NOMBRE.....

NÚM. de MATRÍCULA..... GRUPO.....

Examen Final. Informática. Febrero 2003

Instrucciones

- El examen consta de un conjunto de **diez** preguntas, cada una de las cuales puntuará **cero** o **un** punto.
- Se calificará con **un punto** si la respuesta correcta se indica en la forma **más simple**.
- La duración total del examen será de **dos horas y treinta minutos.**

---

1. Codificar el valor numérico decimal **–7.25** en binario utilizando un formato de coma o punto **fijo** con 12 bits y complemento a 2 y otro formato de coma o punto **flotante** con 8 bits para la mantisa y 4 bits para el exponente (ambos en complemento a 2). Nota: Debe considerarse que la representación en coma flotante está **normalizada** ($1/2 \leq |\text{mantisa}| < 1$).

Coma o punto fijo:

							,					

Coma o punto flotante:

2. Completar la función `horaCorrecta` para que devuelva `true` o `false` si la **hora del día** almacenada en el parámetro `a` de tipo `tiempo` es **correcta** o **no**, respectivamente. Por ejemplo, para la hora 23:59:35 debe devolver el valor `true` y para 24:03:45, 15:60:38 o 6:11:-23 debe devolver el valor `false` (es decir, sólo es válida una hora que se visualice en cualquier reloj digital de uso corriente).

```
type tiempo=record
       hr,mn,sg:integer;
       end;
function horaCorrecta(a:tiempo):boolean;
  begin
horaCorrecta:=
  end;
```

3. Dadas dos variables globales `totalSegundos` (de tipo `longint`) y `t` (de tipo `tiempo`, tipo de dato definido según la pregunta anterior) y asignado un valor determinado a `totalSegundos`, se pide: codificar una rutina denominada `transformacion` de forma que, la **ejecución de la llamada** `transformación(totalSegundos, t);` **almacene** en los campos correspondientes de la variable `t`, el **intervalo de tiempo**, desglosado en horas, minutos y segundos, **equivalente** al almacenado en `totalSegundos`. Por ejemplo, 8140 segundos equivalen a 2 horas, 15 minutos y 40 segundos.

4. Completar la función **recursiva** `n_digitos` para devuelva el **número de digitos** del parámetro entero `n`. Por ejemplo, la llamada a `n_digitos(73)` debe devolver el valor `2`, `n_digitos(0)` devuelve el valor `1`, `n_digitos(-15)` devuelve el valor `2` y `n_digitos(46041)` devuelve el valor `5`.

```
function n_digitos(n:longint):byte;
  begin
```

```
  end;
```

5. El rango de un conjunto de valores numéricos es la diferencia entre el valor máximo y el mínimo ($x_{Max}-x_{min}$). Completar la función `rango` para que devuelva el **rango** de los valores reales almacenados en los elementos de la variable **dinámica** de tipo `vector` cuya dirección de memoria, `a`, se da como parámetro de la función.

```
const N=100;
type indice=1..N;
     vector=array[indice] of real;
     ptr=^vector;
function rango(a:ptr):real;
  var min,max:real; i:indice;
  begin
```

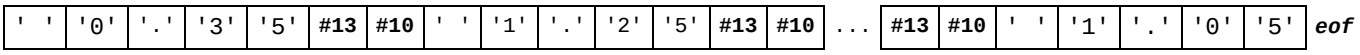
```
  end;
```

6. Completar la función `sumaDosMax` para que devuelva la **suma** de los **dos mayores valores** numéricos enteros almacenados en el archivo de disco cuyo nombre se indica en el parámetro `s`. Si el archivo de disco está vacío, la función debe devolver el valor 0. Si el archivo contiene un único valor entero, la función debe devolver su duplo. Nota: debe considerarse el caso en que todos los valores enteros almacenados en el archivo sean negativos.

```
type enteros=file of integer;
function sumaDosMax(s:string):integer;
  var f:enteros; a,max1,max2:integer;
  begin
```

```
  end;
```

7. En un archivo de disco en formato ASCII se almacenan una serie de valores reales con dos dígitos decimales (fraccionarios) significativos y separados entre sí por marcas de fin de línea en el formato que se muestra a continuación:



Completar el procedimiento `copiaConNuevoSaldo` para que realice una copia de un archivo original, **añadiendo un nuevo valor** real **al final**. El nuevo valor introducido es el resultado de **sumar** el valor del parámetro `incremento` al **último** valor almacenado en el archivo original. El archivo copia generado debe **respetar** el formato anterior. Los nombres de los archivos original y copia se dan en los parámetros `original` y `copia`. Nota: No debe considerarse el caso en que el archivo original no exista, pero se debe considerar el caso en que este archivo pueda estar vacío. En este caso se debe almacenar únicamente el valor del parámetro `incremento` en el archivo copia.

```
procedure copiaConNuevoSaldo(original,copia:string; incremento:real);
```

```

```

```
end;
```

8. Completar el código fuente de la unidad `examen` para que incluya en la **parte pública** una función `soloDigitos` que devuelva `true` si la **variable dinámica** de tipo `string`, cuya dirección de memoria se da en el parámetro `t` de la función, está **vacía** o almacena **exclusivamente** caracteres correspondientes a **dígitos decimales**. En caso contrario, debe devolver el valor `false`. Por ejemplo, si la variable dinámica de tipo `string` es `'7109'` o `''`, la función debe devolver el valor `true` y si la variable dinámica de tipo `string` es `'60A41'` devolverá el valor `false`.

```
unit examen;
```

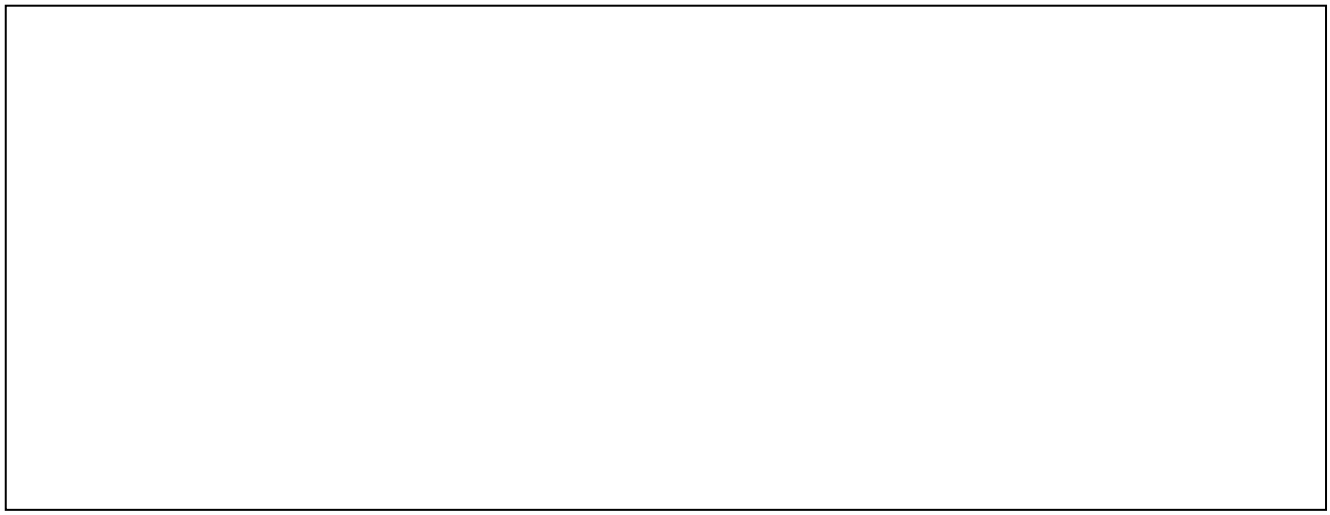
```

```

```
end.
```

9. Completar el procedimiento **recursivo** `inserta_en_orden` para que **introduzca** un **nuevo elemento** en una lista dinámica simple a cuyo primer elemento apunta el parámetro `p`. Los campos `numero` y `nombre` del nuevo elemento deben tomar los valores de los parámetros `n` y `s`, respectivamente. La estructura debe permanecer **ordenada** según el valor del campo `numero` de forma **creciente**. Nota 1: Se admite la inserción de un nuevo elemento con un valor para el campo `numero` repetido en la lista. En este caso es indiferente colocar el nuevo elemento antes o despues del elemento ya existente. Nota 2: Se debe considerar también el caso de lista inicialmente vacía.

```
type ptr=^elemento;
     elemento=record
         numero:integer; nombre:string; sig:ptr end;
procedure inserta_en_orden(var p:ptr; n:integer; s:string);
  var aux:ptr;
  begin
```



```
  end;
```

10. Completar la función `producto` para que devuelva el **producto** de la parte **entera** de los valores reales almacenados en el campo `dato` de **todos los elementos** de una lista dinámica simple **circular** apuntada por `p`. Si la lista está vacía debe devolver el valor 0. Nota: Puede emplearse la función `int` (declarada en la unidad `system`) que devuelve la parte entera de un valor real dado como parámetro en la llamada a la función.

```
type ptr=^elemento;
     elemento=record
         dato:real; sig:ptr end;
function producto(p:ptr):real;
```