INFORMATICA. EXAMEN FINAL Julio 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 personas:

```
CONST
     MAX = 100;
TYPE
     PERSONA = RECORD
                 Nombre, Apellido : String;
                 Edad : Cardinal
               END;
     VECTOR = ARRAY [1..MAX] OF PERSONA;
```

a) (1.5 *puntos*) Escribe un subprograma que tome como parámetro un `VECTOR` y lo ordene usando el método de **ordenación por inserción** de modo **DESCENDENTE,** es decir de mayor a menor, tomando como criterio de ordenación los apellidos de las personas. Para comparar alfabéticamente dos apellidos, puedes usar los operadores relacionales de Pascal (<, <=, >, etc.).

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

2) (3.5 *puntos*) El método de *Newton-Raphson* para encontrar un cero de una función *f* de reales en reales, que supondremos derivable, es el siguiente:

a) Escoger una aproximación inicial a la raíz, x_0
b) Calcular la siguiente aproximación utilizando la fórmula:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

c) **Si** $|x_{n+1} - x_n| <$ *epsilon* **entonces** x_{n+1} es una raíz. **En otro caso** volver al punto b).

Escribe una función en Pascal que tome como parámetros una función *f*, una aproximación inicial x_0 y un valor para *epsilon* y devuelva un cero de la función usando el método de *Newton-Raphson*. Define también los tipos y subprogramas adicionales que sean necesarios.

3) (3.5 *puntos*) La **conjetura de Goldbach** es la siguiente:

Todo número par *mayor que 2 puede escribirse como suma de dos* números primos (se puede emplear dos veces el mismo número primo).

Para comprobar la conjetura, escribe un programa que lea del teclado un número par mayor que 2 y muestre por pantalla todos los pares de números primos tales que su suma igualan al número leído.

INFORMATICA. EXAMEN FINAL Julio 2006. Haskell

Apellidos, Nombre

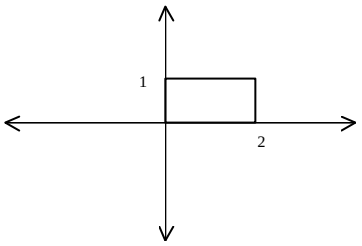
DNI

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
```

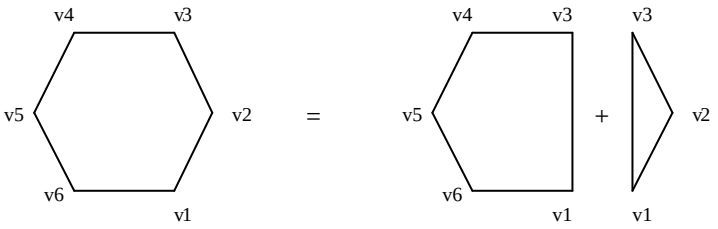
Por ejemplo, el rectángulo



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 *puntos*) El método de *Newton-Raphson* para encontrar un cero de una función *f* de reales en reales, que supondremos derivable, es el siguiente:

a) Escoger una aproximación inicial a la raíz, x_0
b) Calcular la siguiente aproximación utilizando la fórmula:

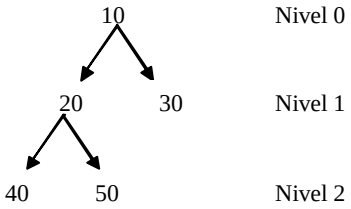
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

c) La primera aproximación x_i tal que $f(x_i) \cong 0$ es el cero buscado.

Escribe una función en Haskell que tome como parámetros una función *f*, una aproximación inicial x_0 y devuelva un cero de la función usando el método de *Newton-Raphson*. Define también las funciones y operadores adicionales que sean necesarios.

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

4) (*1 pto*) Define, **usando** `foldl`, la función `reverseMap`, con el siguiente comportamiento:

`reverseMap f [x1, x2, …, xn] ⇒ [f xn, f xn-1, …,f x1]`