NOMBRE.....

NÚM. de MATRÍCULA..... GRUPO.....

Informática. Examen Final. Febrero 2007

Instrucciones

- El examen consta 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 media.**

---

1. Representar su número de matrícula dividido entre $\mathbf{10^3}$ (mil) como un valor en **punto fijo** y **complemento a dos**, si no admite una representación exacta truncar por la derecha:

Punto fijo y complemento a dos:										.				

2. Completar la función `tresceros` que devuelva `true` si un vector tiene al menos 3 componentes nulas y `false` en caso contrario:

```
type vector=array[1..100] of real;
function tresceros(v:vector):boolean;
```

3. Escribir **3 formas** de calcular el **producto** `p` de todas las componentes de un vector `v` de dimensión `n` (`n>0`). En cada modalidad se debe utilizar una instrucción de Turbopascal distinta: `while`, `repeat` y `for..to`.

```
program producto;
const n=100;
type vector=array[1..n] of real;
procedure davalor(var w:vector);
   begin {No se da el codigo de davalor, pero se incluye en el
   programa para ser utilizado en el mismo} end;
var i:integer; v:vector; p:real;
begin
davalor(v);
```

Usando `while`:

Usando `repeat`:

Usando `for..to`:

```
end.
```

4. Completar la función `premio` para que devuelva como **resultado el premio acumulado** que se gana jugando a la ruleta una cantidad de `x` euros, sabiendo que:
   - si se acierta la apuesta par/impar entonces el parámetro `parimpar` es `true` y el premio es el **doble** de la cantidad apostada. Si no se acierta esta apuesta entonces `parimpar` es `false` y el premio es 0
   - si se acierta la apuesta rojo/negro entonces el parámetro `rojonegro` es `true` y el premio es el **doble** de la cantidad apostada. Si no se acierta esta apuesta entonces `rojonegro` es `false` y el premio es 0
   - si se acierta el número apostado entonces el parámetro `numero` es `true` y el premio es **34 veces** lo apostado. Si no se acierta esta apuesta entonces `numero` es `false` y el premio es 0.

   Ejemplo: la llamada `y:=premio(10,true,true,true)` asignará a `y` 10·2+10·2+10·34=380

```
function premio(x:real;parimpar,rojonegro,numero:boolean):real;
```

5. Completar la función `npares` que devuelve el número de **dígitos pares** de un número entero `n` dado como parámetro. Ejemplos: en el caso de que `n` sea 0 devolverá 1, si n es 1 devolverá 0, si n es 2 devolverá 1, si n es 234 devolverá 2.

```
function npares(n:word):byte;
```

6. Construir la función `decimal` que obtenga como **resultado el valor entero** representado por el parámetro `s` de tipo cadena, sabiendo que `s` está formada sólo por caracteres ceros (‘0’) y unos (‘1’) y que representa un valor **entero natural positivo en binario**. Ej. para `'1110'` devuelve `14`

```
function decimal(s:string):longint;
```

7. Dados dos archivos con tipo de valores enteros, representados por las variables `f` y `g`, que han sido previamente asignados, Se pide completar el procedimiento `incorporar` que añade **al final** del archivo `f,` los datos que están en el otro archivo `g`. Sabiendo que sólo se pueden añadir los **nuevos datos** del archivo asociado a `g,` si **no están** en el archivo asociado a `f`.
Notas: El archivo representado por `f` existe y no está vacío. El archivo representado por `g` existe y puede estar vacío.

`f` | 2 | 3 | `g` | 4 | 4 | 2 | 5 | ➔ `f` **resultante** | 2 | 3 | 4 | 5 |

```
Type archivo=file of integer;
procedure incorporar(var f,g:archivo);
```

8. Construir la función **recursiva** `minimo` que obtenga como **resultado** el menor valor almacenado en el campo dato de los elementos de la lista simple enlazada a cuyo primer elemento señala `prim.`
Nota: La lista no está vacía.

```
type ptr=^elemento; elemento=record dato:integer; sig:ptr; end;
function minimo
```

9. Se representa un **monomio** de coeficiente real en 5 variables (*x,y,z,t,w*) mediante una variable dinámica de tipo `monomio`. Por ejemplo, el monomio $-5\ x^3\ y^2\ t^4\ w^3$ se representa por su coeficiente, *-5*, y los grados, *3,2,0,4,3*, de las variables (*x,y,z,t,w*) en el orden definido por el tipo `indice.` Se pide construir un procedimiento que devuelva la dirección de memoria de una **nueva** variable dinámica que tenga como valor el **monomio producto** de dos monomios señalizados por `p` y `q.`
Ejemplo: $(-5\ x^3\ y^2\ t^4\ w^3) * (3\ x^2\ y^3\ z^7\ t^3\ w^5) = (-15\ x^5\ y^5\ z^7\ t^7\ w^8)$
Nota: Los tipos enumerados son validos como tipo de la variable de control de un bucle

```
type  indice=(x,y,z,t,w);
      vector=array[indice] of byte;
      monomio = record
            coef: real;
            grados: vector;
            end;
     ptr_a_monomio=^monomio;
procedure producto(p,q:ptr_a_monomio; var r:ptr_a_monomio);
```

10. Completar el procedimiento `copia` que construya una **nueva** lista simple enlazada con los mismos elementos de una lista simple circular a cuyo primer elemento señala el parámetro `circ`. **Además**, el procedimiento debe dar como **resultado la dirección del primer elemento** de la nueva lista simple generada. Nota: La lista circular de partida puede estar vacía.

```
type ptr=^elemento; elemento=record dato:integer; sig:ptr; end;

procedure copia(circ: ptr;var prim:ptr);
var b,q: ptr;
begin
if circ=nil then prim:=nil
   else
   begin
   new(prim);
   prim^.dato:=circ^.dato;
   prim^.sig:=nil;
   q:=prim;b:=circ^.sig;
   while b<>circ do
        begin
```

```
        end;
  end; end;
```