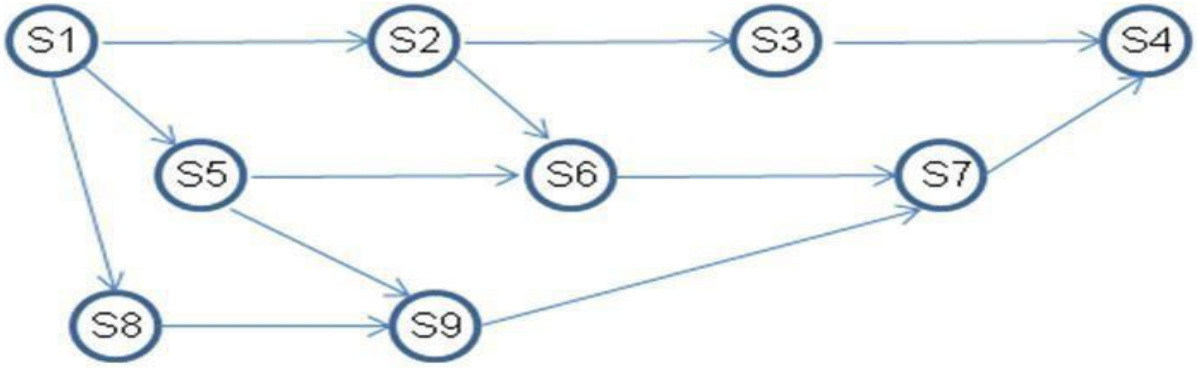
1) Assume you are given the following wait-graph that represents the relationship between multiple threads (s1,s2,s3,…). An arrow from one thread (Sy) to another (Sx) means that thread Sx must finish its computation before Sy starts. (For example: S1 has to wait for S2,S5,S8 to finish, S2 has to wait for s3,s6 to finish and so on.)

Use semaphores to enforce this relationship specified by the graph. Be sure to show the initial values and the locations of the semaphore operations. You will be marked based on finding the best solution with minimum number of semaphores.



- Give that:

S1 has to wait for S2,S5,S8 Before executing.

S2 has to wait for S3,S6.

S3 has to wait for S4.

S4 has to wait for NO ONE

S5 has to wait for S6,S9.

S6 has to wait for S7.

S7 has to wait for S4.

S8 has to wait for S9.

S9 has to wait for S7.

Except for the s1,s2,s5 most of processes wait for single process to execute.

- We will need one semaphore for each process. But none of the process is waiting for execution of the S1 we wont need the semaphore for the S1.
- To find the optimum solution we can see that the above wait graph share many common successor for a given process. That is we can use single semaphore that directs to its immediate successor.

Process	Immediate Predecessor	Immediate Successor
S1		S2,S5,S8
S2	S1	S3,S6
S3	S2	S4
S4	S3,S7	
S5	S1	S6,S9
S6	S2,S5	S7
S7	S6,S9	S4
S8	S1	S9
S9	S5,S8	S7

So, a process will have a V(semaphore) of his own and will hold a P(semaphore) of its successor.

process	Semaphore	V(OWN)	P(SUCCESSOR)
S4	A	V(A) V(A)	
S3	B	V(B)	P(A)
S7	C	V(C) V(C)	P(A)
S6	D	V(D) V(D)	P(C)
S9	E	V(E)	P(C)
S8	F	V(F)	P(E)
S1			P(E),P(G)
S2	G	V(G)	P(B),P(D)
S5	H	V(H)	P(D),P(E)

So process (S3,S7),(S6,S9) will respectively use only common semaphore of the successor. Therefore optimum number of semaphore required are 6. As follows. P(A),P(C),P(E),P(B),P(G),P(D).

2) What is the difference between Mesa and Hoare scheduling for monitors?

Ans:

1) Condition variable is queue for processes waiting for certain condition to happen so that they will continue the processing. What if 2 processes are waiting for monitors time which one will be given priority ?
2) So the process who is already in I/O and completed I/O is waiting also process in entry queue is also waiting which one will be given preference to get monitors time. Both Mesa and Hoare had different approach to the same problem.

Mesa	Hoare
• Mesa believed that Process already completed I/o will take the priority of the monitor.	• Hoare believed that process waiting in condition variable i.e queue processes will take in priority .
• Mesa, the signaling thread continues and the released thread yields the monitor.	• The signaler yields the monitor to released thread.
• Here Process will Wait in Queue and Signaling Process continues to use monitor.	• In Hoare, the release process takes the monitor and signalling process waits in queue.
• To let the signaler continue to run and allow the waiting process to start running only after the signaler has exited the monitor	• Hoare proposed letting the newly awakened process run, suspending the other one.
• The thread that signals keep the lock (and thus the processor).	• The thread that signals gives up the lock and the waiting thread gets the lock.
• In this monitor may evaluate multiple times, even if signal is incorrect.	• It must be very accurate to ensure signaling is not incorrect.
• Mesa Type Monitor uses only one context switch	• When signal occurs, Hoare type monitor uses two context switches. In first signaling process out and in second it processes in
• the waiting thread may need to wait again after it is awakened, because some other thread could grab the lock and remove the item before it gets to run.	• we can change the ‘while’ in Remove to an ‘if’ because the waiting thread runs immediately after an item is added to the queue.
• Mess style is used by (Nachos, Java, and most real operating systems)	• Hoare style is used in textbooks

3) The following pair of processes share a common variable X:

	Process A	Process B
L1:	int Y	int Z
L3:	X = Y	X = Z

X is set to 5 before either process begins execution. As usual, statements within a process are executed sequentially, but since no assumptions can be made regarding each process's speed of execution, statements in either process may execute in any order with respect to statements in the other process.

(a) How many different values of X are possible after both processes finish executing?

(b) Suppose the programs are modified as follows to use a shared binary semaphore S:

	Process A	Process B
L2:	P(S)	P(S)
L5:	V(S)	V(S)

S is set to 1 before either process begins execution or, as before, X is set to 5. Now, how many different values of X are possible after both processes finish executing?

(c) Finally, suppose the programs are modified as follows to use a shared binary semaphore T:

	Process A	Process B
L1:	int Y	int Z
L4:	V(T)	X = Z

T is set to 0 before either process begins execution or, as before, X is set to 5. Now, how many different values of X are possible after both processes finish executing?

Ans:

a) There will be one values of X, X will be either Y and Z. Depending upon which process will run first. As process are not increasing or decreasing the value. The value of x will be Y/Z.
b) The Value of X depends on either Y/Z. Here Int X is shared resource and we are using semaphore to stop corruption of this resource. So whoever grabs the critical section first will grab the int X and other won't be able to access it until. Process leaves the critical section. Hence value of X will depend on which process grabs the semaphore first. If it is Y then X is value is set Y. until the process leaves critical section. If Z grabs , the Value of X is Z until Z leaves the critical Section.

	Process A	PROCESS B
L1	INT Y;	INT Z;
L2	P(S)	P(S)
L3	X=Y;	X=Z;
L4	V(T)	X=Z
L5	V(S)	V(S)

c) Process start executing, x=5;

New semaphore T is introduced who's value is 0

Whoever grabs among process A, process B the control will get priority of execution.

Assume Process A starts executing First.

- For T=0,S=0
  As P(s)==0;
  Process will directly go into block state.
  Process B will start executing but as
  For process B, P(S)==0
  It will also be blocked.

- For T=0,S=1
  As P(s)==1;
  It will set X=Y;
  Check V(T)==0
  Check if any process is in block state
  Wake that process up and execute.
  If no process in block state set T.VAL=1;

V(S) =1
Got to next process B
Set X=Z
Check for block process if not set S.VAL=1.

Assume Process B is executing First

- For T=0,S=0
  As P(S)==0
  Process goes to block state ,
  Execute Process A .
- For T=0,S=1
  As P(S)==1
  SET X=Z
  Check V(S) for the block queue
  And based on the value execute the block process and set value.

As any process can run first there is no dependency among them, The value of X can be either, Y/Z. But if process goes in block state until its execution the value of X will be 5.

Reference
Question 2 : Book Modern Operating System reference page no 139