PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS
2da práctica (tipo a)
(Segundo semestre de 2011)

Horario 0781: prof. V. Khlebnikov
Horario 0782: prof. F. Solari A.

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) ***(Busy Waiting)*** **El algoritmo visto como "cuarto intento" en clase:**

```
volatile int c1=1, c2=1;  /* ninguno en su SC */

/* proceso 1 */
 ...
 while(TRUE){
    c1 = 0;
    while ( c2 == 0 ) {
       c1 = 1;
       /*   …   */
       c1 = 0;
    }
    sc1();
    c1 = 1;
    snc1();
 }
```

a) (1 punto) ¿Cuál es el código para el proceso 2?

b) (1 punto) ¿Qué representa el código comentado como /* … */ ?

c) (1 punto) Este algoritmo no es completamente seguro. ¿Qué condiciones de una buena solución pueden ser incumplidas?

Pregunta 2 (3 puntos) ***(Busy Waiting with Hardware Instructions help)*** **Los procesadores Intel, de tipo IA-32 (antigua iAPX86, conocidos también como x86 y x86_64), no tienen exactamente una instrucción TSL, pero si una instrucción XCHG que permite intercambiar el contenido de un registro con una posición de memoria. Entonces, se plantea el siguiente código para una función** ***void AcquireLock(int *lock);*** **y la respectiva** ***void ReleaseLock(int *lock);***

```
void AcquireLock (int *lock) {
   asm {
    L1: MOV   eax, #00000001h  ; colocar valor 1 en el registro 'eax' de 32 bits del CPU
        XCHG eax, [lock]       ; intercambiar contenido del registro 'eax' con contenido de variable apuntada por lock
        CMP   eax, #0h         ; comparar contenido del registro 'eax' con valor 0
        JNE   L1               ; saltar a L1 si no fue igual a 0
        RET                    ; retorno de la función
   }
}
```

a) (1 punto) Escriba el código para ***void ReleaseLock(int *lock)*** **explicando el significado.**

b) (2 puntos) Explique qué ocurre cuando dos procesos o hilos que comparten el acceso a ***lock*** **invocan AcquireLock() concurrentemente, mediante alguna secuencia que evidencie si se cumple el propósito de solución o no.**

Pregunta 3 (8 puntos) (*W. Stallings, Operating systems: internals and design principles, 5E*) **An Incorrect Solution to the Infinite-Buffer Producer/Consumer Problem Using Binary Semaphores:**

```
/* program producerconsumer */
int n;
binary_semaphore s = 1;
binary_semaphore delay = 0;

void producer()
{
    while (true)
    {
        produce();
        semWaitB(s);
        append();
        n++;
        if (n==1)
            semSignalB(delay);
        semSignalB(s);
    }
}

void main()
{
    n = 0;
    parbegin(producer,consumer);
}
```

```
void consumer()
{
    semWaitB(delay);
    while (true)
    {
        semWaitB(s);
        take();
        n--;
        semSignalB(s);
        consume();
        if (n==0)
            semWaitB(delay);
    }
}
```

a) (3 puntos) Indique las secciones críticas de cada proceso (1 punto) y explique el funcionamiento de cada proceso sin usar los nombres de sentencias del lenguaje de programación.

Considere el siguiente guión de una ejecución posible:

	producer	consumer	s	n	delay
1			1	0	0
2	semWaitB(s)		0	0	0
3	n++		0	1	0
4	if (n == 1) semSignalB(delay)		0	1	1
5	semSignalB(s)		1	1	1
6		semWaitB(delay)	1	1	0
7		semWaitB(s)	0	1	0
8		n--	0	0	0
9		semSignalB(s)	1	0	0
10	semWaitB(s)		0	0	0
11	n++		0	1	0
12	if (n == 1) semSignalB(delay)		0	1	1
13	semSignalB(s)		1	1	1
14		if (n == 0) semWaitB(delay)	1	1	1
15		semWaitB(s)	0	1	1
16		n--	0	0	1
17		semSignalB(s)	1	0	1
18		if (n == 0) semWaitB(delay)	1	0	0
19		semWaitB(s)	?	?	?
20		?	?	?	?
21		?	?	?	?

b) (2 puntos) Complete las 3 últimas líneas del guión y explique qué significa el resultado obtenido en la línea 20.

c) (2 puntos) Para evitar el error se puede mover la sentencia condicional del consumidor adentro de su sección crítica, pero esto va a provocar otro problema serio. ¿Cuál y cómo?

d) (1 punto) Para obtener la solución correcta se puede introducir una variable auxiliar en el código del consumidor. Complete el siguiente código propuesto:

```
void consumer()
{
    int m; /* a local variable */
    semWaitB(delay);
    while (true)
    {
        semWaitB(s);
        take();
        n--;
        m = n;
        ...
    }
}
```

<u>Pregunta 4</u> (3 puntos) *(Monitores de Hoare y variables de condición)*. Considerando un lenguaje que implemente monitores y variables de condición, para plantear una solución al problema clásico de filósofos:

a) (2 puntos) ¿Qué funciones deberían ser definidas en la sección *monitor*? Haga un esbozo de una función Filosofo(i) que invoque dichas funciones.

b) (1 punto) ¿Cómo debería bloquearse un filósofo al no conseguir ambos tenedores?

<u>Pregunta 5</u> (3 puntos) (*W. Stallings, Operating systems: internals and design principles, 5E*) A Solution to the Bounded-Buffer Producer/Consumer Problem Using Messages:

```
const int
    capacity = /* buffering capacity */;
    null = /* empty message */;
int i;

void producer()
{   message pmsg;
    while (true)
    {
        receive(mayproduce, pmsg);
        pmsg = produce();
        send(mayconsume,pmsg);
    }
}
void main()
{
    create_mailbox(mayproduce);
    create_mailbox(mayconsume);
    for (int i = 1; i <= capacity; i++)
        send(mayproduce,null);
    parbegin(producer,consumer);
}
```

```
void consumer()
{   message cmsg;
    while (true)
    {
        ...
        ...
        ...
    }
}
```

Complete el código faltante.

------- o -------

La preguntas han sido preparadas por VK (3,5) y FS (1,2,4).

Profesores del curso: (0781) V. Khlebnikov,
(0782) F. Solari A.

Pando, 21 de septiembre de 2011