CC41B : Sistemas Operativos
Control 1–Semestre Primavera'97
Prof.: Luis Mateu.

Pregunta 1

Se tiene la siguiente solución al problema de los lectores/escritores:

```
void EnterRead()                 void ExitRead()
{                                {
  if (count==0)                    Wait(count_sem);
    Wait(resource_sem);            count--;
  Wait(count_sem);                 Signal(count_sem);
  count++;                         if (count==0)
  Signal(count_sem);                 Signal(resource_sem);
}                                }

void EnterWrite()                void ExitWrite()
{                                {
  Wait(resource_sem);              Signal(resource_sem);
}                                }
```

En donde `count`, `resource_sem` y `count_sem` son variables globales que requieren las siguientes inicializaciones:

```
count= 0;
resource_sem= MakeSem(1);
count_sem= MakeSem(1);
```

Conteste las siguientes preguntas:

1. ¿Para qué tipo de procesos está diseñada esta solución? ¿Procesos pesados o procesos livianos? Explique.
2. Haga un diagrama de progreso de procesos que muestre que con esta solución, un escritor puede entrar junto con un lector.
3. Indique si este error se podría manifestar en un monoprocesador con procesos non-preemptive (sin adelantamiento). Explique.
4. Modifique mínimamente esta solución para que funcione correctamente (no es necesario evitar hambruna).

Pregunta 2

Un grupo de estudiantes pasan sus días estudiando y divirtiéndose en la montaña rusa de Fantasilandia. La montaña rusa tiene asientos para N estudiantes. Cada cierto tiempo la montaña se detiene, los estudiantes se bajan y luego suben otros estudiantes. Si hay a los menos N estudiantes esperando, la montaña se pone en funcionamiento de inmediato. Si no, se espera a que lleguen más estudiantes por un máximo de T unidades de tiempo. La montaña se echa a andar en cuanto estén ocupados los N asientos o cuando haya transcurrido el tiempo T desde el inicio de la detención. Los estudiantes ocupan los asientos en orden de llegada.

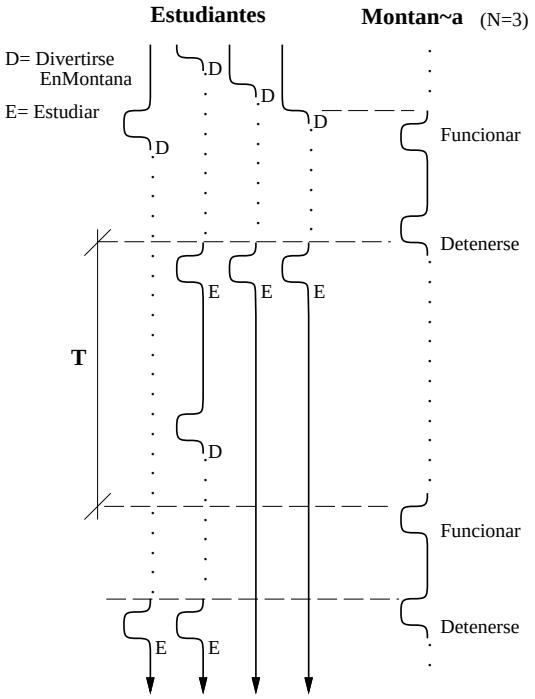
La montaña y los estudiantes son modelados por tareas de nSystem que ejecutan los siguientes procedimientos:

```
MontanaRusa()                    Estudiante()
{                                {
  for(;;)                          for(;;)
```

```
  {                          {
    Detenerse();               DivertirseEnMontana();
    Funcionar();               Estudiar();
} }                        } }
```

Escriba los procedimientos `Detenerse` y `DivertirseEnMontana`.

La siguiente figura muestra un diagrama de progreso de procesos con la sincronización que Ud. debe implementar:



Hint: No pierda tiempo pensando una solución con semáforos o monitores.

Procedimientos de nSystem:

```
int nSend(nTask task, void *msg); /* Envia un mensaje a una tarea */
void *nReceive(nTask *ptask, int max_delay);
                                  /* Recepcion de un mensaje */
void nReply(nTask task, int rc);  /* Responde un mensaje */

int nGetTime(); /* Entrega la hora en milisegundos */

Queue MakeQueue();                      /* El constructor */
void PutTask(Queue queue, nTask task);  /* Agrega una tarea al final */
nTask GetTask(Queue queue);             /* Extrae la primera tarea */
int EmptyQueue(Queue queue);            /* Verdadero si la cola esta vacia */
```