1. Part of the program where a shared resource is accessed is _____________ .

   Critical section

2. The key difference between a binary semaphore and a mutex is that mutex can be owned by two tasks, whereas a binary semaphore can be owned only by one task.

   False

3. A binary semaphore provides the same functionality as a mutex.

   False

4. Mutual Exclusion is necessary to prevent race conditions.

   True

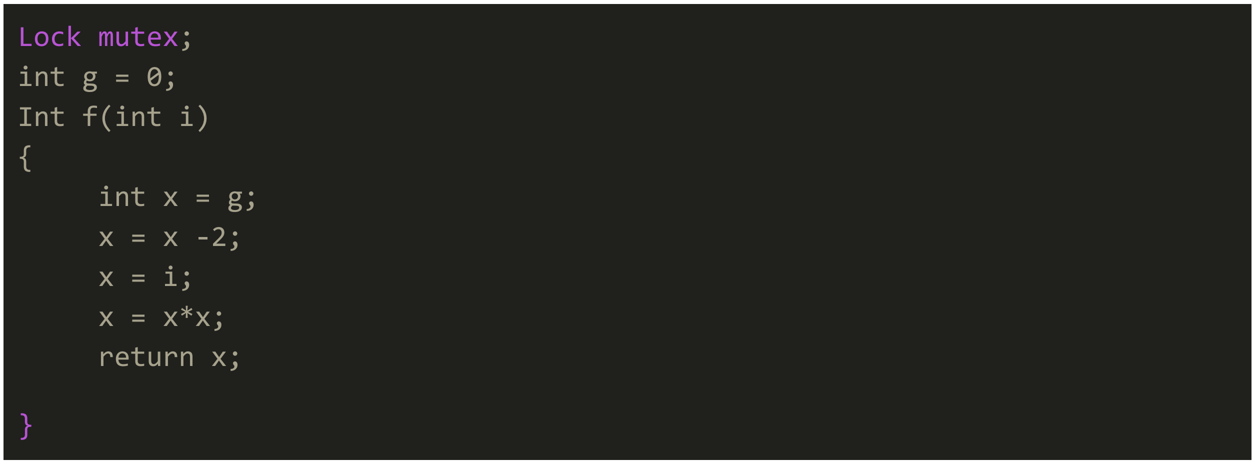
5. In general, with N processes sharing N semaphores, the potential for deadlock decreases as N grows larger.

   False

6. A counting semaphore is initialized to 15. 8 wait operations and 7 signal operations were completed on this semaphore. What is the resulting value of the semaphore?

   14

7. Is the function f() thread-safe? Is f() reentrant?

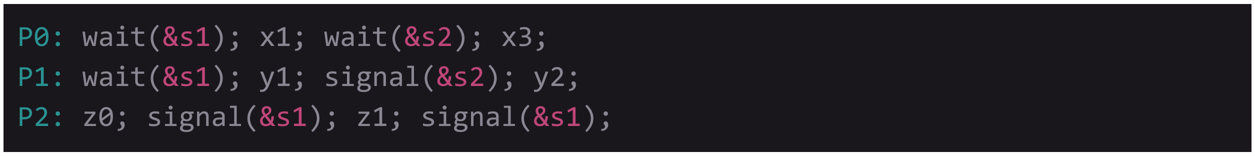


```
Lock mutex;
int g = 0;
Int f(int i)
{
     int x = g;
     x = x -2;
     x = i;
     x = x*x;
     return x;

}
```

Thread-safe, thread-reentrant

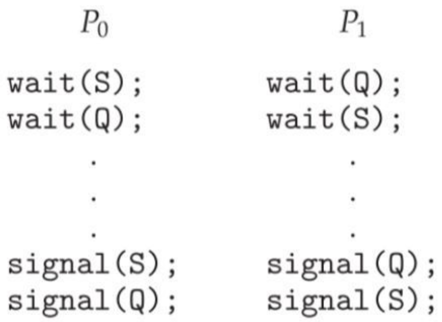
8. Which of the following conditions can you guarantee from the three processes below? (Assue s1 and s2 are semaphores initialized to 0)



```
P0: wait(&s1); x1; wait(&s2); x3;
P1: wait(&s1); y1; signal(&s2); y2;
P2: z0; signal(&s1); z1; signal(&s1);
```

Y1 executes before x3

9. Below are two processes using the semaphores initialized as Q=1 and S=1. Which of the following is true regarding the execution of these processes?



```
     P0                P1

wait(S);          wait(Q);
wait(Q);          wait(S);
    .                 .
    .                 .
    .                 .
signal(S);        signal(Q);
signal(Q);        signal(S);
```

P0 is subject to starvation
P1 is subject to starvation
P0 and P1 can result in deadlock