NOMBRE.....

NÚM. de MATRÍCULA..... GRUPO.....

Informática. Examen Final. Septiembre 2006

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 el siguiente valor negativo dado en hexadecimal **`-AB.C`** en los siguientes formatos binarios, utilizando 13 bits, el punto no ocupa ningún bit:

Punto fijo y signo-magnitud:										.				
Punto fijo y complemento a dos:										.				

2. Una cadena de caracteres `s` almacena una frase con **una o varias** palabras. Cada palabra está separada de la anterior y de la siguiente por un sólo símbolo **&** . Delante de la primera palabra y detrás de la última **no** existe un símbolo **&** . Se pide desarrollar el procedimiento `muestra` que visualice cada palabra de la frase, dada como argumento, en una línea distinta de la pantalla. Los caracteres & no deben visualizarse. Un posible valor de la variable `s` es `'Juan&es&alto'`

```
procedure muestra(s:string);
```

3. Escribir **3 formas** de visualizar los números **impares** mayores o iguales que `1` y menores o iguales que `n` (`n>0`). En cada modalidad se debe utilizar una instrucción de Turbopascal distinta: `while`, `repeat` y `for..to`.

```
program impares;
var n,i:integer;
begin
write('introduzca un valor mayor o igual que 1'); readln(n);
```

Usando `while`:

Usando `repeat`:

Usando `for..to`:

```
end.
```

4. Construir la función **booleana** `proporcional` para que dados dos vectores `v` y `w` de `n` componentes **enteras**, obtenga como resultado `true` si el vector `w` es el producto de un escalar **entero** por el otro vector `v` y `false` en caso contrario. **Todas** las componentes del vector `v` son **distintas** de cero.
   Por ejemplo: Si `n=3`, `v=(1,2,3)` y `w=(2,4,6)` entonces `proporcional` devuelve `true`
   Si `n=3`, `v=(1,2,3)` y `w=(3,4,6)` entonces `proporcional` devuelve `false`
   Si `n=3`, `v=(1,2,3)` y `w=(0,0,6)` entonces `proporcional` devuelve `false`

```
const n=...; {n es un entero positivo }
type vector=array[1..n] of integer;
function proporcional(v,w:vector):boolean;
```

5. Completar la función **recursiva** `maximo` tal que, dado un vector `v` de dimensión máxima `m=10`, obtenga como resultado el valor de la componente de mayor valor de las `n` **primeras** componentes.
   El valor del parámetro `n` satisface: `n>= 2` y nunca es superior al número de elementos `m` de `v`.
   Por ejemplo:
   Si `v=(-2,3,9,99,8,0,-21,-11,98,12)` y `n=3` entonces el resultado de la función `maximo` es `9`.
   Si `v=(-2,3,9,99,8,0,-21,-11,98,12)` y `n=7` entonces el resultado de la función `maximo` es `99`.

```
const m=10;
type vector=array[1..m] of integer;
function maximo(v:vector;n:integer):integer;
```

6. Se parte de una lista **simple circular** a cuyo primer elemento apunta `prim`. El dato almacenado en cada elemento de la lista es una **cadena de caracteres** cuyo valor es el nombre de un día de la semana. El primer elemento de la lista tiene por valor `'lunes'`. La lista sigue el orden natural ascendente de `'lunes'` a `'domingo'`.
   Construir el procedimiento `dias` que, partiendo de los parámetros `inicio,final` con los nombres de dos días de la semana **distintos**, asigne valor a las primeras componentes del parámetro `v` de tipo `vector.` Cada componente del vector `v` es el nombre de uno de los días que hay entre los dos días dados como parámetros, incluidos ambos extremos.
   Nota: Las componentes no utilizadas en el vector `v` deben tomar como valor la cadena vacía, `''`.

Por ejemplo:
Si la llamada fuera: `dias('domingo','martes',prim,v);`
el vector `v` tomaría el valor `v=('domingo','lunes','martes','','','','')`

Si la llamada fuera: `dias('martes','domingo',prim,v);`
el vector `v` tomaría el valor
`v=('martes','miercoles','jueves','viernes','sabado','domingo', '')`

```
type ptr=^elemento; elemento=record dato:string; sig:ptr; end;
     vector=array[1..7] of string;
procedure dias(inicio,final:string;prim:ptr;var v:vector);
```

7. Sea un archivo de texto con nombre `'listin.txt'` que contiene un listín de teléfonos con el siguiente formato:

   `914567890 Jaime Urrutia`***`eoln`***`926789012 Enrique Urquijo`***`eoln...`***

   Los números de teléfono no tienen espacios entre medias, y entre el número y el nombre hay únicamente un espacio así como entre el nombre y el apellido. Cada número de teléfono y su correspondiente nombre y apellido están situados en una línea distinta de longitud menor o igual que 255. Completar el procedimiento `buscar` tal que muestre por pantalla el nombre y apellido de la persona que posee el número de teléfono dado por el parámetro `telefono`, si no se encuentra ha de mostrarse el mensaje `'no se encuentra'`. El listín puede estar vacío.

   ```
   procedure buscar(telefono:longint);
   ```

8. Completar el procedimiento `salvar` tal que escriba la lista dinámica simple dada a través del parámetro `prim` en el **nuevo** archivo `'listin.txt'` siguiendo el mismo formato que el indicado en la pregunta anterior.

```
type ptr=^elemento;
     elemento=record telefono:longint; nombre:string; sig:ptr; end;
procedure salvar(prim:ptr);
```

9. Sea un tablero de ajedrez definido a través del tipo de datos `tablero`, donde cada elemento del tablero es una `casilla`. Si el valor del elemento es `0` significa que la casilla está vacía, cualquier otro valor indica que está ocupada por una figura, si es positivo la figura es blanca y si es negativo es negra. Por ejemplo, si es 1 es un peón blanco y -1 si es negro. Completar el procedimiento `espeon` tal que devuelva `true` si en la posición del tablero dada por `i` y `j` hay un peón, en caso contrario devolver `false`.

```
type casilla= -6..6;
     rango= 1..8;
     tablero=array[rango,rango] of casilla;
function espeon(i,j:rango; t:tablero):boolean;
```

10. Sea el tablero de ajedrez definido en el ejercicio anterior. Completar el procedimiento `moverpeonblanco` tal que mueva el peón blanco indicado por la posición `i` y `j` dados como argumentos en el tablero dado a través de `t`:
    a. Si en esa posición no hay un peón blanco mostrar el mensaje 'No es un peón blanco'
    b. Si la posición final, que será (i, j+1), está fuera del tablero, mostrar el mensaje ‘Fuera del tablero’ y no modificar la posición. El peón sólo avanza una posición en la misma columna.
    c. Si la posición final está ocupada por otra figura, mostrar ‘Casilla destino ocupada’
    d. Finalmente si es posible mover el peón blanco indicado, la casilla destino debe quedar ocupada por el peón y la casilla de su anterior posición debe quedar vacía.

```
procedure moverpeonblanco(i,j:rango;var t:tablero);
```