Assignment 1

Kartikeya Jaiswal - EE18BTECH11025

Download all latex-tikz codes from

https://github.com/kartikeyajaiswal/c−ds

1 Problem

(Q 34) Each of a set of n processes executes the following code using two semaphores a and b initialized to 1 and 0, respectively. Assume that *count* is a shared variable initialized to 0 and not used in CODE SECTION P.

```
CODE SECTION P

Wait (a); count = count + 1;
if (count ==n) signal (b);
signal (a); wait (b); signal (b);

CODE SECTION Q
```

What does the code achieve?

1) It ensures that no process executes CODE SECTION Q before every process has finished CODE SECTION P.
2) It ensures that at most two processes are in CODE SECTION Q at any time.
3) It ensures that all processes execute CODE SECTION P mutually exclusively.
4) It ensures that at most n–1 processes are in CODE SECTION P at any time.

2 Solution

Answer : (A) It ensures that no process executes CODE SECTION Q before every process has finished CODE SECTION P.

Explanation
The definition of wait() function:

```
wait(Semaphore a)
{
    while(a<=0){
        ; //no operation
    }
    a−−;
}
```

and the definition of signal() function:

```
signal(Semaphore a)
{
    a++;
}
```

Consider process 1 (say P1):

```
n;
count = 0;
a = 1;
b = 0;

CODE SECTION P

Wait (a); //a=0
count = count + 1; //count = 1
if (count ==n) signal (b); //since count!=n, b=0
signal (a); // a=1
wait (b); // no operation
signal (b);

CODE SECTION Q
```

In the last section, wait(b) doesn't execute, since wait() does no operation for semaphore = 0. This process P1 is stuck here, but *count* is updated to count=1.

Now, similarly consider process P2:

```
n;
count = 1;
```

```
CODE SECTION P

Wait (a); //a=0
count = count + 1; //count = 2
if (count ==n) signal (b); //since count!=n, b=0
signal (a); // a=1
wait (b); // no operation
signal (b);

CODE SECTION Q
```

Here as well, wait(b) doesn't execute, since wait() does no operation for semaphore = 0. This process P1 is stuck here, but *count* is updated to count=2.

Clearly, for each process executed, the *count* increases by 1. So, at the end of $(n-1)^{th}$ process, *count* = n-1. Consider the n^{th} process:

```
n;
count = n−1;

CODE SECTION P

Wait (a); //a=0
count = count + 1; //count = n
if (count ==n) signal (b); //since count=n, b=1
signal (a); // a=1
wait (b); // b=0
signal (b); // b=1

CODE SECTION Q
```

Here, the wait(b) at the end executes (since count=n), so the n^{th} process completes here and its not stuck. Here signal(b) completes execution changing value of b to b=1, so all the previously stuck processes will also complete execution.

So, option (A) is correct.