PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS

2da práctica (tipo a)
(Primer semestre de 2018)

Horario 0781: prof. V. Khlebnikov
Horario 0782: prof. A. Bello

Duración: 1 h. 50 min.
Nota: No se puede usar ningún material de consulta.
La presentación, la ortografía y la gramática influirán en la calificación.
Puntaje total: 20 puntos

Pregunta 1 (3 puntos – 15 min.) Considere el siguiente algoritmo (donde p_i y q_i son atómicos):

integer n ← 1	
p	**q**
p1: while n < 1 p2: n ← n + 1	q1: while n ≥ 0 q2: n ← n - 1

a) (1 punto – 5 min.) Construya un escenario en el que el lazo en *p* se ejecuta exactamente una sola vez.

b) (1 punto – 5 min.) Construya un escenario en el que el lazo en *p* se ejecuta exactamente una tres veces.

c) (1 punto – 5 min.) Construya un escenario en el que ambos lazos se ejecutan de forma infinita.

Pregunta 2 (4 puntos – 20 min.) A continuación se muestra el algoritmo de Peterson:

```
                    var    flag : array [0 .. 1] of boolean;
                    turn : 0 .. 1;
begin
       flag[0] := false;
       flag[1] := false;
   Parbegin
       repeat                                          repeat
           flag[0] := true;                                flag[1] := true;
           turn := 1;                                      turn := 0;
           while flag[1] and turn = 1                      while flag[0] and turn = 0
              do {nothing};                                   do {nothing};
           { Critical Section }                            { Critical Section }
           flag[0] :=false;                                flag[1] :=false;
           { Remainder of the cycle }                      { Remainder of the cycle }
       forever                                         forever
   Parend
   {P0}                                            {P1}
end.
```

a) (2 puntos) Se hacen los siguientes cambios en el algoritmo de Peterson: la instrucción `flag[0] := true` se cambia por `flag[0] := false` en P0, y de forma análoga se hace el cambio en P1. ¿Qué propiedades de las secciones críticas no se cumplen?

b) (2 puntos) La instrucción `while flag[1] and turn = 1` en P0 del algoritmo de Peterson, se cambia por `while flag[1] or turn = 1`, de forma análoga para el P1.¿Qué propiedades de las secciones críticas no se cumplen?

Pregunta 3 (2 puntos – 10 min.) Dados los siguientes procesos y sus respectivas secuencias de código, indique si existe o no situaciones de interbloqueo y expliqué por qué. En cualquier caso, también indique la salida por pantalla y el valore final de los semáforos. Suponga que inicialmente todos los semáforos tienen valor cero.

```
Proceso 1              Proceso 2              Proceso 3
---------              ---------              ---------
printf("3");           sem_wait(&s1);         sem_wait(&s2);
sem_post(&s3);         printf("1");           sem_wait(&s4);
printf("4");           sem_wait(&s3);         printf("2");
sem_post(&s2);         sem_post(&s4);         printf("5");
sem_post(&s1);         sem_wait(&s3);         sem_post(&s3);
```

Pregunta 4 (3 puntos – 15 min.) Considere que los siguientes fragmentos de código se ejecutan en paralelo:

```
Código A:                   Código B:
---------                   ---------
printf("A1");               printf("B1");
sem_post(&s1);              sem_wait(&s1);
sem_wait(&s2);              printf("B2");
printf("A2");               sem_post(&s3);
sem_wait(&s2);              sem_wait(&s3);
sem_post(&s1);              printf("B3");
printf("A3");               sem_post(&s2);
                            sem_wait(&s1);
                            sem_post(&s2);
                            printf("B4");
```

Sabiendo que todos los semáforos están inicializados en 0, indique todas las posibles salidas que puede proporcionar su ejecución y si ocurre interbloqueo para cada una de ellas.

Pregunta 5 (5 puntos – 25 min.) Observe el siguiente fragmento de código donde los semáforos *sem1* y *sem2* están inicializados a cero, un hilo ejecuta la función incrementa y otro la función decrementa. Describa los valores que, durante la ejecución, puede adoptar la variable *num* así como las posibles situaciones de interbloqueo que pudiera darse.

```
int num=10;

void * incrementa(void *nada) {
       int i;
       for (i=0;i<3;i++){
              sem wait(&sem1);
              num++;
              printf("Inc. Número = %d\n",num);
              sem post(&sem1);
       }
       sem post(&sem2);
       sleep(random() %3);
       sem wait(&sem2);
       pthread exit(NULL);
}

void * decrementa(void *nada){
       int i;
       for (i=0;i<3;i++){
              sem post(&sem1);
              sleep(random() %3);
              sem wait(&sem2);
              num--;
              printf("Dec. Número = %d\n",num);
              sem post(&sem2);
              sem wait(&sem1);
       }
       sem wait(&sem1);
       pthread exit(NULL);
}
```

Pregunta 6 (3 puntos – 15 min.) Resuelve mediante monitores la sincronización entre tres procesos, o lo que es lo mismo, diseñar un monitor con un único procedimiento de nombre `seguir`, que provoque que los dos primeros procesos que llamen a ese procedimiento se suspendan, y el tercero lo despierte, y así cíclicamente.

La práctica ha sido preparada por AB
con LibreOffice Writer en Debian 9.4 Stretch.

Profesores del curso: (0781) V. Khlebnikov
(0782) A. Bello R.

Pando, 19 de abril de 2018