**1. Fatorial: recebe um número natural e retorna o seu fatorial.**

fatorial n = if(n>1)

then n\*fatorial(n-1)

else n

fatorial:: Integer -> Integer

fatorial 0 = 1

fatorial n | n > 0 = n \* fatorial (n-1)

fatorial n = product [1..n]

**2. Fibonacci: recebe um número inteiro positivo e retorna o n-ésimo elemento da sequência de Fibonacci.**

fibonacci n = if(n>1)

then fibonacci (n-1)\* fibonacci (n-2)

else n

fibonacci 0 = 0

fibonacci 1 = 1

fibonacci n = fibonacci (n - 2) + fibonacci (n - 1)

**3. Intercala: recebe duas listas e retorna outra lista com os elementos das listas originais intercalados.**

*Ex: (intercala '(1 2 3) '(8 9)) ==> (1 8 2 8 3) (intercala '() '(1 2 6)) ==> (1 2 6)*

let lista1 = [1,2,3]

let lista2 =[8,9]

intercalate lista1 lista2

**4. uniao: recebe duas listas que não contenham elementos repetidos e retorna uma nova com todos os elementos das duas listas originais (sem repetição)**

*Ex: (uniao '(3 6 5 7) '(2 9 7 5 1)) ==> (3 6 5 7 2 9 1)*

let lista1 = [3,6,5,7]

let lista2 =[2,9,7,5,1]

lista1 `union` lista2

**5. Intersecção: recebe duas listas sem elementos repetidos e retorna uma lista com os elementos que são comuns às duas**

*Ex: (interseccao '(3 6 5 7) '(9 7 5 1 3)) ==> (3 5 7)*

let lista1 = [3,6,5,7]

let lista2 =[9,7,5,1,3]

lista1 `intersect` lista2

**6. Sequencia: recebe dois numeros naturais n e m, e retorna uma lista com n elementos, onde o primeiro é m, o segundo é m+1, etc...**

*Ex: (sequencia 0 2) ==> ( ) (sequencia 3 4) ==> (4 5 6*) *5 6 -> 6 7 8 9 10*

n,m

take n [m,m+1..m+n]

**7. Faça um programa que dada uma lista, retorne uma tupla listalista (de inteiros) onde a lista da esquerda contém os números impares e a lista da direita os números pares**

*ex: func :: [Int] -> ([Int],[Int]) [1,2,3,4,5,6,7] => ([1,3,5,7],[2,4,6])*

let lista1 = [1,2,3,4,5,6,7]

partition ('mod' 2 !=0) lista1

**8. O conjunto de todos os subconjuntos de um segundo conjunto é denominado conjuntos das partes desse segundo conjunto. Faça um programa que encontra o conjunto das partes de uma lista.**

*Ex: partes [2,3,2,31] = [[],[2],[3],[31],[2,2],[2,3],[2,31],[3,31],[2,2,3],[2,2,31],[2,3,31],[2,2,3,31]]*

**9. Compactar: recebe uma lista de números e transforma todas as repetições em sub-listas de dois elementos: sendo o primeiro elemento o número de repetições encontradas e o segundo elemento é o número que repete na lista original. Os números que não repetem na lista original não devem ser alterados.**

*Ex: (compactar '(2 2 2 3 4 4 2 9 5 2 4 5 5 5)) ==> ((3 2) 3 (2 4) 2 9 5 2 4 (3 5))*