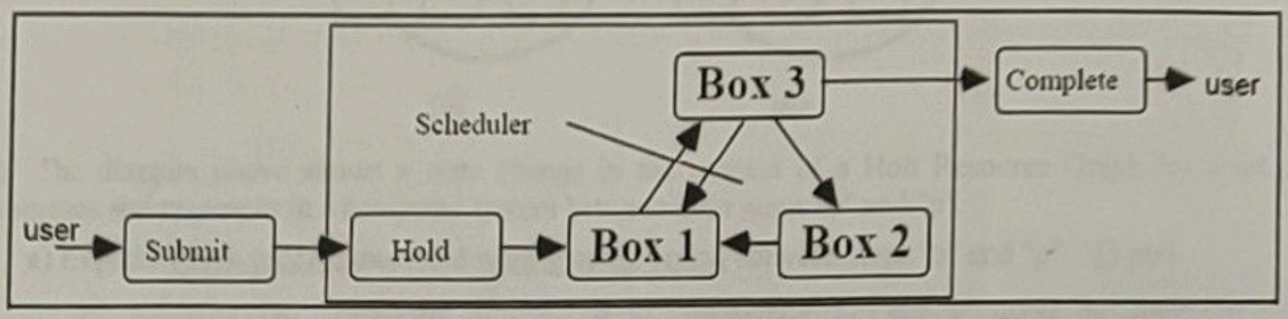
19

Midterm 2

Name: NIMESH NISCHAL

100

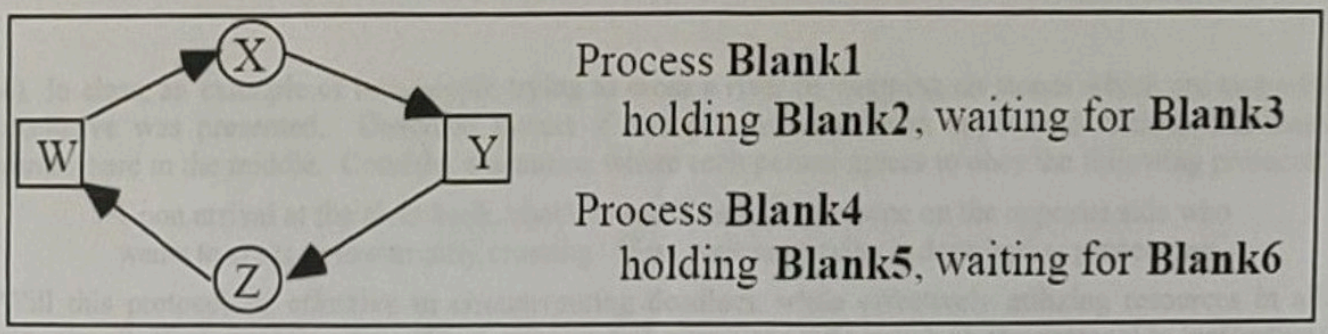


1) The diagram above shows three box labels, one for each state that a process can be in while "active" in the computer system. We used this diagram many times throughout the semester to understand several topics such as Hyper-Threading, Deadlocks, Race Conditions and I/O Buffering. List the process state that each box represents. [3 pts]

Box 1: READY

Box 2: WAIT

Box 3: RUN



2) The diagram above shows six blue blanks in the context of a Holt Resource Graph for a set of processes and resources in a computer system. Describe the current state of the system by filling in the following six blanks with the appropriate letter: W, X, Y, or Z. Note that some letters will be used for more than one blank, and that there is more than one answer possible. [6 pts]

Blank 1: X

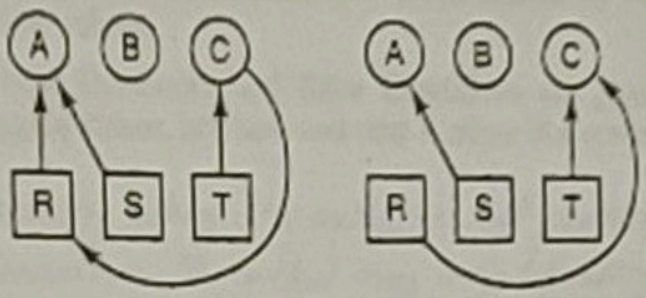
Blank 2: W

Blank 3: Y

Blank 4: Z

Blank 5: Y

Blank 6: W



(o) (p)

3) The diagram above shows a state change in the context of a Holt Resource Graph for a set of processes and resources in a computer system between time steps "o" and "p".

a) Explain which process executed what specific action between steps "o" and "p". [3 pts]

A releases the resource R. As A has completed its task of using the resource R, it releases R which gets assigned to the process C in step "p"

b) Assuming each process shown will only need a maximum of two resources, which process can now run at step "p", but could not at step "o" ? [3 pts]

C can run in step "p" as it has both the resources T and R assigned to itself.

4) In class, an example of two people trying to cross a river by stepping on stones which are mutually exclusive was presented. Deadlock occurs if two people cross from opposite directions and meet somewhere in the middle. Consider a situation where each person agrees to obey the following protocol:

> Upon arrival at the river bank, check to see if there is someone on the opposite side who wants to cross or is currently crossing. If so, wait until (s)he is done before proceeding.

Will this protocol be effective in circumventing deadlock while effectively utilizing resources in all situations? If so, explain why. If not, explain and give a scenario in which the protocol is ineffective. [4 pts]

The protocol is not effective when two tasks are ready to run at exactly the same time. In such a case, deadlock will occur.
In the ~~ex~~ above example, when two people arrive at two ends of the river bank at ~~exact~~ the same time, both of them see that the other person wants to cross the river. So, according to the above protocol, both of them will wait indefinitely. This leads to a deadlock.

5) Bugs caused by Deadlocks are nondeterministic. Bugs caused by Race Conditions are not only nondeterministic, they are also subtle.

a) Explain why and how Deadlocks and Race Conditions are Nondeterministic. Specifically, describe what Nondeterministic means, and how and why it arises in a computer system. [4 pts]

A task is nondeterministic when its output is not guaranteed to be the same on every successful execution. The output may vary depending on external factors like scheduling, access sequences, etc.
Deadlocks and race conditions depend on the sequence of execution of multiple processes. ~~If the processes are~~ The OS may schedule the processes in such a way that no deadlock ~~will~~ or race conditions occurs, or ~~it may~~ in such a way that they do occur. The OS schedules processes randomly, which makes these ~~bugs~~ issues nondeterministic.

b) Explain why and how Race Conditions are Subtle. Specifically, describe what Subtle means in the context of Race Conditions. [5 pts]

Race conditions may not break the execution of a program, may not ~~lead to~~ cause any exception or warning messages to occur. But, they can cause the expected outcome of a ~~program~~ program to change. Because of no exceptions, warnings or system failures, race conditions are difficult to detect, and thus, they are subtle.

6a) Write the pseudo-code (as shown in class) for the semaphore operation performed by P(S). [3 pts]

```
P(S):
    S=S-1
    IF S<0:
        WAIT
```

b) Write the pseudo-code (as shown in class) for the semaphore operation performed by V(S). [3 pts]

```
V(S):
    S=S+1
    IF S<=0:
        SIGNAL
```

7) The semaphore operations P and V are "special" functions. In particular, there is one very important property of semaphores that will guarantee mutual exclusivity between processes executing a P function, between processes executing a V function, and between processes executing a P function and a V function. What is this important property? You may reference the pseudo-code that you wrote in the previous problem and explain this property in the context of the pseudocode. [5 pts]

The semaphore operations P and V are indivisible, i.e., when a process is executing either of the two operations, the OS cannot ~~when~~ replace the process with another process to execute. ~~The Operations~~ This is done with the help of OS and hardware by disabling all the interrupts when any of these two Operations are executing.

(Continued in the next page...)

For example, if a process is executing P(S), and it has executed the first instruction S = S - 1, the OS cannot ~~sta~~ switch the process ~~with another process to execute~~ to run another process. The original process has to run the instruction IF (S < 0) WAIT before ~~it~~ it can be switched by the OS.

Due to this property, no two or more processes can run either of P(S) or V(S) in parallel. This ensures mutual ~~exclusitivity~~ exclusivity between processes executing these functions.

8) Shown below is the semaphore solution to the Readers/Writer problem as discussed in class.

Reader:

```
P(MUTEX)  <-- #2
  RC = RC + 1
  IF (RC = 1) THEN P(DB)  <-- #1
V(MUTEX)
  <read_data_base>
P(MUTEX)
  RC = RC - 1
  IF (RC = 0) THEN V(DB)
V (MUTEX)
GOTO Reader
```

Writer:

```
P(DB)
  <write_data_base>
V(DB)
GOTO Writer
```

a) What is the specific purpose of the MUTEX semaphore? [5 pts]

The MUTEX semaphore ensures only one reader can execute the underlying code at a time. That is, only one reader can update RC at a time. This prevents race condition on the variable RC.

b) The Readers-Writers solution as shown above has a potential flaw. What is the flaw and under what circumstances could it arise? [5 pts]

The readers are given a higher priority than the writers. So, the writers may face starvation.
If there are an infinite number of readers, or the number of readers at any given time never fall to 0, a writer will never be able to execute. This causes deadlock for the writer.

c) Assume a situation where a writer is currently in the critical region <write_data_base> and gets swapped out. Two reader processes start at statement 1 (at the label "Reader:") and try to read the database. Show, by marking arrows on the code, at what statement the first reader and second reader processes would be blocked. Label the arrows #1 and #2 for first and second reader. Explain why each reader gets blocked at that point. [8 pts]

The first reader will run P(MUTEX) and set the value to 0 from 1. Then it will increase the value of RC from 0 to 1. As the if condition is true, it will execute P(DB). Since, the writer is writing in the database, P(DB) is currently 0. Reader 1 sets ~~P(DB)~~ DB to -1 and waits for a signal.

The second reader starts running and executes P(MUTEX). As MUTEX is currently 0 (set by reader 1), it decreases the value to -1 and waits for a signal.

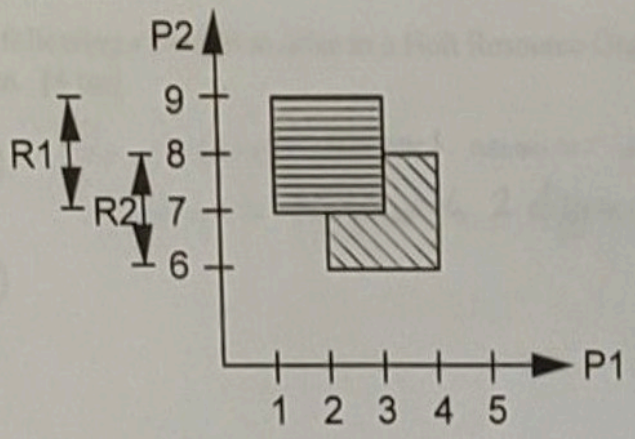
9) Shown below is the semaphore solution to the Producer/Consumer problem as discussed in class. Three semaphores are used: EMPTY, FULL, and MUTEX. Fill in the eight blank lines (denoting Semaphore Calls 1 thru 8 in the given code) with the P and V calls to EMPTY, FULL, MUTEX. [8 pts]

Producer:	Consumer:
<non-critical>	<non-critical>
Semaphore Call 1	Semaphore Call 5
Semaphore Call 2	Semaphore Call 6
<Fill Buffer>	<Consume Buffer>
Semaphore Call 3	Semaphore Call 7
Semaphore Call 4	Semaphore Call 8
<non-critical>	<non-critical>
GOTO Producer	GOTO Consumer

Semaphore Call 1: P(EMPTY)
Semaphore Call 2: P(MUTEX)
Semaphore Call 3: V(MUTEX)
Semaphore Call 4: V(FULL)

Semaphore Call 5: : P(FULL)
Semaphore Call 6: : P(MUTEX)
Semaphore Call 7: : V(MUTEX)
Semaphore Call 8: : V(EMPTY)

10) A resource trajectory plot is shown below for two processes, P1 and P2, which each need resources R1 and R2 at some point in their execution.



a) At what statement is resource R1 released by P1 ? [3 pts] 3

b) At what statement is resource R2 requested by P1 ? [3 pts] 2

c) What is the meaning and significance behind the two shaded regions in the diagram? [5 pts]

The shaded regions denote the area in which ~~the~~ any execution trajectory will violate the mutual execution condition.
The primary aim of any execution trajectory is to avoid these areas.

d) What would it mean if the trajectory entered the rectangle bounded by 1, 2, 6, 7 ? [5 pts]

If a trajectory enters the area, it will ~~bound~~ be bound to get into a deadlock in the future.

11) Show (via pseudo-code) how two semaphores, if used incorrectly across two processes, could lead to deadlock. Explain the specific sequence of operations in your pseudo-code that would lead to deadlock. [6 pts]

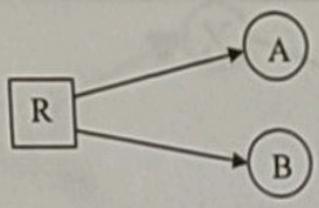
PROCESS 1	PROCESS 2
P(SEM1)	P(SEM2)
P(SEM2)	P(SEM1)
V(SEM2)	V(SEM1)
V(SEM1)	V(SEM2)

When process 1 runs P(SEM1) and then the OS ~~swiths~~ switches it with process 2, the second process runs ~~P(SEM~~ P(SEM2). Now, process 2 ~~runs~~ to runs P(SEM1) and is made to wait for Process 1 to release SEM1. The OS ~~to~~ switches Process 1 which runs P(SEM2) ~~to~~ that makes it wait for Process 2 to release SEM2. Thus, deadlock occurs as both processes are waiting for each other.

12a) Describe a Holt Resource Graph, and how it is useful in the context of Deadlocks. [5 pts]

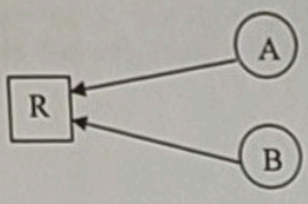
A Holt Resource Graph ~~dos~~ denotes the dependencies between processes and resources using a circle for a process and a square for a resource. A directed edge is used to denote if a process has been assigned a resource (directed towards the process) or a process is waiting for a resource (directed towards the resource).
They are used to detect deadlocks by identifying a circular pattern of edges directed in the same direction.

b) Is it possible for the following situation to arise in a Holt Resource Graph? Explain why not, or why and what it would mean. [4 pts]



No, as ~~the resource~~ each resource is mutually exclusive. R cannot be assigned to 2 different processes at the same time.

c) Is it possible for the following situation to arise in a Holt Resource Graph? Explain why not, or why and what it would mean. [4 pts]



Yes, as 2 ~~dos~~ processes, A and B, can wait for a resource, R, to be available at the same time.