NOMBRE.....

NÚM. de MATRÍCULA..... GRUPO.....

Examen Final. Informática. Junio 2005

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. **A partir** de la representación de -9.30 en punto fijo y complemento a dos, con 10 bits y el punto entre el 5º y 6º bit, se pide obtener dicho valor en binario en coma flotante con 5 bits para la mantisa (m) y los otros cinco para representar el exponente (e), ambos en complemento a dos, utilizando formato normalizado (1/2 <=|m| <1).

Representación en coma flotante normalizada en binario

2. Construir el procedimiento `convertir` para que **devuelva** al programa `pp` desde el que se llama o ejecuta **una cadena de caracteres** con las cifras dispuestas en sentido contrario a las del **valor entero positivo** `n` dado como parámetro del procedimiento. Ejemplo: Si en la instrucción `readln(n);` `n` toma el valor `12345` el valor de `s` visualizado en pantalla es `'54321'`

```
program pp;
procedure convertir
```

```
var s:string; n:longint;
begin
writeln('intro el valor entero no negativo');Readln(n);
if n<>0 then convertir(n,s) else  s:='0';
writeln('si el valor es ', n, ' la cadena es ', s);
end.
```

3. Completar el procedimiento `cambio` para que cambie el valor almacenado en el campo `nota` de todos los registros del archivo de tipo `alumno` cuyo nombre y ubicación se indica en el valor del parámetro `s` del procedimiento. El nuevo valor de nota es el valor antiguo **incrementado** en una unidad. El archivo existe y si está vacío el procedimiento no debe hacer nada.

```
type alumno=record nmat:string[5]; nota:real; end;
procedure cambio(s:string);
```

4. Completar la función booleana `esreal` que devuelva el valor `true` cuando el valor del parámetro `s` de tipo `string` represente **un valor real con el formato** que se describe a continuación y `false` en caso de no cumplir con dicho formato:

   1. el primer carácter debe ser el signo + ó –
   2. tiene que haber al menos un punto
   3. el punto separa dígitos decimales a derecha e izquierda del mismo. El número de dígitos de la parte entera es mayor o igual que 1 y el número de dígitos de su parte fraccionaria es mayor o igual que 1.

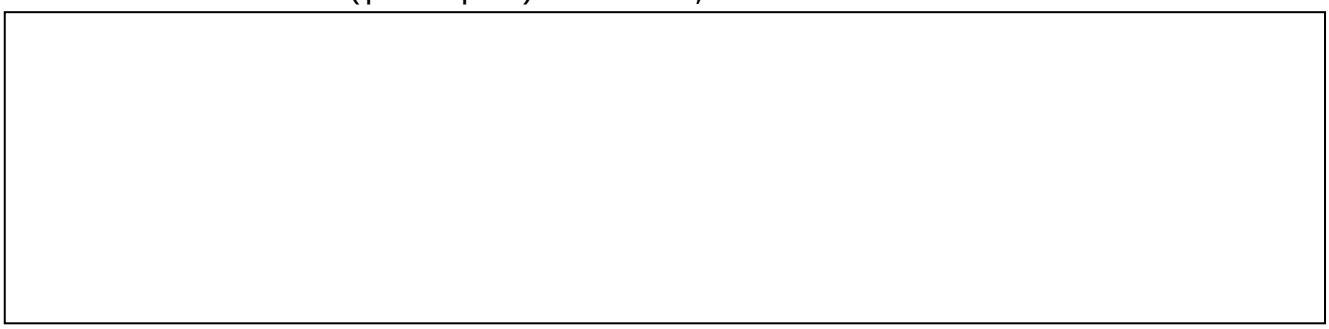
   Nota: Se asegura que el valor de la variable S es una cadena que tiene a lo más un punto.
   Los siguientes valores de la variable `s` obtienen `true` como resultado de la función `esreal`:
   `-3.12  +23.1  -0.12345  +12345.0  -0.0`
   Los siguientes valores de la variable `s` obtienen `false` como resultado de la función `esreal`:
   `-.12  +231  -2345.  -0  +0  +.0  1.234`

```
function esreal(s:string):boolean;
var ok:boolean; i,p:integer;
begin
ok:=true; i:=2; p:=i;
if (s[1]<>'+') and (s[1]<>'-') then ok:=false
          else while ok and (i<=length(s)) do
               begin
```

```
               end;
esreal:=ok and (p<>2) and (p<>length(s));
end;
```

5. Construir la función booleana `todosPos` que obtenga como resultado el valor `true` si **todos** los valores de la lista simple enlazada, cuyo primer elemento es señalizada por `prim`, son números enteros positivos o cero. La función debe devolver el valor `true` si la lista está vacía.

```
type ptr=^elemento; elemento=record dato:integer; sig:ptr; end;
function todosPos(prim:ptr):Boolean;
```



6. Construir la función recursiva `termino` que calcule **el termino n-ésimo** de una sucesión (x_i) (i=1…) tal que $x_n=a.x_{n-1}+b.x_{n-2}$ para n>2. x1 y x2 corresponden a los valores x_1 y x_2

```
function termino(x1,x2,a,b:real;n:word):real;
begin
```

```
end;
```

7. Dada una matriz `A` de dimensión `mxn`, cuyos elementos toman valores **booleanos**. Completar la **declaración de variables** del programa principal y **construir el procedimiento** `indices` que, tras la llamada `indices(A,V);`, obtenga como resultado la **asignación de valor a las k primeras componentes** del vector `V`. El vector `V` esta constituido por pares de valores enteros, y cada par lo componen los índices de la fila y la columna de un elemento de `A` cuyo valor es `true`.

```
program pgm;
const m=10;n=10;
type
 matriz=
 par=record
 vector=
var A:matriz;V:vector;
procedure davalor(var B:matriz);
{ . . . Resto de la declaracion de davalor . . .}
procedure indices(
begin
davalor(A);
indices(A,V);
end.
```

8. Completar la función `ultimo` para que **devuelva el índice del último elemento** del vector `v`, dado como parámetro de la función, al que le fue asignado un valor. Se considera que, previamente, se asignaron valores a los elementos de dicho vector a través del procedimiento `asignar`. Nota: la llamada a la función `random(n)` devuelve un número entero aleatorio en el conjunto [0,n). Ejemplo: si en la llamada a `asignar, random(100)+1` dio los valores 10, 8, 3, 5 y 5 los siguientes elementos de `v` fueron asignados con este orden: `v[1]:=10`, `v[10]:=8`, `v[8]:=3`, `v[3]:=5`, `v[5]:=5`, siendo el elemento 5 el último asignado.

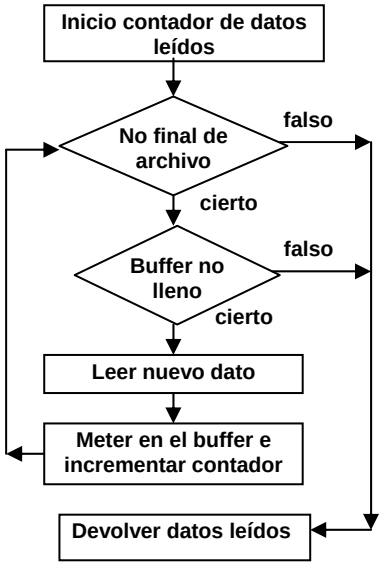
```
type indice=1..100;
     vector= array[indice] of indice;
procedure asignar(var v:vector);
var i,j:indice;
begin
j:=1; repeat
      i:=j; v[i]:=random(100) + 1; j:=v[i];
      until i = j;
end;
function ultimo(v:vector):indice;
```

9. Construir la **función** redondeo que devuelve **el múltiplo** de `d` **más próximo por exceso** al valor entero `n`. Los valores `n` y `d` son los parámetros de la función y ambos son valores enteros positivos. Ejemplos: `redondeo(5,2)` devuelve 6, `redondeo(10,4)` devuelve 12, `redondeo(8,4)` devuelve 8.

10. Completar la función `leer` cuya finalidad es **leer un bloque de registros** a partir de la posición actual del puntero asociado al archivo `f` dado como argumento y previamente abierto. Los registros leídos son alojados en el vector `b` (buffer). La función **devuelve el número de registros leídos**, pudiendo no llenar el buffer si no quedan suficientes registros, y siendo cero cuando el cursor está al final del archivo en el momento de la llamada (el archivo puede estar vacío). El algoritmo ha de corresponderse lo más exactamente posible con el diagrama de flujo.

```
type archivo = file of real;
     buffer = array[1..100] of real;
function leer(var f:archivo; var b:buffer):word;
var i:word; d:real;
begin
```

```
end;
```



11. Completar la función `npasajerosllegada` que devuelve **el número de pasajeros** del vuelo, dado por el identificador `id` como argumento, que ha tomado tierra. El parámetro `prim` señala el comienzo de la lista simple de las pistas de aterrizaje del aeropuerto. En cada pista, el campo puntero `vuelo` apunta al `avion` que ha aterrizado en dicha pista, y es `nil` si la pista está vacía. El campo `pasajeros` de `avion` representa el número de pasajeros que transporta. Si el avión buscado no ha aterrizado la función devolverá -1. La lista de las pistas puede estar vacia.

```
type avion = record id:string; pasajeros:integer; end;
     ptrpista = ^pista;
     pista = record vuelo:^avion; sig:ptrpista; end;
function npasajerosllegada(prim:ptrpista; id:string):integer;
```