



Cuestión 1

2 puntos

Se desea crear el TAD *palabra* con un conjunto de operaciones. Para ello se dispone de la siguiente signatura:

espec PALABRAS

usa cadenas, naturales, booleanos

género palabra

operaciones

p Vacia: → palabra	{crea la palabra vacía}
pon -letra: palabra carácter → palabra	{añade una letra al final de una palabra}
longitud: palabra → natural	{devuelve la longitud de una palabra}
pon -primera: carácter palabra → palabra	{añade una letra al principio de una palabra}
parcial letra: natural palabra → carácter	{devuelve la letra i-ésima}
parcial primera: palabra → carácter	{devuelve la primera letra de una palabra}
parcial última: palabra → carácter	{devuelve la última letra de una palabra}
parcial elimPrim: palabra → palabra	{devuelve la palabra sin la primera letra}
vacia?: palabra → booleano	{indica si una palabra está vacía}
pon -final: palabra → palabra	{desplaza la primera letra al final de la palabra}
pon -principio: palabra → palabra	{desplaza la última letra al principio de la palabra}
ordenada?: palabra → booleano	{devuelve verdadero si las letras de la palabra están ordenadas alfabéticamente}

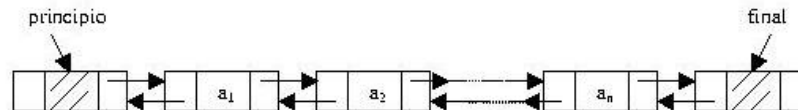
Ejemplos: **pon**-final ($a_1 a_2 \dots a_n$) = $a_2 a_3 \dots a_n a_1$
 pon-principio ($a_1 a_2 \dots a_n$) = $a_n a_1 \dots a_{n-1}$

Clasificar las operaciones y completar la especificación algebraica del TAD PALABRAS con los dominios de definición de las operaciones parciales y las ecuaciones, sabiendo que las operaciones generadoras son *pVacia* y *pon-letra*.

Cuestión 2

1.25 puntos

Dada una lista doblemente enlazada de enteros (LDEE), con nodo cabecera al principio y al final tal y como se describe en la figura.



Se pide:

- a) Escribir en lenguaje algorítmico, la acción **Siniguales**, que dada una lista (definida por sus punteros *principio* y *final*), elimina todos los elementos duplicados.

acción Siniguales (p, f: puntero a nodo)

{* p y f son dos punteros que apuntan a los nodos cabecera y final de la lista con la que se va a operar */}

Ejemplo: La lista con los elementos: 1, 5, 3, 2, 2, 5, 7, 2 quedaría tras la aplicación de la acción como: 1, 5, 3, 2, 7.

- b) Escribir en lenguaje algorítmico, la función **Orden**, que devuelve 1 si la lista esta ordenada ascendentemente de principio a final, 2 si está ordenada ascendentemente de final a principio, y 0 si no esta ordenada de ninguna forma.

función Orden (p, f: puntero a nodo): entero

Cuestión 3

1.5 puntos

Dada una tabla Hash implementada con el siguiente tipo de datos:

```
Constante Max =1000
Tipos PtNodo = puntero a nodo
nodo = registro
        Clave : clave;
        Valor : valor;
        Sig: PtNodo;
registro:
tablaH = tabla [1..Max] de PtNodo;
```

Donde reservamos las primeras 750 posiciones de la tabla para ubicar los elementos y las últimas 250 posiciones para la resolución de colisiones por el mecanismo de recolocación en la propia tabla (dispersión cerrada).

Se pide:

Diseñar la acción **InsertaHash**(var t: tablaH; c: clave; v: valor), que inserta el nuevo par <c,v> en la tabla t en el caso de que la clave c no exista en la tabla, o modifica el valor asociado a la clave c por el nuevo valor v en caso que exista la clave c en la tabla.

El mecanismo de resolución de las colisiones vendrá determinado por el valor devuelto por la función de dispersión. En caso de colisión, se seguirá el criterio siguiente:

- Si H devuelve un valor par, se usará el mecanismo de encadenamiento mediante punteros (dispersión abierta)
- Si H devuelve un valor impar, se usará el mecanismo de recolocación en la propia tabla (dispersión cerrada), siguiendo la siguiente estrategia: el nuevo par se ubicará en la primera posición libre de las reservadas para las colisiones. Si al intentar resolver una colisión por este método, todas las posiciones reservadas para colisiones están ocupadas, se mostrará un mensaje indicando este hecho.

Notas:

- Se supone que existe una función de dispersión $H(c:clave) \text{ dev } 1..Max*(3/4)$, que puede utilizarse
- Se supone que existe una función $ParImpar(N:entero) \text{ dev } \text{booleano}$, de la que se puede hacer uso

Cuestión 4

1.25 puntos

Se desea enriquecer el TAD *pila* con las siguientes operaciones:

```
acción apilar_fondo (var p: pila; e: elemento);
/* inserta el elemento e en el "fondo" de la pila p */
```

```
función elimina_fondo (var p: pila): elemento;
/* devuelve el elemento situado en el "fondo" de la pila p, eliminándolo de ésta */
```

Se pide implementar, **de forma recursiva**, ambas operaciones. **No** se podrá hacer uso de ninguna estructura auxiliar salvo la propia pila p, y sólo se podrán utilizar las operaciones propias del TAD *pila* estudiadas en clase (crearPila, apilar, etc).