NOMBRE.....

NÚM. de MATRÍCULA..... GRUPO.....

Examen Final. Informática. Septiembre 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.**

---

1. Las siguientes secuencias de dígitos **binarios** representan el **mismo valor** numérico **decimal** empleando diversos formatos de representación habituales. Indicar el **valor decimal** representado y el **formato** correspondiente a cada una de las secuencias binarias. En el caso de que exista más de un formato para una determinada secuencia, es suficiente con dar uno de ellos.

Secuencia binaria	Valor decimal	Formato de representación empleado
`00010010`		
`0001 1000`		
`10010010`		
`010010,00`		
`01001 101`		

2. Completar la **expresión** que falta en el siguiente programa para que **asigne** a la variable booleana `ok` el valor `true` si la cadena de caracteres introducida previamente en la variable `s` de tipo `string` es **válida** como contraseña o clave. En caso contrario debe asignar a `ok` el valor `false`. Para que la cadena de caracteres introducida sea válida como clave ha de cumplir los siguientes **requisitos**:
   - Longitud de la cadena **superior** a 3 e **inferior** a 26 caracteres.
   - El primer carácter **no** corresponde a un dígito decimal.

   Por ejemplo: `'abcd'`, `'Eureka2003'` y `'X-0123456789t'` son válidas, mientras que `'abc'`, `'2Eureka'` y `'abcdefghijklmnopqrstuvwxyz'` **no** son válidas. Nota: **no** deben emplearse los operadores de relación de igualdad (=) ni de desigualdad (<>).

```
var ok: boolean; s:string;
begin
write('Introduce una clave: '); readln(s);
ok:= [          ]
if ok then writeln('Clave valida') else writeln('Clave no valida')
end;
```

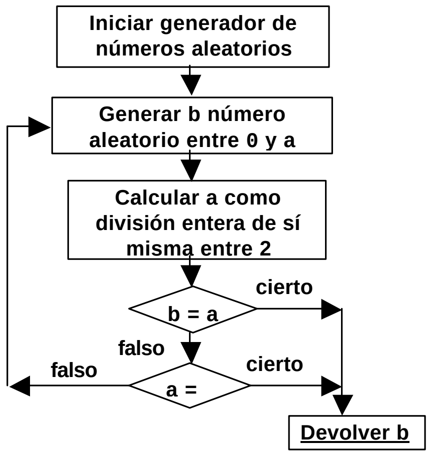
3. Completar la función **recursiva** `binario` para que devuelva como **resultado una cadena de caracteres** con la **representación binaria** correspondiente al número entero **positivo** `n`. Por ejemplo, la llamada a la función `binario(0)` debe devolver la cadena `'0'`, la llamada a `binario(1)` devuelve `'1'` y la llamada a `binario(58)` devuelve `'111010'`. Nota: el ordinal del carácter ASCII `'0'` es 48 y el del carácter ASCII `'1'` es 49.

```
function binario(n:word):string;
   begin
   [                    ]
   end;
```

4. Completar **todas** las declaraciones necesarias del siguiente programa para que, al ejecutarse, genere **100** valores aleatorios reales y posteriormente visualice por pantalla el valor **mínimo** de todos ellos.

```
procedure [                                        ] ;
   [                    ]
   begin
   min:=v[1,false];
   for k:=2 to n do if v[k,false]<min then min:=v[k,false]
   end;
var a:vector; m:real; i:indice;
begin
randomize; for i:=1 to n do a[i,false]:=random;
minimo(a,m);
writeln('El valor minimo es: ',m);
{ ... resto del programa ... }
end.
```

5. Completar la función `aleatorio` que admite un parámetro `a` entero positivo o cero y cuyo comportamiento viene dado por el **diagrama de flujo** de la figura.




```
function aleatorio(a:word):word;
```

6. En un archivo binario se almacena una serie de registros de tipo `comb`. Completar el procedimiento `insercion` para que **inserte un nuevo registro** de tipo `comb` **al final** del archivo de disco cuyo nombre se indica como primer parámetro del procedimiento (`t`). El campo `ident` del nuevo registro debe tomar **el valor** del parámetro `id`, mientras que el campo `numero` debe tomar el valor del campo del mismo nombre del último registro del archivo original **más** el valor del parámetro `incremento`. 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 en el campo `ident` el valor del parámetro `id` y en el campo `numero` únicamente el valor del parámetro `incremento`.

```
type comb=record
        ident:string;
        numero:real;
       end;
     arch=file of comb;
 procedure insercion(t:string; id:string; incremento:real);
```

7. Completar la función `todosMayores` para que devuelva el valor `true` si **todos** los valores almacenados en las líneas **pares** de un archivo de texto almacenado en disco, cuyo nombre se indica en el primer parámetro del procedimiento, `m`, son **estrictamente mayores** que el valor del segundo parámetro `c`. En caso contrario o si el archivo de disco está vacío debe devolver el valor `false`. Notas: No debe considerarse el caso en que el archivo no exista. Si el archivo no está vacío entonces está siempre estructurado en un número par de líneas. En las líneas **impares** del archivo se almacena una cadena de caracteres **no** numérica como, por ejemplo, muestra la figura adjunta.

```
JUAN PEREZ<eoln>
28<eoln>
PEDRO JIMENEZ<eoln>
45<eoln>
MARIA GARCIA<eoln>
61<eoln>
ANA LOPEZ<eoln>
36<eof>
```

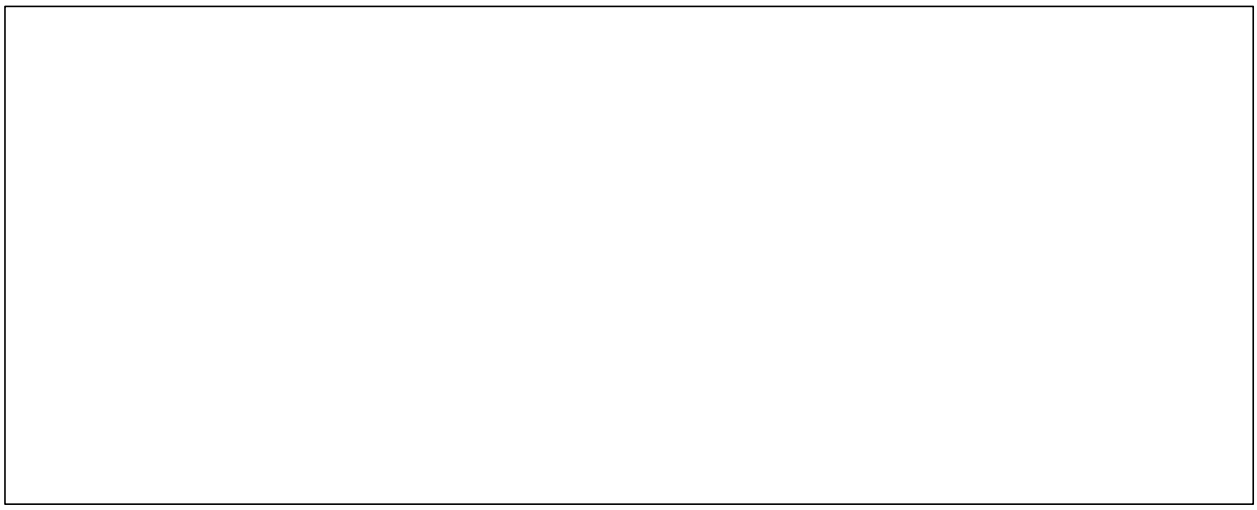
```
function todosMayores(m:string; c:integer):boolean;
```

8. Completar la función `habitacion_cliente` para que devuelva el **número** de la habitación que ocupa un **cliente determinado** en un hotel de 20 habitaciones (indicado a través del parámetro `h`). El nombre del cliente se da a través del parámetro `nombre`. Se considera que una habitación *k* **no** está ocupada si el puntero de índice *k* en la matriz `h` toma el valor `nil`. En caso de no hospedarse en el hotel la función debe devolver el valor `-1` . Se considera que un cliente no ocupa más de una habitación.

```
type  habitacion=record nombre : string; end;
      phabita = ^habitacion;
      hotel = array[1..20] of phabita;
function habitacion_cliente(h:hotel;nombre:string):shortint;
```

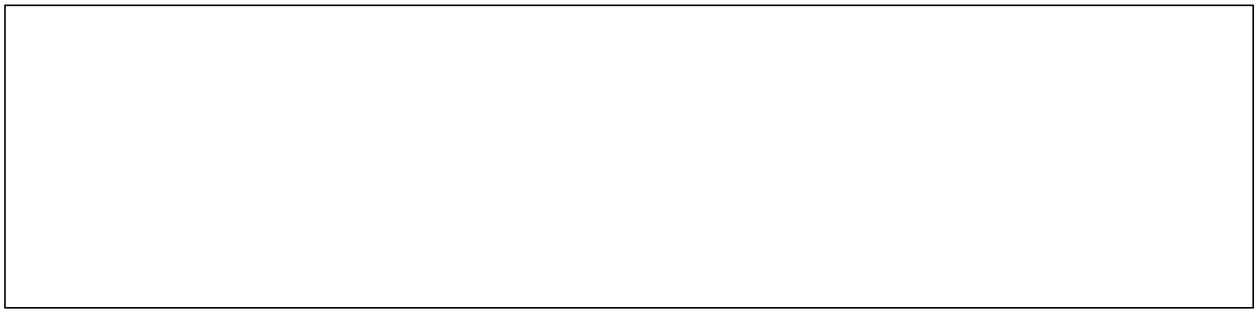
9. Completar la función `reserva` para que devuelva el **puntero** a la habitación que ha de reservarse en un hotel para un **nuevo cliente** cuyo nombre es dado con el parámetro `nombre`. El hotel se indica mediante el parámetro `ph` como un puntero que apunta a la primera habitación de la lista dinámica simple que forman sus habitaciones. Debe reservarse la **primera habitación** de la lista que **no** se encuentre reservada. Una habitación no está reservada cuando su puntero `pcli` tiene el valor `nil`. La reserva supone la creación de una **nueva variable** dinámica de tipo `cliente`, la **asignación** del `nombre` del cliente y la **asignación** de la dirección de memoria al puntero miembro `pcli` de la habitación. En caso de hotel lleno la función debe devolver el valor `nil`. Se supone que el cliente no tiene ninguna reserva previa en el hotel y que el hotel tiene al menos una habitación.

```
type  cliente = record nombre:string; end;
      pcliente = ^cliente;
      phabita = ^habitacion;
      habitacion = record
        pcli : pcliente;
        sig : phabita;
        end;
function reserva(ph:phabita;nombre:string):phabita;
```



10. Completar el procedimiento `soloFebrero` para que asigne un nuevo valor al campo `a` de **todos** los elementos de una lista dinámica simple **circular** a cuyo primer elemento apunta el parámetro puntero `prim`. El nuevo valor asignado al campo `a` de un elemento es el valor de `b` (del mismo elemento) si éste es **estrictamente inferior** a 4, en caso contrario, se le asigna el valor de `b` **más** una décima parte de `c` (también del mismo elemento). Nota: debe considerarse también el caso de lista circular **vacía**.

```
type ptr=^elemento;
     elemento=record
       a,b,c:real;
       sig:ptr
       end;
procedure soloFebrero(prim:ptr);
   var p:ptr;
   begin
```



```
   end;
```