# COMP3500 – Lecture 18: Exam 1 – Exercises

**🟊: >85%, 🟊🟊: 70-85%, 🟊🟊🟊: 55-70%, 🟊🟊🟊🟊: 40-55%, 🟊🟊🟊🟊🟊: < 40%**

**🟊 Exercise 1 (Menti):** What is the execution sequence of Pi and Pj? (20 Seconds)

semaphore *flag* initialized to 0

A. Execute *Pi* only after *Pj is* executed

B. May execute *Pi* after *Pj is* executed

C. Execute *Pj* only after *Pi is* executed

D. After *Pj is* executed, *Pi* can start

Code:

*Proci Procj*

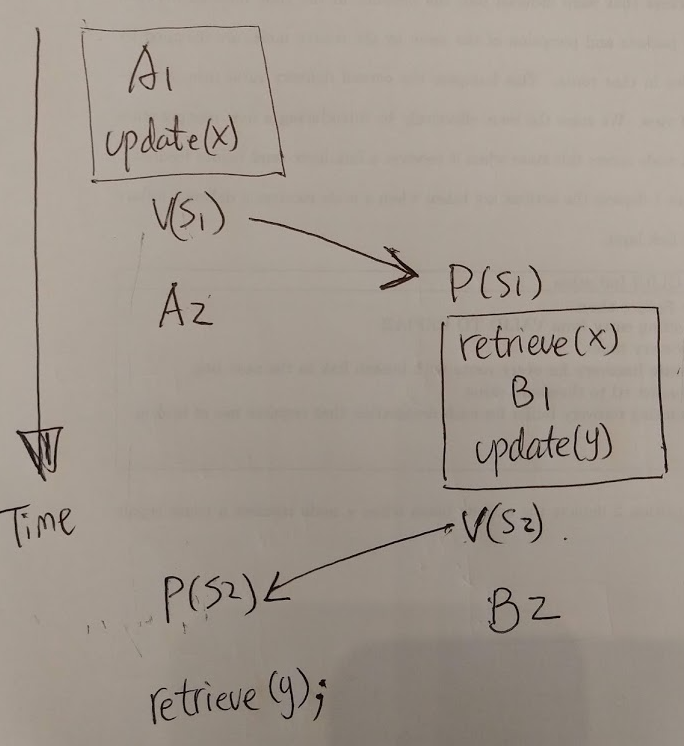
*Pi* *wait*(*flag*)

*signal*(*flag*) *Pj*

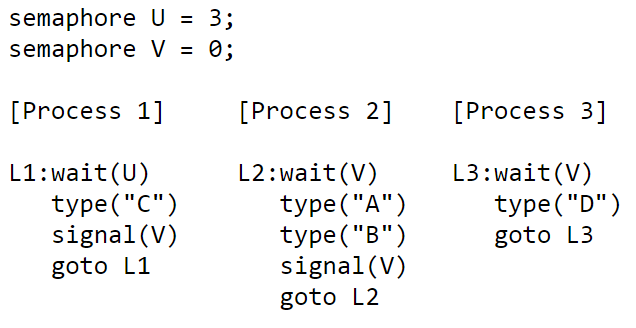
**Answer:** C.

**🟊🟊🟊 Exercise 2:** What is the execution sequence of proc\_A and proc\_B? (2 Minutes)

  
**Answer:**



**🟊🟊🟊 Exercise 3 (Menti):** There are three interacting processes. We assume that the execution is eventually halted. How many C's are printed when the set of processes runs? (2 Minutes)

****

A. Random Number

B. 1

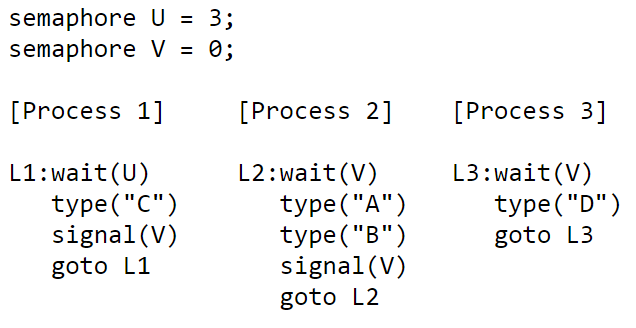
C. 2

D. 3

**Answer: D.** Within each process the statements are executed sequentially, but statements from different processes can be interleaved in any order that's consistent with the constraints imposed by the semaphores. When answering the questions below assume that once execution begins, the processes will be allowed to run until all 3 processes are stuck in a wait() statement, at which point execution is halted.

Exactly 3. Each time Process 1 executes the "wait(U)" statement, the value of semaphore U is decremented by 1. Since there are no "signal(U)" statements, the loop in Process 1 will execute only 3 times (ie, the initial value of U) and then stall the fourth time "wait(U)" is executed.

**🟊🟊🟊🟊🟊 Exercise 4 (Menti):** How many D's are printed when this set of processes runs? (1 Minute)

****

A. Random Number

B. 1

C. 2

D. 3

**Answer: D.** Exactly 3. Process 1 will execute its loop three times (see the answer to the previous question), incrementing "signal(V)" each time through the loop. This will permit "wait(V)" to complete three times. For every "wait(V)" Process 2 executes, it also executes a "signal(V)" so there is no net change in the value of semaphore V caused by Process 2. Process 3 does decrement the value of semaphore V, typing out "D" each time it does so. So Process 3 will eventually loop as many times as Process 1.

**🟊 Exercise 5 (Menti):** Given the above three concurrent processes,what is the smallest number of A's that might be printed when this set of processes runs? (30 Seconds)

**Answer: A.** 0. If Process 3 is scheduled immediately after Process 1 executes "signal(V)", then Process 2 might continue being stalled at its "wait(V)" statement and hence never execute its "type" statements.

A. 0

B. 1

C. 2

D. 3

**🟊🟊 Exercise 6 (Menti):** Is CABABDDCABCABD a possible output sequence when this set of processes runs? Why? (1 Minute)

A. Possible

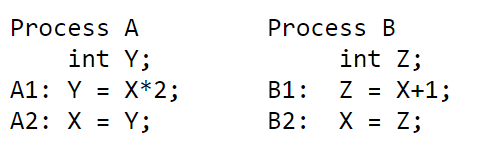
B. Impossible

C. The sequence always occurs

D. It depends on operating systems

**Answer: B.** No. Here are the events implied by the sequence above:start: U=3 V=0 type C: U=2 V=1 type A: U=2 V=0 type B: U=2 V=1 type A: U=2 V=0 type B: U=2 V=1 type D: U=2 V=0 type D: oops, impossible since V=0

**🟊 Exercise 7 (Menti):** Two processes share a common variable X. X is set to 5 before either process begins execution. Statements within a process are executed sequentially. What are different values of X are possible after both processes finish executing? (2 Minutes)

****

**Answer: D.**

There are four possible values for X. Here are the possible ways in which statements from A and B can be interleaved.

A1 A2 B1 B2: X = 11;

A1 B1 B2 A2: X = 10; B1 A1 B2 A2: X = 10;

A1 B1 A2 B2: X = 6; B1 A1 A2 B2: X = 6;

B1 B2 A1 A2: X = 12

A. 6, 10, 11

B. 6, 10, 11, 12, 14

C. 6, 10, 12

D. 6, 10, 11, 12

Use graphs

Y = X\*2

|

V

X= Y -🡪 Z= X+1

|

V

X= Z

Z = X + 1

|

V

X= Z -🡪 Y= X\*2

|

V

X= Y

Y = X\*2

|

V

X= Y

Z = X + 1

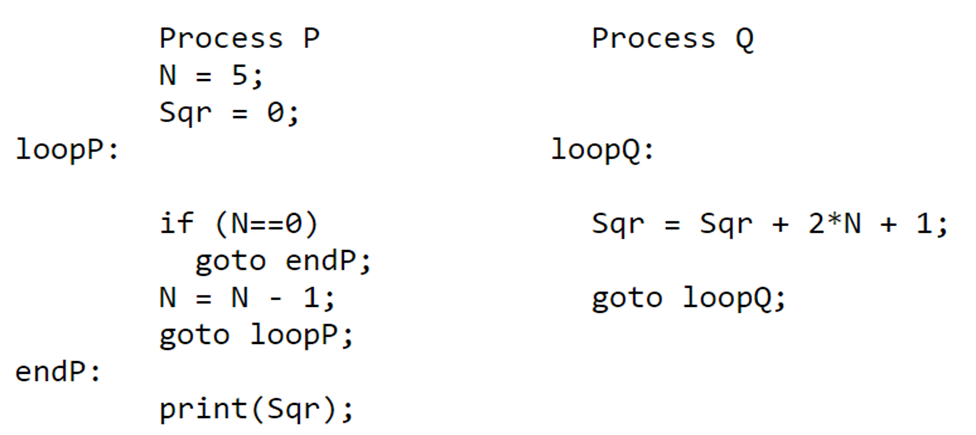
|

V

X= Z

**Exercise 8:** Two processes cooperate in computing the sum of the first N odd integers. Add appropriate semaphore declarations and signal and wait statements to these programs so that the proper value of Sqr (i.e., 25) will be printed out.

1+3+5+7+9=25 Process Q: Sqr=Sqr + 2\*N-1



Solution:

semaphore sqr\_add = 0, N\_minus = 1;

loopP: loopQ:

wait(sqr\_add);

if (N==0)

goto endP

N = N-1;

signal(N\_minus);

goto loopP;

endP:

print(Sqr);

Process Q

wait(N\_minus);

Sqr = Sqr + 2N + 1;

signal(sqr\_add);

goto loopQ;

**Another solution!**

loopP: loopQ:

if (N==0)

goto endP

wait(sqr\_add);

N = N-1;

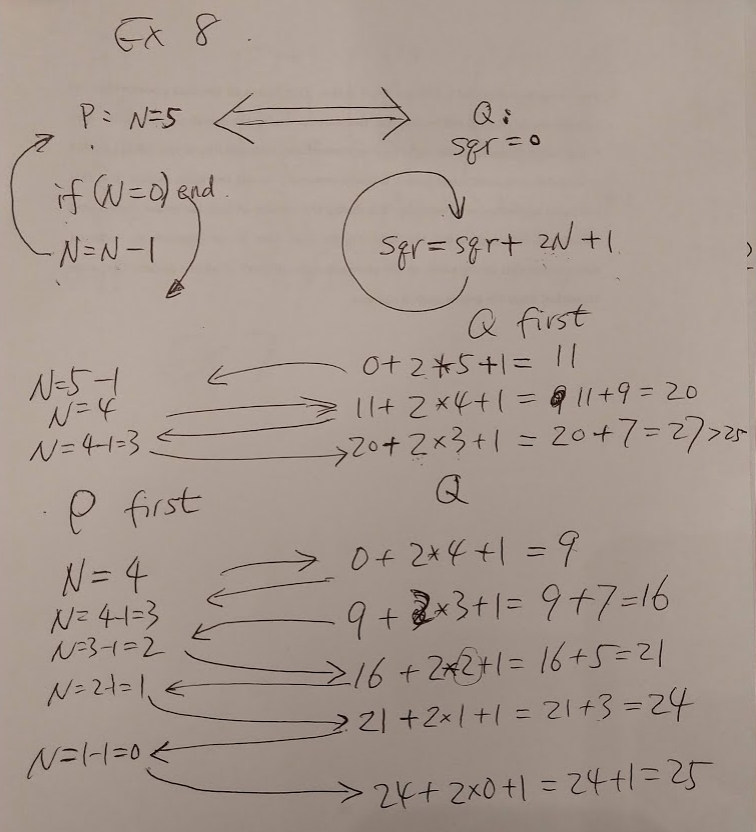
signal(N\_minus);

goto loopP;

endP:

wait(sqr\_add);

print(Sqr);



**Exercise 9 (Optional):** The Cigarette Smoker's Problem. There are three smokers and one cigarette dealer. A smoker must make a cigarette before he or she can smoke it. Making a cigarette requires (1) tobacco, (2) paper, and (3) matches. Each smoker has one of the three items. Specifically, one smoker has tobacco, another has paper, and a third one has matches. The cigarette dealer has an infinite supply of tobacco, paper, and matches. The dealer randomly selects two of the three items and places them on a table. Then, the smoker who has the third item makes a cigarette and smoke. After the first smoker finishes smoking, the dealer will place other two of the three items on the table so that another smoker can enjoy cigarettes. Please use semaphores to synchronize the three smokers and the dealer.

**Solution:**

Dealer

1 do forever {

2 wait( lock );

3 randNum = rand( 1, 3 ); // Pick a random number from 1-3

4 if ( randNum == 1 ) {

5 // Put tobacco on table

6 // Put paper on table

7 signal( smoker\_match ); // Wake up smoker with match

8 } else if ( randNum == 2 ) {

9 // Put tobacco on table

10 // Put match on table

11 signal ( smoker\_paper ); // Wake up smoker with paper

12 } else {

13 // Put match on table

14 // Put paper on table

15 signal ( smoker\_tobacco ); } // Wake up smoker with tobacco

16 signal( lock );

17 wait ( smoker\_done ); // Agent sleeps

18 } // end forever loop

One of the smokers. The others are similar.

1 do forever {

2 wait ( smoker\_tobacco ); // Sleep right away

3 wait ( lock );

4 // Pick up match

5 // Pick up paper

6 signal ( smoker\_done );

7 signal ( lock );

8 // Smoke (but don't inhale).

9 }