NOMBRE.....

NÚM. de MATRÍCULA..... GRUPO.....

Examen Final. Informática. Junio 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 media**

---

1. Dada la secuencia de bits **11001001 1101**, indicar cuál es el valor decimal representado, si corresponde a un número real expresado en coma flotante (los 8 primeros bits corresponden a la mantisa y los cuatro últimos para el exponente, ambos en punto fijo y complemento a dos).

   La anterior secuencia de bits no está normalizada, indicar también la expresión binaria normalizada

2. Completar el procedimiento `suma_bin` para que sume los valores de tres parámetros `a,b,c` de tipo `integer` que representan valores siempre binarios (sólo valores `0` ó `1`) y devuelva como **resultado** el valor de dos parámetros `r,s` de tipo `char.` No es necesario comprobar que los valores `a,b,c` son ceros o unos. El parámetro `s` debe almacenar el bit de menor peso (2^0=1) resultante de la suma de `a,b,c` . El parámetro `r` debe almacenar el bit de peso 2^1=2 resultante de la suma de `a,b,c.`
   - Por ejemplo si `a=1 b=0 c=1` su suma en binario será `1+0+1=10`, entonces `s='0'` y `r='1'`
   - Por ejemplo si `a=1 b=0 c=0` su suma en binario será `1+0+0=01`, entonces `s='1'` y `r='0'`

```
procedure suma_bin
```

3. Sea *a* un número entero positivo y sea una sucesión de números reales x_i dada por

$$x_0=1 \quad \text{y} \quad x_{i+1}=\frac{1}{2}\left(x_i+\frac{a}{x_i}\right), \qquad i=0,1,2,...$$

se puede demostrar que $x_i \to \sqrt{a}$ si $i \to \infty$

Completar el programa en TurboPascal que calcula $\sqrt{a}$ con un error menor que 0.001 utilizando la anterior expresión. Nota: No se puede declarar ninguna variable, ni utilizar la función `sqrt` .

```
program raiz;
var a:word;x:real;
begin
```

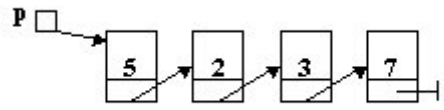
```
end.
```

4. Dada una lista simple enlazada con un número **impar** de elementos, a cuyo primer elemento apunta el puntero `prim`, construir un **procedimiento** que borre el elemento situado en la mitad de la lista. Se debe considerar el caso en que la lista esté vacía.
El tipo de `prim` es `ptr=^elemento; elemento=record dato:integer;sig:ptr end`

```
procedure borrar_medio
```

5. Se tiene una lista simple enlazada tal que el dato almacenado en cada elemento de la lista es un dígito decimal (0,1,2,3,4,5,6,7,8,9). Construir la función **recursiva** `valor` para que obtenga como **resultado** el valor entero decimal `n` almacenado en toda la lista, sabiendo que el dígito de menor peso (10^0=1) es el dato del primer elemento de la lista, el siguiente dígito correspondiente al peso 10^1=10 es el dato del segundo elemento de la lista y así sucesivamente hasta el último elemento de la lista que es el dígito de mayor peso. En el caso de lista vacía el resultado debe ser cero.
El tipo de `p` es `ptr=^elemento; elemento=record dato:0..9; sig:ptr end;`

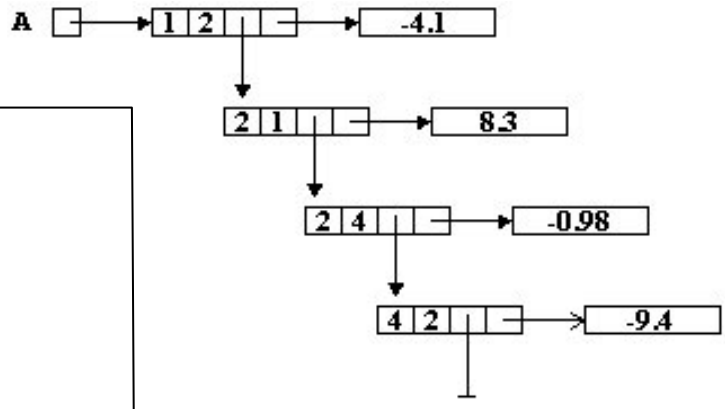
Ejemplo:



La ejecución de la instrucción `n:=valor(p);` aplicada a la lista de la figura asigna a la variable `n` el valor `7325`

```
function valor(p:ptr):word;
```

6. Construir **la zona de declaraciones** (instrucciones `type` y `var`) que permita **definir** la estructura dinámica de la figura para el caso de una matriz $A \in R^{mxn}$**.** Dicha estructura es una lista dinámica simple en la que **cada elemento** almacena la posición (fila y columna) y un puntero apuntando el valor para cada uno de los `k>0` elementos reales no nulos de la matriz $A \in R^{mxn}$.
   Además, construir **el cuerpo del programa** que **genere y asigne** valor a todos los campos de la estructura dinámica **utilizando las declaraciones anteriores**. Los datos de fila, columna y valor de los elementos no nulos de la matriz se deben asignar a través de teclado, sin utilizar ninguna estructura de tipo `array`. Por ejemplo, el programa genera y asigna la estructura por el puntero A de la figura siguiente. En este caso la matriz tiene `k=4` elementos no nulos: $a_{12}=-4.1,\ a_{21}=8.3,\ a_{24}=-0.98,\ a_{42}=-9.4,$



7. Construir el procedimiento `guardar` para que almacene en la variable archivo de texto `g` los **valores no nulos** de la estructura dinámica anterior señalizada por el puntero `A`. En **cada línea** del archivo de texto deben figurar **tres** datos: la **fila** y la **columna** del elemento **no nulo** de la matriz `A` y el **valor** de dicho elemento. Los datos de cada elemento deben poder leerse posteriormente mediante una única instrucción `readln(f,c,valor);`. La variable archivo `g` esta asignada pero el archivo no está abierto.

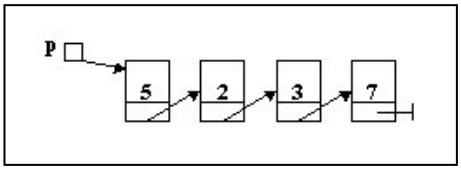
```
procedure guardar(var g:text; A:ptr);
```

8. Construir el procedimiento `copiar` para que lea la información del archivo de texto `g` generado en la pregunta número 7 y genere un **archivo con tipo** (`h`), en el que cada **registro** del archivo contenga tres datos: la fila y la columna del elemento no nulo de la estructura y el valor de dicho elemento. La variable archivo de texto `g` está previamente asignada pero el archivo no está abierto y la variable archivo con tipo `h` no está asignada, ni el archivo abierto.
   Incluir en la sección de declaraciones del procedimiento las instrucciones `type` y `var` que sean necesarias para su correcta ejecución.

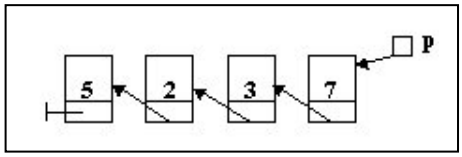
```
procedure copiar(var g:text);
```

9. Construir el procedimiento **NO RECURSIVO** `invertir` que invierta una lista simple enlazada apuntada por `p`. La lista puede estar vacía. No se tiene que generar una nueva lista. El tipo de dato `ptr` es `ptr=^elemento; elemento=record dato:integer; sig:ptr end;`

Situación inicial



Situación final




```
procedure invertir(var p:ptr);
var q,r:ptr;
begin
if p<> nil then begin
        r:=p; q:=p^.sig; p^.sig:=nil;
```

```
                end;
end;
```

10. Construir la función `producto` para que devuelva como **resultado el producto** de las cifras de un número entero positivo `n`.
    Ejemplos: `producto(234)` →`24` `producto(2)` →`2` `producto(0)` →`0`

```
function producto(n:word):word;
```