1. What is the difference between a preemptive resource and a nonpreemptive resource?

A preemptive resource is one that will allow a task to use it for a set amount of time or until another task becomes more eligible for it then it will switch over. Non-preemptive resources will be used for a task until it is finished, even if another more eligible task becomes available.

2. What are the conditions required for a deadlock to occur?

There are four conditions that must be met in order for a deadlock to occur. The first is that a mutual exclusion must take place. A mutual exclusion is when one task is holding a resource that is non-sharable. The second is known as resource holding and refers to when one task is holding resources and is requesting another resources that are also being held. The third is called the no preemption condition, which refers to the operating system not allowing resources to be preemptive so that they cannot be released to different processes. The last is the circular wait, which states that each process must be holding a resource, and waiting on another thus none of them will ever be freed.

3. What are the four approaches to handling deadlock?

It is not a good idea to allow deadlocking to occur because even after it’s been corrected a large amount of effort was wasted in detecting and rectifying the problem. If deadlocking does occur within an operating system there is usually a system detect when a deadlock has occurred. Once it realizes a deadlock has occurred it will usually rollback or reset a few tasks to release the system from deadlock. This method is very costly however.

4. How is deadlock prevented?

The ways of avoiding these conditions are just isolating the causes for them and making sure that they don’t happen. With mutual exclusion the operating system just needs to make sure that no task is exclusively holding on to a resource. With hold and wait, the operating system just needs to require the tasks to request the full amount of resources that they will need before hand and organize the tasks. With preemption it’s a little tricky, because if we just give tasks a certain amount of time thrashing may occur so this avoidance must be combined with one of the other solutions. If the above three are met then circular waiting will never occur.

5. What is starvation and how can it be avoided?

Starvation is when a scheduler does not allow a task to have CPU time because of “high priority” tasks that are in front of it. If it continually schedules higher priority tasks then the lower task will be said to starve. To avoid starvation the scheduler needs to be designed to enable equal usage of the CPU from all of the tasks.

**Chapter 6 problems.**  Answer problems 17, 19, and 20 at the end of chapter 6 on pages 463 to 466.  Each problem is worth 3 points (9 points total).