

**Apellidos
y Nombre:**

D.N.I:

NOTA: Para aprobar son necesarios al menos 5 puntos.

1)

- a) (1.5 pts.) Define una función `esSubcadena` que tome dos cadenas de caracteres y devuelva `True` si la primera es una subcadena de la segunda, es decir, si aparece dentro de la segunda. Por ejemplo:

```
esSubcadena "la" "me gusta la paella" ==> True
esSubcadena "la" "el calamar"          ==> True
esSubcadena "tu" "el calamar"          ==> False
```

- b) (1pt.) Escribe una función `cifrarCon` que permita cifrar una cadena de caracteres para que quede irreconocible. La función tomará dos argumentos: un entero (entre 0 y 255) y una cadena de caracteres. Para cifrar la cadena, se desplazará hacia adelante el código ASCII de cada carácter de ésta tanto como indique el entero. Recuerda que hay 256 caracteres en la tabla ASCII (numerados del 0 al 255), por lo que el desplazamiento debe ser cíclico, es decir, el siguiente al número 255 será el 0. Por ejemplo,

```
cifrarCon 2 "casa" ==> "ecuc"
cifrarCon 3 "casa" ==> "fdvd"
```

- c) (0.5 pts.) Escribe una función `descifrarCon` que permita descifrar una cadena cifrada con el método anterior dados el entero a desplazar y la cadena cifrada. Para ello, se desplazará hacia atrás el código ASCII de cada carácter de la cadena tanto como indique el entero. El desplazamiento debe ser cíclico, es decir, el anterior al número 0 será el 255. Por ejemplo,

```
descifrarCon 2 "ecuc" ==> "casa "
```

- d) (2 pts.) El método que hemos usado para cifrar es fácil de romper, ya que dada una cadena cifrada, basta con probar a descifrarla con los 256 números posibles y ver si alguna de las cadenas obtenidas tiene sentido. Si además conocemos parte del texto que aparecía en la cadena original, podemos descartar todas las posibilidades descifradas que no lo contengan. Escribe **usando una lista por comprensión** una función `romper`, que tome parte del texto original y una cadena cifrada, y devuelva aquellas cadenas descifradas que contengan la parte del texto original dada. Por ejemplo:

```
romper "tu" (cifrarCon 1 "tu cifrado no es seguro")
==>
["tu cifrado no es seguro", "z{&iolxgju&tu&ky&ykm{xu}"]
```

ya que de las 256 cadenas descifradas posibles, sólo hay dos que contengan el texto "tu".

2) (1 pt.) Define una función `esPrimo` que tome como parámetro un entero y devuelva `True` si es un número primo.

3) (1 pt.) Define una función `estáDesordenada` que devuelva `True` si la lista que toma como parámetro NO está ordenada ascendentemente. Por ejemplo:

```
estáDesordenada [1,5,3,4] ==> True
estáDesordenada [1,3,5,6] ==> False
```

4) Consideremos la siguiente función polimórfica:

```
iter :: (Integer -> a -> a) -> a -> (Integer -> a)
iter f e = fun
  where
    fun 0          = e
    fun m@(n + 1) = f m (fun n)
```

Cuya propiedad universal es:

$$g = \text{iter } f \ e \Leftrightarrow \left\{ \begin{array}{l} g \ 0 = e \\ g \ m@(n+1) = f \ m \ (g \ n) \end{array} \right\}$$

Define usando `iter` las siguientes funciones:

a) (1.5 pts.) `palos` que dado un número natural x , construya una cadena de caracteres con x letras I:

```
palos 3 ==> "III"
```

b) (1.5 pts.) `listaPalos` que dado un número natural x , construya una lista con la sucesión decreciente de naturales desde x a 1, pero sustituidos por el número correspondiente de letras I:

```
listaPalos 5 ==> ["IIIII", "IIII", "III", "II", "I"]
```

INFORMATICA. EXAMEN Septiembre 2005.

Pascal

Apellidos
y Nombre:

D.N.I:

NOTA: Para aprobar son necesarios al menos 5 puntos.

1) (3 puntos) Definimos la secuencia modificada de *fibonacci* como la secuencia cuyos tres primeros términos son 0,1,2 y donde cualquier otro término se obtiene sumando los tres que le preceden, es decir, la secuencia es

0, 1, 2, 3, 6, 11, 20, ...

Escribe un programa que lea del teclado un número natural N y muestre por pantalla los N primeros términos de esta secuencia. Por ejemplo, si se introduce un 5 por teclado el programa mostrará por pantalla los cinco primeros términos:

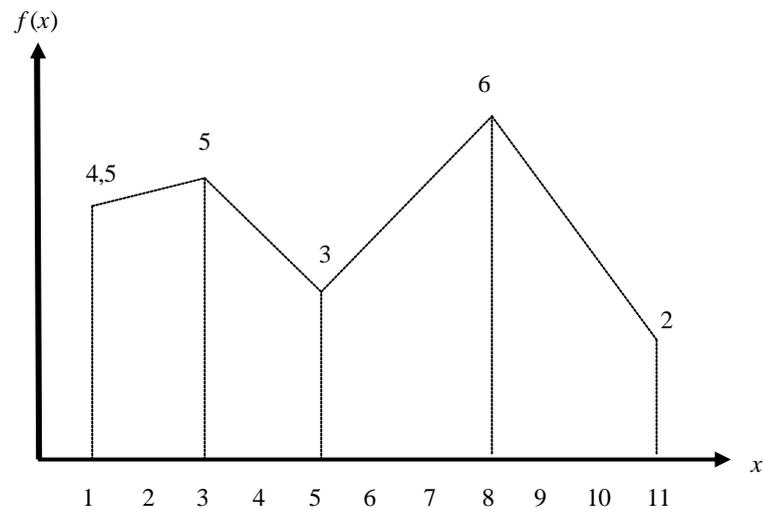
```
¿Cuántos términos?  
5  
Los términos son  
0  
1  
2  
3  
6
```

2) (3 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 los pares de números primos tales que su suma igualan al número leído.

3) (4 puntos) Se conocen los valores de una función f para ciertos valores de x . Por ejemplo:



Se quiere usar estos valores para estimar los valores de $f(x)$ para otros valores de x distintos, dentro del intervalo determinado por los puntos menor y mayor conocidos. Esta estimación se calculará mediante el *método de interpolación lineal*. Para ello, se trazará una recta entre cada dos puntos conocidos adyacentes y se considerará que el valor de f en los puntos intermedios viene dado por el valor de la recta.

Los valores de x y $f(x)$ estarán almacenados en un fichero llamado 'datos.num' que contiene valores reales correspondientes a cada pareja x y $f(x)$ conocidas. Por ejemplo, para la figura mostrada arriba, el fichero sería el siguiente:

'datos.num'

1.0
4.5
3.0
5.0
5.0
3.0
8.0
6.0
11.0
2.0

Escribe un programa que lea los datos del fichero y un punto z del teclado, y estime el valor de $f(z)$ correspondiente.

Recuerda que la ecuación de la recta que pasa por los puntos $(x_a, f(x_a))$ y $(x_b, f(x_b))$ es:

$$f(z) = f(x_b) + \frac{f(x_a) - f(x_b)}{x_a - x_b} \cdot (z - x_b)$$