22-10-2024

PRÁCTICA 1
Modelos avanzados de computación
EJERCICIOS SOBRE LISTAS EN HASKELL

wadadi sidelgaum limam.

Contenido

1. Introducción

EL objetivo de esta práctica es familiar con las funciones básicas sobre listas en haskell, para ello, se pide la realización de los siguientes ejercicios:

- **cambia_el_primero (a, b)**: Cambia el primero valor de la lista b por el valor de a.
- **cambia_el_n (a, n, b)**: Cambia el valor que se encuentra en la posición n de la lista b por el valor de a.
- **get_mayor_abs a**: Obtiene el mayor valor, en valor absoluto, de la lista a.
- **num_veces (a, b)**: Obtiene el número de veces que se repite el valor a en la lista b.
- **palabras_mayores_n (n, a)**: Obtiene un subconjunto de la lista a con los elementos con una longitud mayor a n.
- **es_palindroma palabra**: Comprueba si "palabra" es palíndroma.
- **es_primo (x):** Devuelve si el número introducido es primo o no.
- **sumparesimp []:** suma los pares y resta los impares de una lista.
- **show_foldr_suma_n (n):** muestra por pantalla los pasos del foldr (+) [1,2..n]
- **palindromas []:** comprueba son palíndromas todas las palabras de una lista.

Además, para completar la práctica se necesita la implementación de los siguientes 5 ejercicios:

- **sumparesimp_divisible(a,b):** Obtiene el resultado de suma los pares y resta los impares de los numeros divisibles por b de una lista a.
- **mos_busCadena(a,b):** muestra todos los valores intermedios de buscar la cadena invertida que conicida con b en una lista a.
- **n_divB(a,n,b):** Obtiene los n primeros elementos de la lista a divisible por b.
- **maxComun_dosprimeros(a):** Obtener el maximo comun divisor de los dos primeros pares de la lista a.
- **todos_primo(a):** Comprueba si todos los elemntos de la lista a son primos.

2. Ejercicios

2.1. Cambia el Primero

En este ejercicio tenemos cambiar el primer valor de la lista **b** por el valor **a**.

Definición de la cabecera de la función:

- Entradas: recibimos una lista de datos enteros sin decimales y un valor de entero.
- Salidas: devolvemos una lista de enteros con la modificación realizada.

```
cambia_el_primero:: Integral a => a-> [a] -> [a]

cambia_el_primero x b = x : (tail b)
```

Figura 1: Código de cambia_el_primero

Para cambiar el primer valor de la lista aplicamos la función tail**,** que nos proporciona una lista que contiene todos los elementos de la lista b excepto el primero, después con los dos puntos insertamos el valor a en la primera posición.

```
Main> cambia_el_n 2 4 [0,1,2,3,4,5,6,7]
[0,1,2,3,2,5,6,7]
```

Figura 2: Resultado de cambia el primero

2.2. Cambia el N

Este ejercicio consiste en sustituir el elemento que se encuentre en la posición **n** de la lista **b** por el valor de **a**.

Definición de la cabecera de función:

- Entradas: los parámetros de entrada son dos valores enteros y una lista de datos enteros.
- Salidas: como la salida la función nos devuelve una lista de enteros con el valor de la posición n actualizado.

```
cambia_el_n:: Int->Int->[Int]->[Int]

cambia_el_n x n b = take n b ++ [x] ++ drop (n+1) b
```

Figura 3: Código de cambia_el_n

Para la resolución del ejercicio, primero obtenemos los primeros elementos de la lista b que se encuentra en las posiciones desde 1 hasta n-1, para ello se utiliza la función take, después con el ++ se concatena con una lista que contenga solamente el valor de a, finalmente, se concatena el resultado de la concatenación anterior con el resto de los valores de la lista b, la función drop de n+1 elimina los elementos de la lista que estén en las posiciones desde 1 hasta n.

```
Main> cambia_el_n 2 4 [0,1,2,3,4,5,6,7]
[0,1,2,3,2,5,6,7]
```

Figura 4: Resultado de cambia el n

2.3. Obtener Mayor Absoluto

Este ejercicio consiste en obtener el mayor valor de una lista en valor absoluto.

Definición de la cabecera de función:

- Entradas: la entrada es una lista de datos enteros sin decimales.
- Salidas: la salida de la función es un parámetro de tipo entero.

```
get_mayor_abs:: Integral a=>[a]->a

get_mayor_abs b =  maximum (map abs b)
```

Figura 5: Código de get_mayor_absoluto

Para la resolución de este ejercicio, se utiliza la función abs que devuelve el valor absoluto de un número, para poder aplicar dicha función a todos los elementos de la lista se hace uso de la función map donde nos devuelve una lista con todos los valores de la lista b tras aplicarle la función abs, finalmente, para calcular el máximo de la lista generada se utiliza la función maximum.

```
Main> get_mayor_abs [0,-3,-45,6,7]
45
```

Figura 5: Resultado de get mayor absoluto

2.4. Número de Veces

Este ejercicio consiste en obtener el número de veces que se repite un valor en una lista.
Definición de la cabecera de función:

- Entradas: los parámetros de entrada son un valor entero y una lista de datos enteros.
- Salidas: como la salida la función nos devuelve un numero entero.

```
num_veces:: Int->[Int]->Int

num_veces a b = length (filter (==a) b)
```

Figura 7: Código de num_veces

Para filtrar todos los valores que coincide con el valor a, usamos la función filter, donde nos devuelve una lista con todos los valores que sean idénticos con a, finalmente, con la función length obtenemos la longitud de la lista anterior.

```
Main> num_veces 3 [0,2,3,3,3,3,3,3,35]
6
```

Figura 8: Resultado_de num veces

2.5. Palabras mayores que N

Este ejercicio se trata de encontrar las palabras que tengan una longitud mayor que n de una lista pasada por parámetro.

Definición de la cabecera de la función:

- Entradas: recibimos una lista de datos de tipo string y un valor de entero.
- Salidas: devolvemos una lista de string.

```
palabras_mayores_n:: Int -> [[a]] -> [[a]]

palabras_mayores_n n b = filter (\x -> length x>n) b
```

Figura 9: Código de palabras_mayores

Para la resolución, se utiliza la función filter, donde le indicamos como función de entrada para toda x, o sea cada elemento de la lista, filtra la que tenga longitud mayor que n.

```
Main>  palabras_mayores_n 3 ["aaaaa","ffff","dwfq3wf","s","33"]
["aaaaa","ffff","dwfq3wf"]
```

Figura 10: Resultado de palabras mayores

2.6. Es palindroma

Este ejercicio se trata de comprobar si una palabra es palíndroma.

Definición de la cabecera de la función:

- Entradas: recibe una cadena de String.
- Salidas: devuelve un booleano.

```
es_palindroma:: String->Bool

es_palindroma x = x == (reverse x)
```

Figura 11: Código de es palindroma

Para la resolución, se compara la cadena de entrada con la cadena invertida mediante la función reverse.

```
Main> es_palindroma "abad daba"
True
Main> es_palindroma "Hols"
False
```

Figura 12: Resultado de es palindroma

2.7. Palindromas []

Este ejercicio se trata de comprobar si todas las palabras son palíndromas.

Definición de la cabecera de la función:

- Entradas: recibe una lista de string.
- Salidas: devuelve un booleano.

```
palindromas:: [String] -> Bool

palindromas x = and (map (es_palindroma) x)
```

Figura 13: Código de palindromas

Para la resolución, se ha uso de la función anterior aplicándola a cada elemento de la lista con la función map.

```
Main> palindromas ["ana","ala"]
True
Main> palindromas ["holaa","ala"]
False
```

Figura 14: Resultado de palindromas

2.8. Sumar pares restar impares

Este ejercicio consiste en devolver el resultado de sumar todos los elementos pares de una lista restándole todos los elementos impares de la misma lista.
Definición de la cabecera de la función:

- Entradas: recibe una Lista de enteros.
- Salidas: devuelve un entero.

```
sumparesimp:: [Int]->Int

sumparesimp a = sum (filter even a) - sum (filter odd a)
```

Figura 15: Código de Sumpresimp

Para la resolución, se filtra todos los elementos pares y impares de la lista con las funciones even y odd, y mediante la función sum se suman todos los elementos del resultado de cada filtración, finalmente, se restan para devolver el resultado.

```
Main> sumparesimp [1,2,3,4,5,6]
3
```

Figura 16: Resultado de Sumparesimp

2.9. Es primo

Este ejercicio se trata de comprobar si un número es primo o no.

Definición de la cabecera de la función:

- Entradas: recibe un entero.
- Salidas: devuelve un booleano.

```
esprimo:: Int-> Bool

esprimo b = not( or (map(\x -> mod b x == 0) [2..(b `div` 2)]))
```

Figura 17: Código de esprimo

Para la resolución, se crea una lista que empieza desde el numero 2 hasta llegar a la mitad del número pasado por parámetro, con la función map se comprueba que el numero b no es divisible por cada elemento de dicha lista, la función or sirve para comprobar que haya algún elemento de la lista es múltiplo de b y el not para negar la salida del or ya que si no existe ningún elemento que sea múltiplo de b el or nos devuelve false.

```
Main> esprimo 11
True
Main> esprimo 15
False
```

Figura 18: Resultado de esprimo

3. Ejemplos

3.1. Sumar pares y restar impares divisibles por b

Este ejercicio consiste en obtener el resultado de sumar los pares y resta los impares de los numeros divisibles por b de una lista a.

Definición de la cabecera de la función:

- Entradas: recibimos una lista de enteros y un valor entero.
- Salidas: devolvemos un valor enetro.

```
sumparesimp_divisible:: [Int]->Int->Int

sumparesimp_divisible a b = sumparesimp (filter(\x -> mod x b == 0) a)
```

Figura 19: Código de Sumparesimp_divisible

Para la resolución, se ha hecho uso de la función sumpresimp sobre una lista obtenida al filtrar los numero divisibles por b en la lista a.

```
Main> sumparesimp_divisible [4,3,5,6,7,9,8] 2
18
Main> sumparesimp_divisible [4,3,5,6,7,9,12] 3
6
```

Figura 20: Resolución de Sumparesimp_divisible

3.2. Mostrar Cadenas

Este ejercicio consiste en comprobar si los elementos de una lista coinciden con la cadena invertida b.

Definición de la cabecera de la función:

- Entradas: recibimos una lista de string y un dato string.
- Salidas: devolvemos una lista de booleano.

```
mos_busCadena:: [String]->String->[Bool]
cad_invertida s = reverse s
mos_busCadena a b = map (\x -> x==cad_invertida b) a
```

Figura 21: Código de mos_busCadena

Para la resolución, se comprueba que para cada elemento de la lista a coincide con el reverse de b.

```
Main> mos_busCadena ["aloh","ana","abad"] "hola"
[True,False,False]
```

Figura 22: Resolución de mos_busCadena

3.3. Número de elementos divisible por b

Este ejercicio se trata de obtener los n primeros elementos de la lista a divisible por b.

Definición de la cabecera de la función:

- Entradas: recibimos una lista de entero y dos valores enteros.
- Salidas: devolvemos una lista de enteros.

```
n_divB:: [Int]->Int->Int->[Int]

n_divB a n b = take n (filter (\x -> mod x b ==0) a)
```

Figura 23: Código de n_divB

Para la resolución, se filtra los elementos de la lista a que son divisibles por b, y después con la función take se escoge n elementos de la lista filtrada.

3.4. Máximo Común divisor

Este ejercicio se trata de obtener el maximo comun divisor de los dos primeros pares de la lista a
Definición de la cabecera de la función:

- Entradas: recibimos una lista de entero.
- Salidas: devolvemos un valor entero.

```
maxComun_dosprimeros:: [Int]->Int

maxComun_dosprimeros a = gcd (head (filter even a)) (head(tail (filter even a)))
```

Figura 25: Código de maxComun_dosprimeros

Para la resolución, se hace uso de la función gcd que nos permite calcular el máximo común divisior , los elementos que se la pasan son el primer elemento par de la lista , y el segundo quitando la cabecera del filtrado de los elementos pares de la lista.

```
Main> maxComun_dosprimeros [1,44,38,4]
2
```

Figura 26: Resolución de maxComun_dosprimeros

4.5. Comprobar Todos primo

Este ejercicio consiste en comprobar si todos los elementos de la lista a son primos.
Definición de la cabecera de la función:

- Entradas: recibimos una lista de entero.
- Salidas: devolvemos un booleano.

```
todos_primo:: [Int]->Bool

todos_primo a = and (map esprimo a)
```

Figura 27: Código de todos_primo

Para la resolución, se hace uso de la función esprimo implementada anteriormente y las funciones map para aplicarla a todos los elementos de la lista y la función and para comprobar que todos sean verdaderos.

```
Main> todos_primo [1,2,3,4,5]
False
Main> todos_primo [2,3,7,11]
True
```

Figura 28: Resolución de todos_primo