PROGRAMACIÓN BÁSICA – Examen Global
Convocatoria Ordinaria
19 de Enero de 2018

PREGUNTA 1 (3 puntos). Tipos Básicos

1.1 Especifica e implementa en Ada un subprograma *Numero_Impar_Montaña* que indique si un número dado es *número impar montaña,* éste es un número que cumple que, tiene un número impar de dígitos y el dígito que ocupa el lugar central de dicho número es estrictamente mayor que los otros dígitos que componen el número.

Por ejemplo: 1**6**5 y el 376**9**572

En la implementación de este subprograma se pueden utilizar los subprogramas siguientes que se suponen ya implementados:

```
procedure Digito_I (Num: in Integer; I: in Positive; DigI: out Integer);
-- pre: Num > 0, 1<= I <= cantidad de dígitos de Num
-- post: DigiI es el I-ésimo dígito de Num
-- Ejemplo: para el número 53, el primer dígito es 5 y
--    el segundo es 3.

function Contar_Digitos (Num : in Integer) return Natural;
   -- Post: devuelve el número de dígitos de Num
```

1.2. Especifica e implementa en Ada un programa *Suma_Impar_Montaña* que, leída una secuencia de números positivos terminada en 0, indique si la suma de los números impares montaña de la secuencia también es un número impar montaña.

Por ejemplo, si la secuencia de entrada es <**162**, 1331, 4, **121**, 356 0> indicará que es cierto ya que 162 y 121 tienen un número impar de dígitos y son montaña y su suma es 283, que es un número impar montaña.

En la implementación se debe utilizar el subprograma implementado en el apartado 1.1.

PROGRAMACIÓN BÁSICA – Examen Global
Convocatoria Ordinaria
19 de Enero de 2018

PREGUNTA 2 (2 puntos) – Vectores y Matrices

Tenemos datos de los 3.546 vecinos de una comunidad de vecinos. Por cada vecino tenemos información de su nombre, número de piso y mano donde vive. La comunidad vive en un rascacielos de 100 pisos (numerados del 1 al 100) y en cada piso hay 10 manos (nombradas desde la 'A' a la 'J'). Todos los datos los representamos con las siguientes estructuras de datos:

```
Max: constant Natural := 3546;
type T_Vecino is record
   nombre: String(1..25);
   piso: integer range 1..100;
   mano: character range 'A'..'J';
end record;
type T_Comunidad is array (1..Max) of T_Vecino;
```

Para acceder más fácilmente al número de habitantes por cada vivienda (piso y mano concretos), queremos transformar los datos en otra estructura de datos (matriz t_Rascacielos):

```
type T_Rascacielos is array (1..100, 'A'..'J') of Natural;
```

Implementa en Ada el <u>subprograma</u> *Obtener_Num_Vecinos_Por_Vivienda* que, dados los datos de toda la comunidad, obtenga la distribución de los vecinos por vivienda en el rascacielos y cuya especificación en la siguiente:

```
procedure Obtener_Num_Vecinos_por_Vivienda (C: in T_Comunidad;R:out T_Rascacielos);
--pre: C contiene los datos de los vecinos de la comunidad
--post: R contiene los datos de cuántos vecinos hay por cada vivienda según lo que
--     aparece en C
```

Por ejemplo, si en el 1º A viven 5 personas en el vector de la comunidad habrá 5 posiciones en las que aparecerá información de vecinos del piso 1 y mano 'A'. Tras el proceso de ir revisando el vector de la comunidad de vecinos, en R (de tipo T_Rascacielos) aparecerá el valor 5 en la primera celda, es decir, R(1, 'A')=5.

Jon 1 A	Aiora 3 B	Koldo 2 A	Miren 2 A		Mireia 1 A	Leire 99 A		Iker 1 A			Aritz 1 A			Ane 1 A		
1	2			...	32		....	145		...	1345		...	2500		....

	'A'	'B'		...	'J'
1	5				
2	2				
3		1			
4					
...					
99	1				
100					

NOTA: procura resolver el problema con un único recorrido del vector de la Comunidad.

PROGRAMACIÓN BÁSICA – Examen Global
Convocatoria Ordinaria
19 de Enero de 2018

PREGUNTA 3 (1,75 puntos) – Listas estáticas

Disponemos de las siguientes declaraciones:

```
type T_Vector is array (1..10) of Natural;
type T_Lista_Estatica is record
  Elem: T_Vector;
  Cont: Natural;
end record;
```

Especifica, implementa en Ada y realiza un estudio de casos de prueba para el subprograma *Quitar_Repetidos_Primero* que, a partir de una lista estática de naturales L de tipo T_Lista_Estatica, elimine de dicha lista los elementos que son iguales al primero.

Por ejemplo, para **L**= [8, 8, 2, 8, 6, 0, 5, 6, 8] debería quedar [8, 2, 6, 0, 5, 6]

PREGUNTA 4 (**1,25 ptos**) – **Listas dinámicas**

Dada la siguiente definición de lista dinámica:

```
type t_nodo;
type a_nodo is access t_nodo;
type t_nodo is record
      valor: Natural;
      sig: a_nodo;
end record;
```

Implementa en Ada el subprograma *Quitar_Repetidos_Primero* que, a partir de una lista dinámica de naturales L de tipo a_nodo, elimine de dicha lista los elementos que son iguales al primero.

Por ejemplo, para **L**= [8, 8, 2, 8, 6, 0, 5, 6, 8] debería quedar [8, 2, 6, 0, 5, 6]