HASKELL

1. Definir las siguientes funciones y determinar su tipo:

   - `apply`, que toma una función y un valor, y devuelve el resultado de aplicar la función al valor dado .
   - `first`, que toma un par ordenado, y devuelve su primera componente.
   - `sign`, la función signo.
   - `abs`, la función valor absoluto (usando `sign` y sin usarla).
   - `pot`, que toma un entero y un número, y devuelve el resultado de elevar el segundo a la potencia dada por el primero.
   - `xor`, el operador de disyunción exclusiva.
   - `max3`, que toma tres números enteros y devuelve el máximo entre ellos.
   - `swap`, que toma un par y devuelve el par con sus componentes invertidas.

2. Indicar si cada una de las siguientes expresiones está o no bien formada. En caso de que lo esté determinar el valor que denota, en caso contrario especificar si el error es sintáctico o de tipos:

   - `if (five 'a' < 1) then 5 else pot 4 2`
   - `if true then false else true where false = True; true = False`
   - `if if then then else else`
   - `False == (5 >= 4)`
   - `1 < 2 < 3`
   - `1 + if ('a' < 'z') then - 1 else 0`
   - `if fst p then fst p else snd p where p = (True,2)`
   - `if fst p then fst p else snd p where p = (True,False)`

3. Definir una función que dada una lista `xs` y un elemento `z`, y recorriendo la lista `xs` una sola vez, obtenga el par formado por:

   a) La lista que resulta de quitar en `xs` todas las apariciones de z.
   b) El número de apariciones de `z` en `xs`.

4. Escriba una función `noDoble` que dada una cadena de caracteres retorne como resultado la misma cadena de caracteres eliminando los caracteres repetidos dos o más veces de manera consecutiva.

   ```
   E: noDoble "abccccfabaddeff"
   S: abcfabadef
   ```

5. Sea la siguiente definición de función:

   ```
   mifun s = foldr op 0 s
             where op x r = head x + r
   ```

a) ¿Cuál es el tipo de datos de `mifun`?
b) ¿Cuál es el resultado de evaluar `mifun [[1,2,3],[2,4],[3,5]]`?
c) ¿Cuál de las definiciones siguientes es una definición alternativa equivalente?

```
(Def 1) mifun s = sum (map head s)
(Def 2) mifun s = map op s
                  where op x r = head x + r
(Def 3) mifun [] = 0
                  mifun (x:xs) = x + mifun xs
```

6. Generalizar la función:

```
iSort [] = []
iSort (x:xs) = insertar x (iSort xs)
     where insertar z [] = [z]
           insertar z (x:xs)
               | x<z = (x:(insertar z xs) )
               | otherwise = (z:x:xs)
```

Para obtener una función de orden superior `iSortGen` que tome como argumento (además de la lista) un predicado binario que represente el orden con respecto al cual ordenará. En particular, debe cumplirse que `iSort = iSortGen (<)`
También debe ocurrir que:

```
iSortGen (>) [2,3,1,5,3] => [5,3,3,2,1]
iSortGen (<) ["casa","perro","coche"] =>
                      ["casa","coche","perro"]
iSortGen (==) [2,3,1,5,3] => [2,3,1,5,3]
iSortGen (==) ["casa","perro","coche"] =>
                      ["casa","perro","coche"]
```

Además, se pide:

a) Decir cuál es el tipo de `iSort` y el de `iSortGen`.
b) ¿Cuál sería el resultado de `iSortGen (>) [(1,2,3), (1,2,5), (3,4,4)]`?
c) Definir usando `iSortGen` una función `OrdTrip` que ordene listas de triples por orden decreciente de su tercera componente. Por ejemplo:

```
OrdTrip [(1,2,3),(1,2,5),(3,4,4)]
     =>[(1,2,5),(3,4,4),(1,2,3)]
```

7. Defina una función `esta` que controle si existe cierto elemento en una lista de elementos. Defina la función de las siguientes maneras:

a) Tome todos los elementos iguales al elemento buscado y coloque éstos en una lista. Compruebe después si esta lista está vacía o no.

b) Haga una nueva lista en la que todos los elementos iguales al elemento buscado sean reemplazados por 1 y los otros elementos por 0. Sume los elementos de la lista resultante y compruebe si el resultado es igual a 0 o no.
c) Compruebe para cada elemento si es igual al elemento buscado o no. Después compruebe si alguna de estas verificaciones se evaluó en `True`.

8. Escriba una función `posiciones` que devuelva una lista de índices de las posiciones de un elemento determinado en una lista de elementos. Por ejemplo:

```
E:posiciones 4 [1,4,3,7,4,2]
S:[2,5]

E:posiciones [3,5] [[3,6],[2,5]]
S:[]
```

9. Escriba una función `elimDobles`, que dada una lista finita, devuelva una nueva lista, con solamente una ocurrencia de cada elemento de la lista original.

10. Escriba una función `dividir`, que dados una lista no decreciente `xs` y un elemento `x`, devuelva una dupla de dos listas `(ys,zs)`, con `xs = ys ++ zs`, donde todos los elementos de `ys` sean menores o iguales que `x`, y todos los elementos de `zs` sean mayores que `x`.

```
dividir :: a -> [a] -> ([a],[a])
```

11. Escriba una función `insertar`, que dados una lista no decreciente `ys` y un elemento `y`, devuelva una lista no decreciente igual a `ys` más el elemento e insertado en el lugar correspondiente.

12. Escriba una función `unico`, que dada una lista devuelva una lista que contenga exactamente los elementos que se encuentran solamente una vez en la lista dada. Por ejemplo:

```
E: unico "Cuales son las letras unicas en esta frase?"
S: Coicf?
```

13. Escriba una función `multiplicar`, que dado un numero positivo `m` (m < 10), y una lista de números *n*, (que representa un número), retorne una lista que represente la multiplicación `n*m`. Ejemplos:

```
E: multiplicar 3 [4,8,9]
S: [1,4,6,7]
E: multiplicar 5 [9,9,9,1,4,6]
S: [4,9,9,5,7,3,0]
```

14. Una lista de enteros es palíndrome, si cada uno de los números que la conforman son palíndromes y además la lista se puede leer de derecha a izquierda y viceversa, es decir,

[1,2,3,2,1] es palíndrome, pero [1,21,3,21,1] no, ya que 21 no es palíndrome, [1,2,3,4,5] tampoco porque [1,2,3,4,5] es diferente de [5,4,3,2,1]. Se quiere que Ud. Implemente un predicado `getPals(L)`, que reciba como entrada una lista de listas de enteros (`[[Int]]`) y retorne una lista de listas con todas aquellas listas que sean palíndromes.

15. Escriba programas en el lenguaje de programación Haskell que le permitan resolver cada uno de los siguientes problemas:

    a) Dada dos listas A y B, verificar si eliminando elementos de la lista A se puede obtener la lista B.
    b) Dada dos listas A y B, que representan conjuntos, implemente la intersección y unión de los mismos.
    c) Dada una lista que representa una matriz, calcule la transpuesta de esta.

16. Sean las siguientes definiciones de tipo `f :: c → d y g :: a → b → c` y la función `h x y = f (g x y)`. Cuáles de las siguientes definiciones de `h` son equivalentes a la dada:

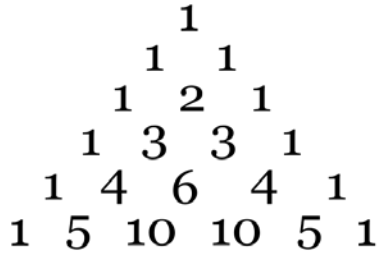
    a) `h = f · g`
    b) `h x = f · (g x)`
    c) `h x y = (f · g) x y`

17. Diremos que una matriz es palíndrome según el eje horizontal, si todas sus columnas son palíndromos (se leen igual de izquierda a derecha que al revés). Por ejemplo, las siguientes matrices son simétricas según el eje horizontal:

```
1 2 2 1                    1 5 3
3 7 9 9          ->        2 2 4
3 7 9 9                    1 5 3
1 2 2 1
```

    Escribe una función `esPalHor`, que dada una Matriz compruebe si es palíndrome según el eje horizontal.

18. El triángulo de Pascal, también conocido como triángulo de Tartaglia, es un triángulo de números enteros, infinito y simétrico cuyas diez primeras líneas han sido representadas en la figura:



Se desea que implemente una función Pascal que muestre las N primeras filas del Triángulo de Pascal.

19. Escribir una función `m_Sublista:: Eq a => [a] -> a -> [[a]]` que reciba una lista de elementos de cualquier tipo y un elemento "e", del mismo tipo de dato, obtener un lista de lista de elementos, donde cada sublista esta determinada por los elementos que se encuentran entre los elementos que sean iguales a "e".

```
E: m_Sublista [1, 2, 3, 1, 4, 2, 1, 5,7] 1
S: [[], [2,3], [4,2], [5,7]]

E: m_Sublista [1, 2, 3, 1, 4, 2, 1, 5, 7,2] 2
S: [[1], [3, 1,4], [1, 5,7], []]
```

20. Realice una función `m_Matriz :: [[Int]] -> [[Int]] -> [[Int]]` que permita realizar la multiplicación de dos matrices NxM y MxO. (Sugerencias: Realice primero la multiplicación de una matriz por un vector. La matriz de entrada es valida).

```
E: m_Matriz [[1,2,3,4],[5,6,7,8],[9,10,11,12]] [[1,5,10],
   [2,6,11],[3,7,12],[4,8,13]]

S: [[30, 70,120], [70, 174,304], [110, 278,488]]
```

21. Escriba una función `M_Lista`, que dada una lista de entrada L de números enteros, con valores repetidos, retorne una lista de sublistas (sin elementos repetidos), donde cada una de las cuales esté formada por los siguientes componentes:

a) El número entero de la lista L.
b) El número de veces que aparece dicho número en la lista L
c) La sumatoria de las ocurrencias del número anterior en la lista L.

```
E: M_Lista  [5,7,7]
S: [[5,1,5] , [7,2,14]]

E: M_Lista  [1,2,3,2,3,4]
S: [[1,1,1] , [2,2,4],[3,2,6],  [4,1,4]]
```

22. Escriba una función `prodCart` que dado dos listas retorne una lista de duplas resultante de realizar el producto cartesiano entre las dos listas.

```
E: prodCart  [1,2,3] [1,2]
S: [(1,1),(1,2),(2,1),(2,2),(3,1),(3,2)]
```

23. Escriba una función `permutaciones` que dado una lista retorne una lista de listas con todas las permutaciones de los valores en la lista.

```
E: permutaciones  [1,2,3]
S: [[1,2,3],[1,3,2],[2,3,1],[2,1,3],[3,1,2],[3,2,1]]
```

24. Dado una Lista de Listas cualquiera, realice una función que retorne una Lista de Listas que cumpla con la condición: La sublista Xi no debe tener elementos de la sublista Xj con i > j.

```
E: [[1,2,3],[2,4],[3,4,5,6]]
S: [[1,2,3],[4],[5,6]]
```

25. Escriba una función que dado la estructura de datos `Arbol` permita determinar si es un heap. Un árbol es heap si cumple las siguientes condiciones:
    - El árbol debe ser binario, es decir cada nodo tiene grado 2.
    - Los nodos y hojas poseen un valor entero.
    - Para cada nodo, todos sus hijos deben tener un valor mayor o igual que el suyo.

    La estructura `Arbol` a utilizarse se presenta a continuación:

```
data Arbol= Hoja Int | Nodo Int Arbol Arbol

E: heap (Nodo 2 (Nodo 3 (Hoja 4) (Hoja 7)) (Hoja 6))
S: True

E: heap (Nodo 3 (Nodo 4 (Hoja 3) (Hoja 6)) (Hoja 7))
S: False
```