INFORMATICA. EXAMEN Septiembre 2006. Pascal

Apellidos, Nombre

DNI

La puntuación total del examen es de 10 puntos. Para aprobar será necesario un mínimo de 5 puntos.

1) Consideremos los siguientes tipos para representar los datos relativos a alumnos:

```
CONST
     MAX = 100;
TYPE
     ALUMNO = RECORD
                 Nombre, Apellido : String;
                 Nota : Real
              END;
     VECTOR = ARRAY [1..MAX] OF ALUMNO;
```

a) (1.5 *puntos*) Escribe un subprograma que tome como parámetro un `VECTOR` y lo ordene usando el método de **ordenación por selección** de modo **DESCENDENTE,** es decir de mayor a menor, tomando como criterio de ordenación la nota del alumno.

b) (1.5 *puntos*) Escribe una función que tome como parámetro un `VECTOR` de alumnos ordenado **DESCENDENTE** y una nota, y realice una búsqueda binaria dentro del vector. Si se encuentra un alumno con dicha nota, la función debe devolver la posición de éste dentro del vector. Si no se encuentra la nota, la función debe devolver cero.

2) Considera el siguiente tipo para representar matrices de tamaño menor o igual a 30x30:

```
CONST
   TAM = 30;
TYPE
   Rango  = 1 .. TAM;
   Matriz = RECORD
              NFilas,NCols : Rango;
              M : ARRAY [Rango, Rango] OF REAL
            END;
```

donde los dos primeros campos indican el tamaño de la matriz representada, y el tercero almacena los elementos de la matriz.

a) (3 *puntos*) En álgebra lineal, una **matriz tridiagonal** es una matriz "casi" diagonal. De un modo más exacto, una matriz tridiagonal es una matriz cuadrada que tiene elementos distintos a cero solo en la diagonal principal, la primera diagonal sobre ésta, y la primera diagonal bajo la diagonal principal. Por ejemplo, la matriz

1 2 0 0
3 4 6 0
0 9 5 7
0 0 8 3

es tridiagonal. Escribe una función que tome como parámetro un `Matriz` y compruebe si es tridiagonal. La función **debe ser eficiente**, en el sentido de que, si la matriz no fuese tridiagonal, tan pronto como se pueda determinar esto se dejen de examinar los demás elementos de la matriz.

3) (4 *puntos*) La **conjetura débil de Goldbach** es la siguiente:

> *Todo* número impar *mayor que* 7 *puede expresarse como suma de tres números primos impares* (se puede emplear más de una vez el mismo número primo).

Para comprobar la conjetura, escribe un programa que lea del teclado un número impar mayor que 7 y muestre por pantalla todos los tríos de números primos impares tales que su suma igualan al número leído.

INFORMATICA. EXAMEN Septiembre 2006. Haskell

Apellidos, Nombre

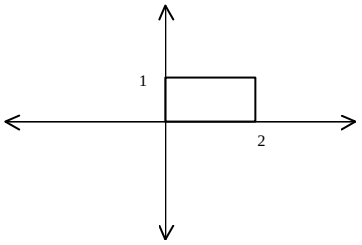
DNI

La puntuación total del examen es de 10 puntos. Para aprobar será necesario un mínimo de 5 puntos.

1) Considera los siguientes tipos para representar figuras geométricas

```
type CoordenadaX = Float
type CoordenadaY = Float
type Vértice   = (CoordenadaX, CoordenadaY)
data Polígono  = Poli [Vértice] deriving Show
```

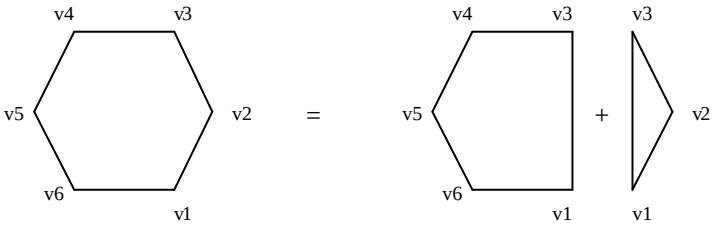
Por ejemplo, el rectángulo



puede ser representado por

```
r :: Polígono
r = Poli [(0.0,0.0), (0.0,1.0), (2,0,1.0), (2.0,0.0)]
```

a) (2 *puntos*) Supongamos que trabajamos tan solo con polígonos convexos. Un modo de calcular el área de un polígono lo ilustra el siguiente ejemplo



Es decir, se calcula el área que delimita el triángulo formado por los tres primeros vértices del polígono y se suma ésta al área del polígono que se obtiene al eliminar el segundo vértice. El proceso se repite hasta que el polígono considerado tenga menos de tres vértices, en cuyo caso el área correspondiente será cero. Sabiendo que el área de un triángulo de lados *a*, *b* y *c* puede ser calculada usando la formula de *Herón*

$$A = \sqrt{s(s-a)(s-b)(s-c)} \quad \textit{donde } s = \frac{1}{2}(a+b+c),$$

define una función que tome un `Polígono` convexo arbitrario y devuelva su área.

b) (1.5 *puntos*) Las funciones `all` y `and` se encuentran predefinidas del siguiente modo:

```
all p   = and . map p
and     = foldr (&&) True
```

Usando la función `all`, define una función que tome como argumento un `Polígono` y compruebe si es regular (si todos sus lados miden lo mismo).

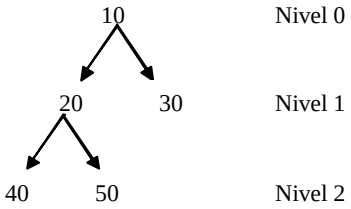
2) (2.5 *puntos*) La **conjetura débil de Goldbach** es la siguiente:

> *Todo* número impar *mayor que* 7 *puede expresarse como suma de tres números primos impares* (se puede emplear más de una vez el mismo número primo).

Para comprobar la conjetura, escribe una función en Haskell que tome un número impar mayor que 7 y devuelva una lista con todos los tríos de números primos impares tales que su suma igualan al número argumento.

3) Numeramos los niveles de un árbol a partir de cero (que corresponde a la raíz) de modo consecutivo.

a) (*1 pto*) Define una función *conNivelB* que tome como parámetro un árbol binario (tal como se han definido en clase) y devuelva una lista de pares, donde cada par es un valor en el árbol junto con el nivel en el que se encuentra. Por ejemplo, para el siguiente árbol



conNivelB a ==> [(10,0), (20,1), (30,1), (40,2), (50,2)]

b) (*1 pto*) Define una función *porNivelesB*, que tome un árbol binario y devuelva una lista de listas, donde cada sublista contiene los nodos en cada nivel. Por ejemplo, para el árbol anterior:

porNivelesB a ==> [[10], [20,30], [40,50]]

c) (*1 pto*) Define la función *porNiveles* similar a la anterior pero para árboles generales (tal como se han definido en clase).

4) (*1 pto*) Calcula el tipo polimórfico de la función `g` y da una definición equivalente **usando una lista por comprensión**:

```
g  = map .  ($)
```