INFORMATICA.
EXAMEN PARCIAL Junio 2007. Pascal

Apellidos, Nombre	
DNI	

La puntuación total del examen es 10 puntos. Para aprobar será necesario un mínimo de 5 puntos.

1. Con objeto de escribir un programa para una máquina expendedora, consideremos las siguientes declaraciones:

```
CONST
 TIPOS_DE_MONEDAS = 7;
TYPE
 VALORES = ARRAY [1 .. TIPOS_DE_MONEDAS] OF REAL;
 MONEDAS = ARRAY [1 .. TIPOS_DE_MONEDAS] OF CARDINAL;
VAR
 Val : VALORES;
```

Consideremos además que la variable `Val` se inicializa del siguiente modo al principio del programa:

```
Val[1] := 0.01;
Val[2] := 0.05;
Val[3] := 0.10;
Val[4] := 0.20;
Val[5] := 0.50;
Val[6] := 1.0;
Val[7] := 2.0;
```

La idea es que la máquina expendedora puede manejar siete tipos de monedas, cada uno con los valores indicados (1 céntimo, 5 céntimos, 10 céntimos, 20 céntimos, 50 céntimos, 1 euro y 2 euros), y `Val[i]` (con $1 \leq$ `i` ≤ 7) indica el valor en euros del tipo de moneda `i`-ésimo.

El tipo `MONEDAS` se utiliza para representar una cantidad de dinero indicando el número de monedas de cada clase. Por ejemplo, si `c` es una variable de tipo `MONEDAS` con los siguientes valores:

```
c[1] := 1;
c[2] := 0;
c[3] := 0;
c[4] := 0;
c[5] := 1;
c[6] := 2;
c[7] := 0;
```

la cantidad representada es 2.51 euros (1 moneda de 1 céntimo, 1 moneda de 50 céntimos y 2 monedas de 1 euro).

a) (1 *punto*) Escribe una función `Valor` que tome un parámetro de tipo `VALORES` y otro de tipo `MONEDAS` y devuelva un `REAL` que se corresponda con el valor en euros.

b) (3 *puntos*) Con objeto de resolver el problema de devolver el cambio de una compra, escribe una función `Cambio` que tome un parámetro de tipo `VALORES`, otro de tipo `REAL` que indique la cantidad total en euros introducida por el usuario de la máquina, otro de tipo `REAL` que indique el precio del producto comprado y devuelva un resultado de tipo `MONEDAS` que indique el cambio a devolver, de modo que el número total de

monedas devuelto sea mínimo. Asume que la máquina dispone de tantas monedas de cada tipo como sea necesario para el cambio.
Por ejemplo, si `Val` está definida tal como se indicó anteriormente, `Cambio(Val, 10.0, 2.5)` debe devolver un vector `v` tal que:

```
v[1] = 0
v[2] = 0
v[3] = 0
v[4] = 0
v[5] = 1
v[6] = 1
v[7] = 3
```

ya que el cambio (7.5 euros) se corresponde a 1 moneda de 50 céntimos, una moneda de 1 euro y 3 monedas de 2 euros).

c) (1 *punto*) Resuelve el problema del apartado **b)**, pero suponiendo que el número de monedas disponibles de cada clase para el cambio está limitado. Para ello, define una función `CambioLimitado`, que además de los parámetros de la función `Cambio`, tome otro de tipo `MONEDAS` que indique la cantidad disponible de cada tipo de moneda.

2. El método de ordenación *Bucket Sort* permite ordenar un vector de valores que están dentro de cierto rango. A modo de ejemplo, supongamos que los valores que queremos ordenar están en el rango [1,10] y que éstos son:

7 6 1 3 7 1 10 7

Para ordenar estos valores, se crea un *cubo* por cada valor del rango. La idea es que el cubo c_i ha de contar el número de veces que aparece el valor i en el vector que queremos ordenar. Para el ejemplo dado, los cubos calculados serían:

2	0	1	0	0	1	3	0	0	1
c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}

Una vez calculados los cubos, es fácil obtener el vector ordenado correspondiente. Basta con recorrer los cubos de izquierda a derecha y añadir al vector de datos ordenados tantos elementos con valor i como valor tenga cada uno de los cubos c_i :

$$\underbrace{1\ 1}_{\text{por } c_1}\quad \underbrace{3}_{\text{por } c_3}\quad \underbrace{6}_{\text{por } c_6}\quad \underbrace{7\ 7\ 7}_{\text{por } c_7}\quad \underbrace{10}_{\text{por } c_{10}}$$

Supongamos definidas las siguientes constantes y tipos:

```
CONST
      TAMANYO_VECTOR = 8;
      MIN_RANGO = 1;
      MAX_RANGO = 10;
TYPE
      RANGO = MIN_RANGO .. MAX_RANGO;
      VECTOR = ARRAY [1..TAMANYO_VECTOR] OF MAX_RANGO;
```

Donde `TAMANYO_VECTOR` es el número de elementos del vector a ordenar, y `MIN_RANGO` y `MAX_RANGO` delimitan el rango de los valores a ordenar.

a) (4 ptos.) Escribe un programa en Pascal que incluya un procedimiento `BucketSort` que tome como parámetro un `VECTOR` y lo ordene usando el método descrito. Define los tipos adicionales que necesites.

b) (1 pto.) Razona el número de comparaciones y movimientos que realiza tu programa para ordenar un vector de tamaño N.