CO327: Operating Systems
Assignment 5
E/15/271

1. List three examples of deadlocks that are not related to a computer system environment.

- Two cars crossing a single-lane bridge from opposite directions.
- A person going down a ladder while another person is climbing up the ladder.
- Two trains traveling toward each other on the same track.

2. Consider the traffic deadlock depicted in Figure given in right

i. Show that the four necessary conditions for deadlock hold in this example.

ii. State a simple rule for avoiding deadlocks in this system.

i) The four necessary conditions for a deadlock are (1) mutual exclusion; (2) hold-and-wait; (3) no preemption; and (4) circular wait. The mutual exclusion condition holds since only one car can occupy a space in the roadway. Hold-and-wait occurs where a car holds onto its place in the roadway while it waits to advance in the roadway. A car cannot be removed (i.e. preempted) from its position in the roadway. Lastly, there is indeed a circular wait as each car is waiting for a subsequent car to advance. The circular wait condition is also easily observed from the graphic.

ii) A simple rule that would avoid this traffic deadlock is that a car may not advance into an intersection if it is clear it will not be able immediately to clear the intersection.

3. A single-person-wide suspension bridge connects the two sides (North side and South side) of a big University. Students residing on both sides use this bridge to attend classes on the other side. The bridge can become deadlocked if a northbound and a southbound student get on the bridge at the same time (students of this University are known to be stubborn and are unable to back up).

i. Using semaphores and/or mutex locks, design an algorithm in pseudocode that prevents deadlock. Initially, do not be concerned about starvation (the situation in which northbound farmers prevent southbound farmers from using the bridge or vice versa).

```
semaphore ok_to_cross = 1;
void enter_bridge() {
        P(ok_to_cross);
} void exit_bridge() {
         V(ok_ to_cross);
}
```

ii. How would you modify your solution above so that it is starvation-free.

```
Monitor Bridge {
      int nWaiting=0, sWaiting=0, sOnBridge=0, nOnBridge=0;
      condition north_turn, south_turn;
      enterNorth() {
            if (sWaiting>0 || sOnBridge>0)
            {
                  nWaiting++;
                  wait(north_turn);
                  while (sOnBridge>0)
                        wait(north_turn);
                  nWaiting--;
            }
            nOnBridge++;
            if (sWaiting==0)
                  signal(north_turn);
      }
      exitNorth() {
            nOnBridge--;
            if (nOnBridge ==0)
                  signal(south_turn);
      }
      enterSouth() {
            if (nWaiting>0 || nOnBridge>0)
            {
                  sWaiting++;
                  wait(south_turn);
                  while (nOnBridge>0)
                        wait(south_turn);
                  sWaiting--;
            }
            sOnBridge++;
            if (nWaiting==0)
                  signal(south_turn);
      }
      exitSouth() {
            sOnBridge- -;
            if(sOnBridge == 0)
                  signal(north_turn);
      }
}
```

Grading guide:

- Wait on a condition variable is not matched by signal/broadcast as a result the blocked process will never wake up.
- The solution is not starvation-free.
- Wait on Condition and semaphore is not the same. First wait on a Condition blocked the process. So your solution blocked all farmers and no one could enter the bridge.
- Solution does not contain any pseudo-code/program. Only a solution approach is provided. Comment (no point deducted): Many of you used "if" instead of "while" before waiting on a conditional variable. For mesa style monitors (actual implementation of

monitor is of this type), you have to use "while" because Condition might not hold when the waiter returns. See class lecture slides for details.

4. A In a real computer system, neither the resources available nor the demands of processes for resources are consistent over long periods (months). Resources break or are replaced, new processes come and go, and new resources are bought and added to the system. If deadlock is controlled by the banker's algorithm, which of the following changes can be made safely (without introducing the possibility of deadlock), and under what circumstances?
i. Increase Available (new resources added).
ii. Decrease Available (resource permanently removed from system).
iii. Increase Max for one process (the process needs or wants more resources than allowed).
iv. Decrease Max for one process (the process decides it does not need that many resources).
v. Increase the number of processes.
vi. Decrease the number of processes

i. Increase Available (new resources added)—This could safely be changed without any problems.

ii. Decrease Available (resource permanently removed from system)—This could have an effect on the system and introduce the possibility of deadlock as the safety of the system assumed there were a certain number of available resources.

iii. Increase Max for one process (the process needs more resources than allowed, it may want more)— This could have an effect on the system and introduce the possibility of deadlock.

iv. Decrease Max for one process (the process decides it does not need that many resources)—This could safely be changed without any problems.

v. Increase the number of processes—This could be allowed assuming that resources were allocated to the new process(es) such that the system does not enter an unsafe state.

vi. Decrease the number of processes—This could safely be changed without any problems.

5. Consider the following snapshot of a system:

	Allocation	Max	Available
	A B C D	A B C D	A B C D
P_0	2 0 0 1	4 2 1 2	3 3 2 1
P_1	3 1 2 1	5 2 5 2	
P_2	2 1 0 3	2 3 1 6	
P_3	1 3 1 2	1 4 2 4	
P_4	1 4 3 2	3 6 6 5	

Answer the following questions using the banker's algorithm:
i. Illustrate that the system is in a safe state by demonstrating an order in which the processes may complete.

	Needed			
	A	B	C	D
P0	2	2	1	1
P1	2	1	3	1
P2	0	2	1	3
P3	0	1	1	2
P4	2	2	3	3

Safe sequence: <P0, P3, P4, P1, P2>

ii. If a request from process P1 arrives for (1, 1, 0, 0), can the request be granted immediately?

	Allocated				MAX				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P0	2	0	0	1	4	2	1	2	2	2	2	1
P1	**4**	**2**	**2**	**1**	**5**	**2**	**5**	**2**				
P2	2	1	0	3	2	3	1	6				
P3	1	3	1	2	1	4	2	4				
P4	1	4	3	2	3	6	6	5				

Needed

	A	B	C	D
P0	2	2	1	1
P1	**1**	**0**	**3**	**1**
P2	0	2	1	3
P3	0	1	1	2
P4	2	2	3	3

Yes. The request can be granted, Safe sequence: <P0, P3, P4, P1, P2>

iii. If a request from process P4 arrives for (0, 0, 2, 0), can the request be granted immediately?

	Allocated				**MAX**				**Available**			
	A	B	C	D	A	B	C	D	A	B	C	D
P0	2	0	0	1	4	2	1	2	2	2	2	1
P1	4	2	2	1	5	2	5	2				
P2	2	1	0	3	2	3	1	6				
P3	1	3	1	2	1	4	2	4				
P4	**1**	**4**	**5**	**2**	**3**	**6**	**6**	**5**				

Needed

	A	B	C	D
P0	2	2	1	1
P1	1	0	3	1
P2	0	2	1	3
P3	0	1	1	2
P4	**2**	**2**	**1**	**3**

This is an unsafe state. Deadlock can occur. P4's request will be denied.

6. Is disk scheduling, other than FCFS scheduling, useful in a single-user environment? Explain your answer.

In a single-user environment, the I/O queue usually is empty. Requests generally arrive from a single process for one block or for a sequence of consecutive blocks. In these cases, FCFS is an economical method of disk scheduling. But LOOK is nearly as easy to program and will give much better performance when multiple processes are performing concurrent I/O, such as when aWeb browser retrieves data in the background while the operating system is paging and another application is active in the foreground.

7. Explain why SSTF scheduling tends to favour middle cylinders over the innermost and outermost cylinders.

The center of the disk is the location having the smallest average distance to all other tracks. Thus the disk head tends to move away from the edges of the disk. Here is another way to think of it. The current location of the head divides the cylinders into two groups. If the head is not in the center of the disk and a new request arrives, the new request is more likely to be in the group that includes the center of the disk; thus, the head is more likely to move in that direction.

8. None of the disk-scheduling disciplines, except FCFS, is truly fair (starvation may occur)

i. Explain why this assertion is true.
ii. Describe a way to modify algorithms such as SCAN to ensure fairness.
iii. Explain why fairness is an important goal in a time-sharing system.
iv. Give three or more examples of circumstances in which it is important that the operating system be unfair in serving I/O requests.

i) New requests for the track over which the head currently resides can theoretically arrive as quickly as these requests are being serviced.

ii) All requests older than some predetermined age could be “forced” to the top of the queue, and an associated bit for each could be set to indicate that no new request could be moved ahead of these requests. For SSTF, the rest of the queue would have to be reorganized with respect to the last of these “old” requests.

iii) To prevent unusually long response times.

iv) Paging and swapping should take priority over user requests. It may be desirable for other kernel-initiated I/O, such as the writing of file system metadata, to take precedence over user I/O. If the kernel supports real-time process priorities, the I/O requests of those processes should be favored.

9. Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4,999. The drive is currently serving a request at cylinder 2,150, and the previous request was at cylinder 1,805. The queue of pending requests, in FIFO order, is: 2069, 1212, 2296, 2800, 544, 1618, 356, 1523, 4965, 3681
Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms?
a. FCFS
b. SSTF
c. SCAN
d. LOOK
e. C-SCAN
f. C-LOOK

a. The FCFS schedule is 143, 86, 1470, 913, 1774, 948, 1509, 1022,1750, 130. The total seek distance is 7081.
b. The SSTF schedule is 143, 130, 86, 913, 948, 1022, 1470, 1509, 1750,1774. The total seek distance is 1745.
c. The SCAN schedule is 143, 913, 948, 1022, 1470, 1509, 1750, 1774,4999, 130, 86. The total seek distance is 9769.
d. The LOOK schedule is 143, 913, 948, 1022, 1470, 1509, 1750, 1774,130, 86. The total seek distance is 3319.
e. The C-SCAN schedule is 143, 913, 948, 1022, 1470, 1509, 1750, 1774,4999, 86, 130. The total seek distance is 9813.
f. (Bonus.) The C-LOOK schedule is 143, 913, 948, 1022, 1470, 1509,1750, 1774, 86, 130. The total seek distance is 3363.