Examen de Lenguajes de Alto Nivel

Febrero 2006

1) (2 puntos)

Escribir el cuerpo del paquete cuya especificación se muestra debajo, de modo que el procedimiento `Copiar` cree una copia exacta del fichero binario cuyo nombre es `Origen` en el fichero con nombre `Destino`. El tipo de fichero que se maneja viene dado en el paquete privado `Fichero_IO`. El procedimiento `Copiar` debe lanzar la excepción `Error_Al_Copiar` en caso de que se produzca cualquier error en la manipulación de los ficheros.

```
with Ada.Sequential_IO;

generic
   type Elemento is private;
package Copias is

   procedure Copiar (Origen  : String;
                     Destino : String)

   Error_Al_Copiar : exception;

private
   package Fichero_IO is new Ada.Sequential_IO (Elemento);
end Copias;
```

2) (2 puntos)

Se dispone del tipo de dato `Coordenada` definido en el siguiente paquete, que consta de la coordenada UTM y de una descripción de texto:

```
package Coordenadas is

   subtype Tipo_Descripcion is String(1..20);

   type UTM is record
      X : Float;
      Y : Float;
      Huso : Integer;
   end record;

   type Coordenada is record
      Posicion : UTM;
      Descripcion : Tipo_Descripcion;
   end record;

   type Coordenada_Ref is access Coordenada;

   function Cercanas
      (C1,C2 : Coordenada; Dist : Float) return Boolean;
   -- Devuelve True si C1 y C2 estan separadas menos de Dist
   -- en Km.

end Coordenadas;
```

Basado en este tipo se define una ruta como una lista de punteros a coordenadas de la siguiente manera:

```
with Coordenadas, Listas_Doblemente_Enlazadas;
use Coordenadas;

package Rutas is new
   Listas_Doblemente_Enlazadas(Coordenada_Ref,"=");
```

Crear un procedimiento compilado separadamente al que se le pasa una ruta (lista del paquete `Rutas`), una coordenada y una distancia (`Float`) y pinta en pantalla separadas por comas las descripciones de los puntos de la ruta que están a una distancia menor de la indicada de la coordenada dada.

3) (4 puntos)

Se desea realizar una aplicación de gestión del suministro de combustibles a las estaciones de servicio en la planta de distribución de una refinería. La planta dispone de cuatro grupos de depósitos (para gasolina 95, 98, súper y gasóleo) con capacidad total de 25.000.000 de litros por grupo. Los depósitos tienen una capacidad máxima de 1.000.000 de litros. El objetivo es disponer de una base de datos capaz de realizar la gestión de la planta distribuidora controlando los llenados y extracciones de combustible de cada grupo de depósitos.

El tipo `Deposito` se implementa en el paquete `Depositos`, que funciona tal como se explica en los comentarios:

```
package Depositos is

   Capacidad_Max : constant := 1_000_000;
   type Capacidad is range 0..Capacidad_Max;

   type Combustible is (SP95,SP98,Super,Gasoleo);

   type Deposito is private;
   type Deposito_ID is private;
   type Deposito_Ref is access Deposito;

   procedure Asigna_Combustible_E_ID
     (Dep : in out Deposito; Comb : Combustible);
   -- Asigna un combustible a un deposito.
   -- Solo se puede asignar una vez a cada deposito.
   -- La asignación de combustible asigna tambien
   -- un ID al deposito.
   -- El intento de asignacion a un asignado eleva Asignado.

   procedure Rellena (Dep : in out Deposito;
                      Comb : Combustible;
                      Cant : Capacidad;
                      Resto : out Capacidad);
   -- Rellena Cant hasta la capacidad maxima.
   -- Lo que no consigue rellenar lo devuelve en Resto.
   -- Puede elevar Incorrecto si el combustible no coincide
   -- con el asignado

   procedure Extrae (Dep : in out Deposito;
                     Comb : Combustible;
                     Cant : Capacidad;
                     Resto : out Capacidad);
   -- Extrae Cant hasta que queda vacio.
   -- Lo que no consigue extraer lo devuelve en Resto.
   -- Puede elevar Incorrecto si el combustible no
   -- coincide con el asignado
```

```
   function Identificador (Dep : Deposito) return Deposito_ID;
   -- Devuelve el ID asignado al deposito o eleva No_Asignado

   Asignado : exception;
   No_Asignado : exception;
   Incorrecto : exception;

private

   type Deposito_ID is ...

   type Deposito is ...

end Depositos;
```

Se pide realizar la implementación de la parte privada de este paquete así como del cuerpo para que funcione de acuerdo con la especificación.

Se pide además realizar el diseño de un paquete capaz de gestionar una planta de distribución cualquiera (en particular la propuesta). Los requisitos para esta planta consisten en la implementación de las siguientes operaciones:

- Rellenar combustible: como entrada se da el número de litros y el tipo de combustible, y devuelve una lista con los identificadores de los depósitos en los que se debe hacer el relleno y la cantidad a rellenar en cada uno. Se deberá dar como error el hecho de que no se pueda almacenar todo el combustible (los depósitos deben quedar como estaban).
- Extraer combustible: como entrada se da el número de litros y el tipo de combustible, y devuelve una lista con los identificadores de los depósitos de los que se debe extraer el combustible y la cantidad a extraer de cada uno. Se deberá dar como error el hecho de que no se pueda extraer todo el combustible (los depósitos deben quedar como estaban).

Implementar únicamente la operación de relleno de combustible. En el diseño e implementación se deben utilizar los ADTs vistos en clase que se consideren más adecuados, justificando su uso.

4) (1 punto)

Para el problema anterior (3), proponer un mecanismo que permita el uso del tipo `Deposito` en aplicaciones concurrentes. La propuesta se debe hacer considerando que la interfaz del paquete `Depositos` (parte visible de la especificación) debe permanecer igual. Describir las modificaciones necesarias sobre la implementación original.

5) (1 punto)

Se dispone del siguiente tipo etiquetado definido en un paquete:

```
package Viviendas is

   subtype Superficie is Float range 0.0..400.0;

   type Vivienda is tagged record
      Dimension : Superficie;
      Lugar : String(1..50);
   end record;

   type Vivienda_Ref is access Vivienda'Class;

   function Superficie_Total (V : Vivienda) return Superficie;

end Viviendas;
```

Se pide hacer otro paquete (especificación y cuerpo) con un tipo etiquetado que sea extensión de `Vivienda`, y que posea un campo más correspondiente a la superficie del garaje (`Dimension_Garaje` de tipo `Superficie`). Además, debe redefinir `Superficie_Total` para que calcule la suma de la dimensión de la vivienda y del garaje.