NOMBRE.....

NÚM. de MATRÍCULA..... GRUPO.....

Examen Final. Informática. Junio 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. Sea el operador booleano **NN** definido mediante la siguiente tabla de verdad (0=*falso*, 1=*verdadero*):

P	Q	P NN Q
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

Escribir la **expresión más simple** posible del operador **NN** en función de los operadores booleanos básicos (AND, OR y NOT).

P **NN** Q =

2. El sistema de cobro de agua de una compañía suministradora considera una cuota **fija** de servicio (10 euros) y una cuota **variable** que penaliza el consumo excesivo de la forma que se indica en la tabla siguiente:

Consumo (m^3)	euros/m^3
Primeros 20 m^3	0.6
De 20 a 40 m^3	0.8
De 40 a 60 m^3	1.0
A partir de 60 m^3	1.2

Completar el programa `cuotagua` para que, dados unos metros cúbicos consumidos (valor introducido por teclado durante la ejecución del programa), devuelva el **coste total** del servicio suministrado, considerando que en la tabla se indica lo que hay que cobrar en la cuota variable por los metros cúbicos que se encuentran en el intervalo correspondiente. Por ejemplo, si se han consumido **55** m^3 se debería pagar en total: **10** + **20***0.6 + **20***0.8 + **15***1.0 = 53.00 euros.

```
program cuotagua;
```

```
writeln('El coste total es: ',ct:10:2,' euros');
end.
```

3. Completar la función `esPrimo` para que devuelva `true` si el valor del parámetro `n` (mayor que cero) es **primo** y `false` en caso contrario. Def.: un número entero es *primo* si sólo es divisible por sí mismo y por la unidad. Por ejemplo: **2**, 3, 5, 13 y 29 son números primos, mientras que **4**, 99, 169 no lo son. El valor entero **1 no** se considera *primo*.

```
function esPrimo(n:word):boolean;
```

4. Completar el procedimiento `incremento` para que dada una hora, expresada en horas y minutos, **devuelva** la hora **incrementada** en `x` minutos. Los parámetros formales de la rutina son **tres**: la hora **inicial**, el **incremento** de minutos y la hora **final**. Por ejemplo, si la hora inicial es *23:57* y x vale *15*, entonces la hora final debe tomar el valor *0:12*. Nota: Los intervalos de valores enteros **válidos** para representar las horas y los minutos son `0..23` y `0..59` respectivamente.

```
type horas = 0..23; minutos = 0..59;
     horaCompleta = record
       hr: horas; mn: minutos
       end;
procedure incremento
```

5. Completar el procedimiento `cambioClave` para que, al ejecutarse, **sustituya** una clave de usuario almacenada en un archivo de **texto** ASCII. Los parámetros del procedimiento son el **nombre** del archivo de disco (`archivo`), la clave **antigua** (`ca`) y la clave **nueva** (`cn`). El cambio de clave sólo se debe efectuar si el valor de la clave antigua (`ca`) coincide con la almacenada previamente en el archivo. Nota: se supone que el archivo ASCII existe (no es necesario verificar su existencia) y tiene una **única clave** almacenada (que es, además, la única línea de texto contenida en el archivo).

```
procedure cambioClave(archivo:string; ca,cn:word);
```

6. En tres parámetros (`e1`, `e2` y `e3`) de tipo `vector` se almacenan las temperaturas alcanzadas en las correspondientes estaciones meteorológicas en cada hora en punto desde las 0 a las 23 horas. Completar el procedimiento `tmedia` para que **almacene** una estructura de datos de tipo **vector** con el valor **medio** de las temperaturas alcanzadas a cada hora en un archivo en disco. El nombre del archivo creado en disco debe ser `media.dat`.

```
type vector = array[0..23] of real;
     archivo = file of vector;
procedure tmedia(e1,e2,e3:vector);
   var f:archivo;
```

7. Completar la **unidad** `examen` para que incluya una **rutina** que **intercambie** el valor de **dos** variables de tipo `real` dadas como parámetros de la rutina. Si las variables `a` y `b` declaradas en un programa toman los valores, `13.4` y `-4.5`, respectivamente, tras la ejecución de la llamada a `cambio(a,b);` dentro de este programa que declara el uso de la unidad, las variables `a` y `b` deben tomar los valores `-4.5` y `13.4`.

```
unit examen;
```

```
end.
```

8. Completar el procedimiento `insertaEnCadena` para que **inserte** un carácter dentro de una **cadena** almacenada en una **variable dinámica** a la que apunta una variable de tipo `ptrCad`. Los parámetros de la rutina son la dirección de memoria de la variable dinámica, el carácter a insertar y la posición final del carácter en la cadena. Por ejemplo, si la variable dinámica tiene el valor inicial `'Cata'`, el carácter es `'n'` y la posición es `3`, entonces el valor final debe ser `'Canta'`. Si la posición final indicada es mayor que la longitud de la cadena original, ésta no debe ser modificada.

```
type ptrCad=^string;
procedure insertaEnCadena
```

9. Completar la función `esPar` para que **devuelva** el valor `true` si el **número** de **elementos** de una **lista** dinámica simplemente enlazada es **par** y `false` en caso contrario. El parámetro formal `p` de la función indica la dirección de memoria del primer elemento de la lista. Si la lista está **vacía** la función debe devolver el valor `true`.

```
type ptr=^elemento;
     elemento = record
        dato: string;
        sig: ptr
        end;
function esPar(p:ptr):boolean;
```

10. Escribir una función **recursiva** que **genere** una **lista** de `n` rectángulos a partir de un rectángulo dado. En la lista resultado, cada rectángulo es la **mitad derecha** (si el rectángulo es más ancho que alto) o la **mitad inferior** (en caso contrario) del rectángulo anterior. Por tanto, el primer rectángulo de la lista es el dado como parámetro. Cuando `n` es **0** debe producirse la lista **vacía**. Un rectángulo se define por su esquina superior izquierda (`TL`) y su esquina inferior derecha (`BR`). Los lados del rectángulo son paralelos a los ejes de coordenadas.

```
type punto = array['x'..'y'] of real;
     rectangulo = record
         TL, BR: punto;        { TL: Top Left; BR: Bottom Right }
         end;
     lista=^elemento;
     elemento = record
         dato: rectangulo;
         sig: lista;
         end;
function crear_rectangulos (rect: rectangulo; n: integer): lista;
```