Examen de Lenguajes de Alto Nivel

Septiembre 2003

Cuestiones (4 cuestiones, 4 puntos en total; contestar brevemente de forma razonada)

1) El siguiente procedimiento usa variables globales para comunicarse con sus subprogramas internos. Reescribirlo para que no use variables globales. Para ello, se deben declarar las variables en un lugar inaccesible a los subprogramas internos, y pasarles a éstos los datos por parámetros. Pensar cuidadosamente si los parámetros deben ser de entrada, salida, o entrada/salida.

```
with Ada.Text_IO, Ada.Integer_Text_IO;
use Ada.Text_IO, Ada.Integer_Text_IO;

procedure Muestra is
   A,B : Integer;
   X1, S : Integer;

   procedure Dist is
   begin
      X1:=A*A+B*B;
   end Dist;

   procedure Suma is
   begin
      S:=A+B+X1;
   end Suma;
begin
   Put("A:"); Get(A); Skip_Line;
   Put("B:"); Get(B); Skip_Line;
   Dist;
   Suma;
   Put_Line("X1="&Integer'Image(X1));
   Put_Line("S ="&Integer'Image(S));
end Muestra;
```

2) Crear un programa con dos tareas. Una debe activarse cada 2 segundos aproximadamente, y poner un mensaje en la pantalla cada vez. La otra debe activarse una sola vez, transcurrida una hora desde el comienzo del programa, y poner un solo mensaje en pantalla. El programa principal no hace nada.

3) Se dispone de un procedimiento compilado separadamente, `P1`, que obedece a la especificación de abajo. Escribir otro procedimiento que lo invoque sucesivamente para valores de `i`=3, 6, 12, ..., duplicando el valor a cada llamada, hasta que `P1` eleve la excepción `Constraint_Error`. En ese momento el nuevo procedimiento terminará de manera normal (es decir, sin elevar a su vez la excepción).

```
procedure P1(I : Integer);
```

4) Escribir un programa que lea números enteros de un fichero de texto y calcule su valor medio, mostrando este dato en pantalla al finalizar. El fichero contiene un número indeterminado de números enteros, uno por línea. El nombre del fichero es `datos.txt`.

Examen de Lenguajes de Alto Nivel

Septiembre 2003

Problema (6 puntos)

Se desea escribir parte del software de control de una planta de envasado en la que se transportan objetos entre varias máquinas, por medio de cintas de transporte. Para cada cinta de transporte se dispone de un sensor que indica si existe un objeto a su entrada, y otro sensor que indica si se detecta un objeto a su salida. Para cada cinta hay que llevar la cuenta del número de objetos que contiene. La cinta se conecta al detectar un objeto a la entrada, y se desconecta al cabo de dos segundos de salir el último objeto. Su velocidad se regula a dos posibles valores según el número de objetos contenidos en ella: rápida si es menor que cinco, o lenta si no. El número de cintas de transporte es variable, con un máximo de 20.

Para controlar las cintas se dispone de un paquete ya realizado con la siguiente especificación, en la que la función de cada subprograma está explicada mediante comentarios:

```
package Cinta is

   Max_Cintas : constant Integer :=20;

   subtype Num_Cinta is Integer range 1..Max_Cintas;

   type Estado_Detector is (Nada, Hay_Objeto);

   procedure Apaga (N : Num_Cinta);
   -- Apaga la cinta indicada por N

   procedure Enciende (N : Num_Cinta; Rapida : Boolean);
   -- Enciende la cinta indicada por N, a velocidad lenta si
   -- Rapida vale False, y rapida en caso contrario

   procedure Pon_Rapida (N : Num_Cinta);
   -- Pone la cinta indicada por N a velocidad rapida,
   -- si estaba encendida. Si estaba apagada, eleva
   -- Estado_Incorrecto

   procedure Pon_Lenta (N : Num_Cinta);
   -- Pone la cinta indicada por N a velocidad lenta,
   -- si estaba encendida. Si estaba apagada, eleva
   -- Estado_Incorrecto

   function Detector_Entrada (N : Num_Cinta)
                              return Estado_Detector;
   -- Devuelve el valor leido por el detector de entrada de
   -- la cinta indicada por N. Nada si no detecta objeto, y
   -- Hay_Objeto si está detectando uno. Además, eleva
   -- Esta_Atascada si observa que la cinta está atascada.

   function Detector_Salida (N : Num_Cinta)
                             return Estado_Detector;
   -- Lo mismo que la anterior, pero para el detector de salida

   Estado_Incorrecto, Esta_Atascada : exception;

end Cinta;
```

El sistema dispone de una pantalla gráfica donde se representa el estado de cada cinta; para dibujar en esta pantalla se dispone del siguiente paquete, ya realizado:

```
with Cinta;

package Pantalla is

   type Estado_Cinta is (Inexistente, Atascada, Apagada,
                         Lenta, Rapida);

   -- Los valores de este tipo indican el estado de la cinta.
   -- Si vale inexistente, no se pinta su estado en pantalla

   procedure Pon_Estado_Cinta
     (N : Cinta.Num_Cinta;
      E : Estado_Cinta;
      Num_Objetos : Natural);
   -- Muestra en una pantalla grafica el estado de la cinta N
   -- (representado por E), y el numero de objetos indicado
   -- por Num_Objetos

   procedure Pon_Mensaje (S : String);
   -- Muestra un mensaje de texto en la zona inferior de
   -- la pantalla

end Pantalla;
```

La especificación del paquete de gestión de cintas será:

```
with Cinta;

package Gestion_Cintas is

   procedure Anade_Cinta;
   -- Inserta una cinta nueva, poniendola inicialmente apagada
   -- y con cero objetos, y representándola en la pantalla
   -- Si el numero de cintas fuera a exceder el maximo, se eleva
   -- Demasiadas

   function Num_Cintas return Natural;
   -- Retorna el numero de cintas actual

   procedure Gestiona;
   -- Gestiona todas las cintas del robot.

   Demasiadas : exception;
   Error_Num_Objetos : exception;

end Gestion_Cintas;
```

Se pide escribir el cuerpo de este paquete. Debe actuar como una máquina de estados abstracta, que almacene los siguientes datos:

- el número de cintas (inicialmente cero)
- los datos de cada cinta: su estado, el número de objetos, el estado último de los detectores de entrada y salida, y el tiempo que transcurrió desde que salió el último objeto.

Los procedimientos `Anade_Cinta` y `Num_Cintas` son muy simples y se describen en los comentarios. El procedimiento `Gestiona` debe hacer los siguiente, para cada cinta:

- Si la cinta es inexistente, poner su estado en la pantalla.
- Si la cinta existe, debe hacer lo siguiente:
  - Detección de entrada: si el valor anterior del detector de entrada era `Nada` y el actual es `Hay_Objeto`, incrementar el número de objetos en la cinta. Si al valor anterior era cero, encender la cinta a velocidad rápida. Luego, actualizar el valor anterior almacenado para ese detector.
  - Detección de salida: si el valor anterior del detector de salida era `Hay_Objeto` y el actual es `Nada`, decrementar el número de objetos de la cinta y anotar el instante de salida del objeto (igual a la hora actual). Si el número de objetos de la cinta quedase por error negativo, ponerlo a cero y elevar `Error_Num_Objetos`. Finalmente, actualizar el valor anterior almacenado para ese detector.
  - Si el número de objetos de la cinta está entre 1 y 4, poner la cinta a velocidad rápida. Si es mayor que 4, ponerla a velocidad lenta.
  - Si el número de objetos de la cinta es cero y la diferencia entre la hora actual y el instante de salida del último objeto es menor que 2 segundos, poner la cinta a velocidad rápida; si es mayor o igual a 2 segundos, apagar la cinta.
  - Representar en pantalla el estado actual de la cinta.
  - Si se elevase `Estado_Incorrecto` poner un mensaje en pantalla (con `Pon_Mensaje`) que indique el error y el número de cinta; luego continuar con la siguiente cinta.
  - Si se elevase `Esta_Atascada`, pasar el estado de la cinta a `Atascada`, representar su estado en pantalla, y seguir con la siguiente cinta.