Examen de Lenguajes de Alto Nivel

Septiembre 2002

Cuestiones (4 cuestiones, 4 puntos en total)

1) El siguiente fragmento de código Ada implementa el algoritmo de ordenación de burbuja para ordenar un array de strings (del subtipo `Nombre`). Modificarlo para hacerlo más eficiente haciendo que en lugar de un array de strings sea un array de punteros a esos strings.

```
subtype Nombre is String(1..20);
N : constant:=100;
type Muchos_Nombres is array(1..N) of Nombre;
procedure Burbuja (A : in out Muchos_Nombres) is
   Temp : Nombre;
begin
   for i in 1..N-1 loop
      for j in reverse i+1..N loop
         if A(j) < A(j-1) then
            Temp:=A(j);
            A(j):=A(j-1);
            A(j-1):=Temp;
         end if;
      end loop;
   end loop;
end Burbuja;
```

2) La función `Calcula1` cuya especificación se muestra abajo puede elevar las excepciones `Constraint_Error` y `Error_de_Algoritmo`; la función `Calcula2` sólo eleva `Constraint_Error`. Se pide escribir una función compilada separadamente que llame a `Calcula1` y retorne el valor retornado por ella; si se eleva `Error_De_Algoritmo` entonces llama a `Calcula2` y retorna lo retornado por ella; si en cualquiera de los dos casos se elevase `Constraint_Error`, pone un mensaje de error en pantalla y vuelve a elevar la misma excepción.

```
package Algoritmos is
   function Calcula1(X : Float) return Float;
   function Calcula2(X : Float) return Float;

   Error_De_Algoritmo : exception;
end Algoritmos;
```

3) Se dispone del siguiente paquete:

```
package Libros is
   type Libro is record
      Autor, Titulo : String(1..20);
      N_Autor, N_Titulo : Integer range 0..20;
   end record;
end Libros;
```

`N_Autor` y `N_Titulo` son, respectivamente, el número de caracteres válidos del autor y de título. Se pide añadir a la especificación un procedimiento que tenga como parámetros un fichero de texto (del tipo `Ada.Text_IO.File_Type`) y un libro, y que sirva para leer del fichero indicado los datos de un libro. Estos datos estarán en una línea, separados ambos por un carácter `'|'`. Se pide también escribir el *pseudocódigo* de la implementación del procedimiento (no se pide el cuerpo de `Libros`).

4) Se dispone del siguiente paquete Ada:

```
package Computadores is
   type Computador is tagged record
      Modelo : String(1..4);
      Memoria : Integer;
   end record;

   procedure Imprime (C : Computador);
end Computadores;

with Ada.Text_IO; use Ada.Text_IO;
package body Computadores is
   procedure Imprime (C : Computador) is
   begin
      Put_Line("Computador : "&C.Modelo);
      Put_Line("   Memoria : "&Integer'Image(C.memoria));
   end Imprime;
end Computadores;
```

Se pide implementar en un nuevo paquete otro tipo etiquetado que sea una extensión de `Computador`, y tenga un campo más: dirección de red (String de hasta 16 caracteres). Asimismo, debe redefinir la operación `Imprime` para que llame a la operación `Imprime` del `Computador` y posteriormente escriba el nuevo campo con un formato similar. Se pide la especificación y el cuerpo del nuevo paquete.

Examen de Lenguajes de Alto Nivel

Septiembre 2002

Problema (6 puntos)

Se desea implementar un sistema para contabilizar estadísticas de acceso a un servidor. Se dispone para ello de un paquete ya implementado que contiene el tipo de datos que define los datos básicos del servidor:

```
package Servidores is
   type Byte is range 0..255;
   type Direccion is array(1..4) of Byte;
   type Servidor is record
      Nombre: String(1..20);
      N_Nombre : Integer range 0..20; -- num de caracteres de Nombre
      Dir_IP : Direccion;
   end record;
   function "=" (S1,S2 : Servidor) return Boolean;

   type Mes is (Ene,Feb,Mar,Abr,May,Jun,Jul,Ago,Sep,Oct,Nov,Dic);
   type Estadistica is array (Mes) of Natural;
end Servidores;
```

La función `"="` compara las direcciones IP de los servidores y retorna `True` si son iguales y `False` en caso contrario.

Se dispone asimismo de una implementación de un mapeado mediante técnicas de troceado (*hash*) con la siguiente especificación:

```
with Servidores, use Servidores;
package Mapas_Servidores is

   type Mapeado is private;
   No_Cabe : exception;
   No_Hay : exception;

   procedure Haz_Nulo (El_Mapa : in out Mapeado);
   procedure Asigna (El_Dominio : in Servidor;
                     El_Rango   : in Estadistica;
                     El_Mapa    : in out Mapeado);
   procedure Elimina (El_Dominio : in Servidor;
                      El_Mapa    : in out Mapeado);
   procedure Calcula (El_Dominio : in Servidor;
                      El_Mapa    : in Mapeado;
                      El_Rango   : out Estadistica;
                      Existe     : out Boolean);
   function Num_Relaciones (El_Mapa : Mapeado) return Natural;
   function Primer_Dominio (El_Mapa : Mapeado) return Servidor;
   function Siguiente_Dominio (El_Mapa : Mapeado) return Servidor;
private
   type Mapeado is ...
end Mapas_Servidores;
```

El mapeado permite asignar a cada `Servidor` un dato del tipo `Estadistica`, en el que se almacenan el número de accesos a ese servidor para cada uno de los meses del año.

Las cuatro primeras operaciones de este paquete son como las descritas en clase para los *Mapeados*. Las otras tres sirven para recorrer todos los elementos del mapeado y se describen a continuación:

- `Num_Relaciones`: retorna el número de relaciones almacenadas en el mapeado actualmente.
- `Primer_Dominio`: retorna el primer dominio almacenado en el mapeado. Si no hay ninguno, eleva `No_Hay`.
- `Siguiente_Dominio`: retorna el dominio siguiente al último obtenido por medio de esta misma función o de `Primer_Dominio`. Si no existe el dominio siguiente, eleva `No_Hay`.

Se pide implementar el cuerpo del siguiente paquete, que facilita la obtención de las estadísticas de acceso:

```
with Servidores; use Servidores;
package Accesos is
   procedure Registra (S : Servidor; M : Mes; Num : Natural);

   function Max_Accesos_Mes(M : Mes) return Servidor;

   function Max_Accesos return Servidor;

   function Maximo_Mes (S : Servidor) return Mes;

end Accesos;
```

El cuerpo contendrá una variable del tipo `Mapeado` que almacenará los datos estadísticos. Las operaciones descritas trabajan con esa variable, y deben realizar lo siguiente:

- `Registra`: Si el servidor `S` está almacenado en el mapeado, añade a su estadística del mes `M` un número de accesos igual a `Num` (para ello debe obtener la estadística, actualizarla, y volverla a asignar). Si no está almacenado, crea una nueva relación en el mapeado donde la estadística de todos los meses refleja cero accesos, excepto la del mes `M`, que reflejará un número de accesos igual a `Num`. Si se eleva `No_Cabe`, pone un mensaje en la pantalla.
- `Max_Accesos` (un parámetro): retorna el servidor cuyo número de accesos es máximo en el mes indicado por `M`, de entre todos los servidores almacenados en el mapeado. Si se eleva `No_Hay`, pone un mensaje de error.
- `Max_Accesos` (cero parámetros): retorna el servidor cuyo número de accesos es máximo para la suma de todos sus meses, de entre todos los servidores almacenados en el mapeado. Si se eleva `No_Hay`, pone un mensaje de error.
- `Maximo_Mes`: retorna el mes para el que el número de accesos del servidor `S` es máximo.