Process synchronization

Week 3 - Tutorial

Exercises 1

- A counting semaphore **S** is initialized to **10**.
- Then, 6 **wait()** operations and 4 **signal()** operations are performed on **S**.
- What is the **final value** of S?

- **P** operation also called as **wait** operation decrements the value of semaphore variable by 1.
- **V** operation also called as **signal** operation increments the value of semaphore variable by 1.

Hints..

- **Wait()** operation **decrements** semaphore value; if value negative, process is blocked
- **Signal()** operation **increments** semaphore value; one of the blocked processes is unblocked

Exercises 2

The following program consists of **3** concurrent processes and 3 binary semaphores. The semaphores are initialized as **S0 = 1, S1 = 0** and **S2 = 0**.

What is the **maximum number** of times process P0 can print '**0**'?

a. At least twice
b. Exactly twice
c. Exactly thrice
d. Exactly once

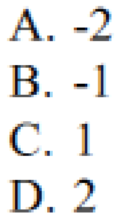
Process P0	Process P1	Process P2
while (true)	wait (S1);	wait (S2);
{	signal (S0);	signal (S0);
wait (S0);		
print '0'		
signal (S1);		
signal (S2);		
}		

Hints..

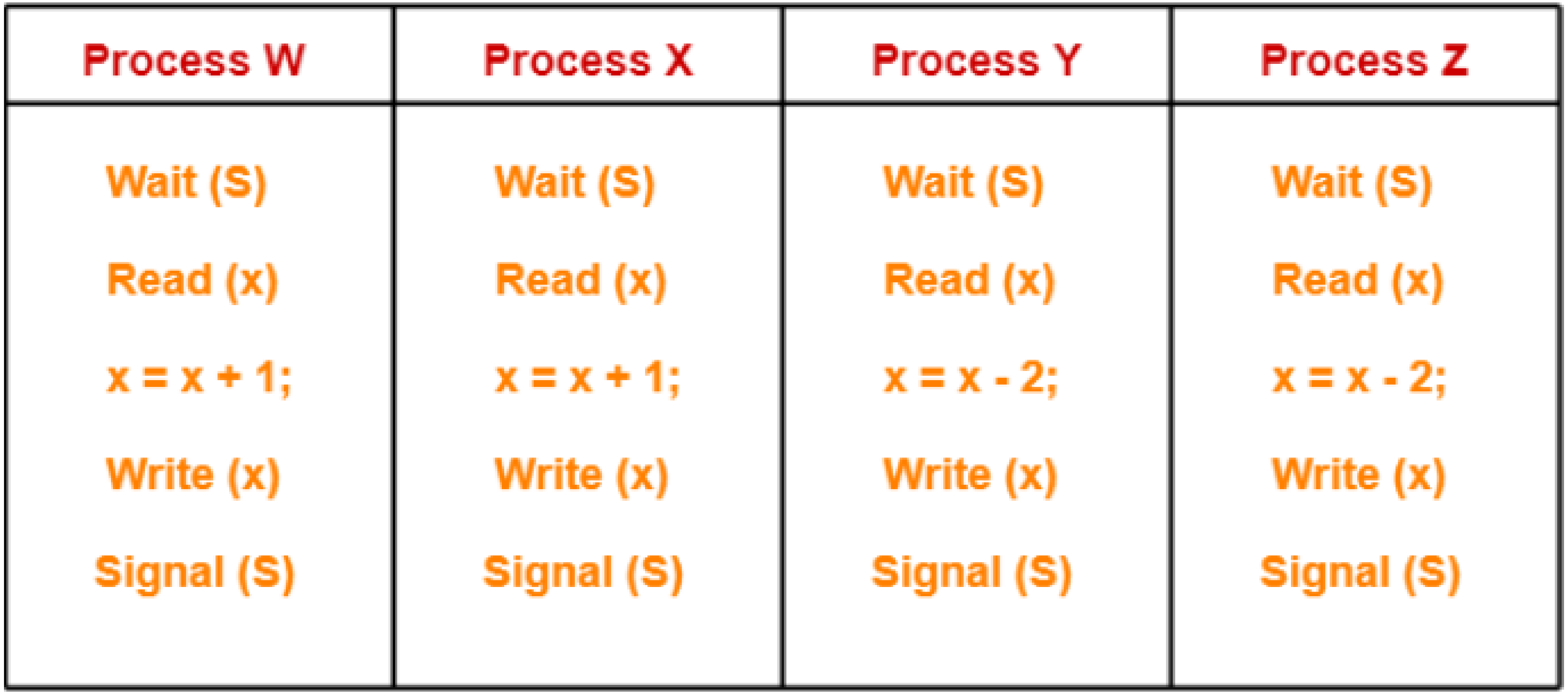
- **wait()** operation **decrements** semaphore value; *if value negative, process is blocked*
- **signal()** operation **increments** semaphore value; *one of the blocked processes is unblocked*

Exercises 3

- A shared variable **x**, **initialized to zero**, is operated on by four concurrent **processes W, X, Y, Z** as follows.
- Each of the processes **W** and **X** reads **x** from memory, increments by one, stores it to memory and then terminates.
- Each of the processes **Y** and **Z** reads **x** from memory, decrements by two, stores it to memory, and then terminates.
- Each process before reading **x** invokes the **wait()** operation and invokes the **signal()** operation after storing **x** to memory.
- Semaphore **S** is initialized to **two**.
- Find the **maximum** possible value of **x** after **all process complete execution**.



A. -2
B. -1
C. 1
D. 2



Process W	Process X	Process Y	Process Z
Wait (S) Read (x) x = x + 1; Write (x) Signal (S)	Wait (S) Read (x) x = x + 1; Write (x) Signal (S)	Wait (S) Read (x) x = x - 2; Write (x) Signal (S)	Wait (S) Read (x) x = x - 2; Write (x) Signal (S)

Hints..

- **wait()** operation **decrements** semaphore value; if value negative, process is blocked

- **signal()** operation **increments** semaphore value; one of the blocked processes is unblocked