. 1**
  1. There should not be deadlock happening because as there are three resouces, there should always enough resource for one process to have two.

  2. n should be at most 5, which can guarantee that at least one process can get the second resource.

  3. $x \leq (1000 - 20 \times 35ms - 10 \times 20ms - 5 \times 10ms)/4 = 12.5ms$

  4. It means that the process will be triggered twice in the whole cycle. It might make sense in the implementing scheduling of processes with different priorities.

  5. From the source code it can be done. If a program operates data together with inputing and outputing them, then it might be I/O bound. In the runtime, maybe syscalls such as top which reveals the running status of CPU and I/O devices can help.

- **EX. 2**

  1. It should be
  $$\begin{bmatrix} 7 & 4 & 3 \\ 1 & 2 & 2 \\ 6 & 0 & 0 \\ 0 & 1 & 1 \\ 4 & 3 & 1 \end{bmatrix}$$

  2. It is in a safe state. Run process as 2,4,5,1,3 can help.

  3. Yes, the order above can help.

- **Ex. 4**
  This can be done by searching the word "scheduling" in the derictory usr/src. And it can be found in minix/kernel/main.c. In teh code, what I found is that the kernel will check whether certain process is schedulable on startup, and if it is , the kernel will reset its priority and run it. And it can also been seen that root commands and syscalls have higher priority than other processes.

- **Ex. 5**

  1. This should be done using count_lock. Use this to access the counter and change it for reading.

```
void read_lock(){
    down(count_lock);
    if (counter==0) down(db_lock);
    counter++;
    up(count_lock);
}

void read_lock(){
    down(count_lock);
    if (counter==1) up(db_lock);
    counter--;
    up(count_lock);
}
```

2. It means that the writer cannot find time to write on the db, bercause readers are coming continuously.

3. This can be done by adding down(read_lock) and up(read_lock) in both read_lock() and write_lock()

4. No as the writer still has higher priority, so the problem should not be regarded as solved.