3)

An advantage of HRRN is that it takes into account how long a process has been waiting. This can help make sure one process doesn't get starved from resources. However, a downside is that smaller processes grow in scale much faster. For example, if you had a process that expected 10 steps and a process that expected 1, the ratio of the first process is growing at 1/10 of the speed of the second process. This leads to HRRN favoring shorter processes and delaying longer processes unfairly.

4)

a)

No 2 would not be a good time quantum in this scenario. This time quantum is way to short. You would loose a lot to overhead from switching processes.

b)

20 would also not be a good time quantum. Since the max is 15 it would effectively turn our scheduler into FCFS. Since each task would complete in 1 turn, and they happen in the same order they arrived, we would loose the benefit of RR.

5)

Schedulers like SRT, SPN, and HRRN require us to have a somewhat accurate guess about how long a process could take. This can be extremely difficult as all processes vary and it can be hard to estimate. Schedulers like FCFC, RR, and Multilevel Feedback do not need this because they don't care how long the process will take. They just give each of the processes their turn and move on. Determining the time a process will take is difficult which makes using SRT, SPN, and HRRN difficult.

6)

It could be beneficial to keep threads running on the same core when they share information. If multiple threads share data between them a lot they would both easily be able to access it from the same cache without having to move stuff around. It could be beneficial to move it to another core so both could actually run at the same time which would get them could possibly get them done faster.

7)

This can benefit us because if threads are similar, they may be interacting with similar parts of memory. If we put all related threads into the same processor, it means the memory that one is

accessing is already in the cache so another thread could easily also access it without having to pull it in.

8)

The goal of a real time system isn't speed, the goal is just to have tasks done at the most valuable times. One example of where speed wouldn't matter in a RTOS is if you have a periodic process A that takes 5 steps and must be completed every 50 steps. You would have a while to complete A so you don't need to rush it. This becomes clearer when we are on a system where once a process starts, it's not off until it finishes. If you knew that sometime soon process B would pop up, which needed to be started right away, and you were pretty sure it would happen in the next 3 steps, the correct choice would be to do nothing and wait for B to arrive. Since B must start right away, we can't kick A off to do it, and A isn't due for a while, the best option is just to wait. This example shows us that getting everything done as fast as possible isn't exactly what we need, just getting it done at the correct time.