Universidad Rey Juan Carlos Curso 2000/01

Ingenierías Técnicas en Informática de Sistemas y de Gestión

Estructuras de Datos y de la Información

Examen Parcial Febrero 2001 Fecha: 6−2−2001

Normas:

- La duración del examen es de 3 horas.

- Todos los ejercicios se entregarán en hojas separadas.
- En cada ejercicio se indica su puntuación total y, en su caso, el valor de los apartados que lo componen.

Ejercicio 1. (Puntuación total: 5 puntos)

Se desea extender el TAD *TipoLista* estudiado en clase (véanse hojas adjuntas), añadiéndole las siguientes operaciones:

- ***Duplicado:* operación que recibe como entrada una lista *no vacía* y devuelve como salida el primer elemento de la lista tal que, o bien el que le sigue es igual que él, o bien no tiene ningún elemento siguiente.**

<u>Ejemplos</u>: *Duplicado* ([e1]) = e1 *Duplicado* ([e1,e2,e1,e2,e2,e3]) = e2 *Duplicado* ([e1,e2,e1,e2,e3]) = e3

- ***Ampliar*: operación que recibe como entrada una lista y devuelve como salida la misma lista pero tal que a todo elemento que cumpla una determinada propiedad se le añade justo a continuación un nuevo elemento, obtenido del primero aplicando a éste una cierta función de transformación.**

<u>Ejemplo</u>:

Si se trabaja con una lista de caracteres, y se considera que la propiedad consiste en que el carácter sea un dígito y que la función de transformación convierte un carácter en su sucesor en la tabla de caracteres, entonces:

***Ampliar* ([]) = [] *Ampliar* ([`a','b','1']) = [`a','b','1','2']**

***Ampliar* ([`2','a','3','z']) = [`2','3','a','3','4','z']**

Se pide:

- **(2 puntos) Especificar algebraicamente la extensión propuesta.**
- **(2 puntos) Implementar en *Ada 95* la extensión anterior. Para ello se deberá extender el paquete *Listas* estudiado en clase mediante la programación de un paquete hijo denominado *Listas.Examen*, del que se deberá dar tanto su interfaz (fichero *ads*) como su cuerpo (fichero *adb*). La operación *Duplicado* se deberá implementar obligatoriamente de forma *recursiva*, mientras que la operación *Ampliar* se deberá implementar *iterativamente* y de forma que *modifique* la lista de entrada en lugar**

de crear una nueva. No se pide estimar la complejidad de ninguna operación.

- **(1 punto) Se necesita un programa en *Ada 95* que realice las siguientes tareas: 1) leer una lista de enteros pedida al usuario; 2) indicar cuál es el primer elemento que aparece duplicado (en el sentido explicado más arriba); 3) transformar la lista leída añadiendo, inmediatamente detrás de cada número cuyo valor absoluto sea par, su número opuesto; 4) mostrar por pantalla la nueva lista. Dicho programa, incompleto, es:**

```
WITH Ada.Exceptions, Ada.Text_IO, Ada.Integer_Text_IO, Auxiliar_IO;

WITH Listas, Listas.IO, Listas.Examen;

PROCEDURE Usa_examen IS

−− instanciación del paquete Listas y su E/S

PACKAGE ListaEnt IS NEW Listas (TipoElemento => Integer);

PACKAGE ListaEntIO IS NEW ListaEnt.IO

(Leer_Elemento => Auxiliar_IO.Leer_Entero,

Escribir_Elemento => Auxiliar_IO.Escribir_Entero);

*** COMPLETAR ***

−− variables

lista: ListaEnt.TipoLista;

BEGIN −− programa

Ada.Text_IO.Put ("Introduzca una lista de enteros);

*** COMPLETAR ***

END Usa_examen;
```

Se pide completar el programa anterior, suponiendo que el usuario puede no introducir nada o bien introducir una serie de números enteros que se supondrán escritos correctamente (es decir, no es necesario controlar las posibles excepciones producidas en la entrada de datos).

Ejercicio 2. (Puntuación total: 1,5 puntos)

Especificar algebraicamente una extensión de la especificación del TAD *Conjuntos* (véanse hojas adjuntas) que contenga las dos siguientes operaciones:

- *Diferencia*: operación que recibe como entrada dos conjuntos y devuelve como salida la diferencia de ambos (es decir, el conjunto formado por todos aquellos elementos que pertenecen al primero pero no al segundo).

- *DifSimétrica*: operación que recibe como entrada dos conjuntos y devuelve como salida la diferencia simétrica de ambos (es decir, el conjunto formado por todos aquellos elementos que pertenecen a uno

de los dos conjuntos de entrada −pertenecen a su unión− pero no a ambos −no pertenecen a la intersección−).

Ejercicio 3. (Puntuación total: 1 punto)

- **(0,5 puntos)** Se considera una cola de caracteres, implementada mediante un *vector circular con campo longitud*, cuya situación, después de realizar una serie de operaciones, es la siguiente:

1	2	3	4	5	6	7	8	9	10
A	B	C	D	E	F	G	H	I	J

primero = 3

último = 8

longitud = 6

Se pide:

  - Partiendo del estado anterior, describir las situaciones intermedias de la estructura después de aplicar de forma consecutiva **cada una** de las operaciones que se indican a continuación. En caso de que alguna de estas operaciones no pueda llevarse a cabo, se indicará el motivo y se ignorará, continuando por la siguiente.

Insertar(K) ; Eliminar ; Insertar(L) ; Insertar(M) ; Insertar(N) ; Insertar(O) ; Insertar(P) ; Eliminar

  - Una vez realizadas las operaciones anteriores, ¿cuál sería el resultado de ejecutar la operación *PrimeroCola*?

- **(0.5 puntos)** Indicar, **justificando** la respuesta, cómo debe declararse el tipo de datos *TipoCola*, PRIVATE o LIMITED PRIVATE, al implementar una cola en *Ada95* mediante un vector circular con campo longitud.

Ejercicio 4. (Puntuación total: 1 punto)

Dada la especificación algebraica del TAD *Pilas*, se extiende dicha especificación con la operación *Misterio*, cuya sintaxis y semántica se describe a continuación:

PARÁMETROS GENÉRICOS

OPERACIONES

Propiedad: TipoElemento −> Booleano

FIN PARÁMETROS

Misterio: TipoPila −> TipoPila

Misterio (CrearPilaVacia) = CrearPilaVacia

Misterio (Apilar(e, pila)) =

SI Propiedad(e) −>

Apilar(e, Misterio(pila))

| CrearPilaVacia

- **(0,5 puntos)** Sea *P1* la pila de números naturales obtenida partiendo de una pila vacía y apilando sucesivamente los elementos 5,4,3 y 10, y *P2* la pila obtenida partiendo de una pila vacía y apilando sucesivamente los elementos 10, 6, 1 y 2. Aplicar las ecuaciones de la operación *Misterio*, paso a paso, en ambos casos, suponiendo que el parámetro genérico *Propiedad* es una operación que devuelve cierto si y sólo si el número natural de entrada es menor o igual que 5.
- **(0,5 puntos)** Explicar en una frase qué hace, en el caso general, dicha operación. ¿Qué haría si en su semántica se sustituye la última línea de la segunda ecuación por Misterio(pila)?

Ejercicio 5. (Puntuación total: 1,5 puntos)

Razonar cuál es la complejidad (tiempo de ejecución en el peor caso) de la función F que se describe a continuación:

```
FUNCTION F(n: Natural) IS

FUNCTION Aux(n: Natural) RETURN Natural IS

BEGIN

IF n = 0 THEN

RETURN 0;

ELSE

RETURN 1 + Aux(n−1);

END IF;

END Aux;

suma : Natural := 0;

prod : Natural := 1;

i: Natural := 1;

seguir: Boolean := True;

LIMITE: CONSTANT Positive := 20;

BEGIN −− función F

WHILE seguir AND i <= n LOOP
```

```
IF n MOD 2 = 0 THEN

suma := suma + Aux(i);

ELSE

prod := prod * Aux(i);

ENDIF;

seguir := suma <= LIMITE;

i := i+1;

END LOOP;

RETURN suma + prod;

END F;
```

En el cálculo se podrán emplear las soluciones a las siguientes familias de recurrencias:

$$T(n) = \begin{cases} cn^k & \text{si } 0 \quad n < b \\ aT(n-b) + cn^k & \text{si } n \quad b \end{cases}$$

$$T(n) \quad \begin{cases} \mathbf{O}(n^k) & \text{si } a < 1 \\ \mathbf{O}(n^{k+1}) & \text{si } a = 1 \\ \mathbf{O}(a^{n \text{ DIV } b}) & \text{si } a > 1 \end{cases}$$

$$T(n) = \begin{cases} cn^k & \text{si } 1 \quad n < b \\ aT(n/b) + cn^k & \text{si } n \quad b \end{cases}$$

$$T(n) \quad \begin{cases} \mathbf{O}(n^k) & \text{si } a < b^k \\ \mathbf{O}(n^k \log n) & \text{si } a = b^k \\ \mathbf{O}(n^{\log_b a}) & \text{si } a > b^k \end{cases}$$

3

4