Exercises
List Comprehensions

* Exercise 1

Using list comprehension define the list of cubes of the values between (and including) 1 and 10.

```
squares = [(1,1),(2,4),(3,9),(4,16),(5,25),(6,36),
                (7,49),(8,64),(9,81),(10,100)]
```

* Exercise 2

Using list comprehension define the following list (note that the second element in the 2-tuple is always 1.

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

** Exercise 3

Write down the values as defined in the following lists l1, l2, l3. Check your answers.

```
f1 :: [(Int, Int)]
f1 = [(x, y) | x <-[1..3], y<- [4..5]]

f2 :: [(Int, Int)]
f2 = [(x, y) | y<- [4..5], x <-[1..3]]

f3 :: [(Int, Int)]
f3 = [(y, x) | x <-[1..3], y<- [4..5]]
```

** Exercise 4

Given the following definition of

```
isEven :: Integer -> Bool
isEven n = (n `mod` 2 == 0)
```

Write down the values as defined in the following list: Check your answer.

```
[2*n | n <- [2,4,7], isEven n, n>3]
```

** Exercise 5

Give a definition of a function

```
doubleAll :: [Integer] -> [Integer]
```

which doubles all the elements of a list of integers.

** Exercise 6

Give a definition of a function

```
capitalize :: String -> String
```

which converts all small letters in a String into capitals.
Hint:You can use the following function (having imported Data.Char):

```
import Data.Char
toupper :: Char -> Char
```

** Exercise 7

Using a list comprehension, give an expression that calculates the sum of

$$\sum_{i=1}^{i=100} i^2$$

** Exercise 8

Using a list comprehension,write a function sigma'

```
sigma' :: Int-> Int
```

that takes an integer n and calculates

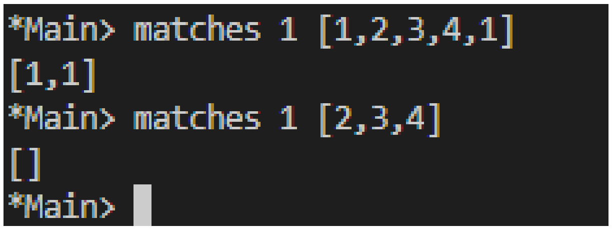
$$\sum_{i=1}^{i=n} i^2$$

**** Exercise 9

Define the function

```
matches :: Integer -> [Integer] -> [Integer]
```

which picks out all occurences of an integer in a list. For instance:



```
*Main> matches 1 [1,2,3,4,1]
[1,1]
*Main> matches 1 [2,3,4]
[]
*Main>
```

Using matches or otherwise, define a function

```
elem':: Integer -> [Integer] -> Bool  --elem is already defined in Prelude
```

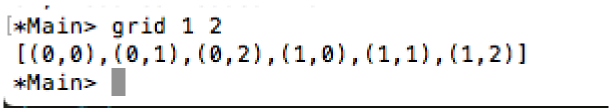
which is True is the Integer is an element of the list, and False otherwise.

*** Exercise 10

Suppose that a *coordinate grid* of size $m \quad x \quad n$ is given by the list of all pairs (x, y) of integers such that $0 \leq x \leq m$ and $0 \leq y \leq n$. Using a list comprehension, define a function:

```
grid :: Int -> Int -> [(Int, Int)]
```

that returns a coordinate grid of a given size. For example:



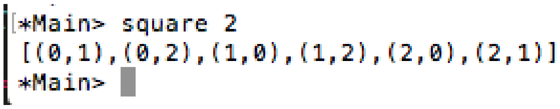
```
*Main> grid 1 2
[(0,0),(0,1),(0,2),(1,0),(1,1),(1,2)]
*Main>
```

*** Exercise 11

Using a list comprehension and the function **grid** above, define a function

```
square :: Int ->  [(Int, Int)]
```

that returns a coordinate square of size n, excluding the diagonal from $(0, 0)$ to (n, n). For example:



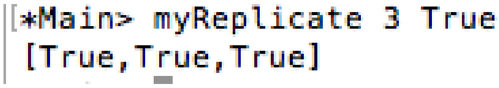
```
*Main> square 2
[(0,1),(0,2),(1,0),(1,2),(2,0),(2,1)]
*Main>
```

*** Exercise 12

In a similar way to the function ***length***, show how the library function

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

that produces a list of identical elements can be defined using list comprehension. (Call your version **myReplicate**) For example:



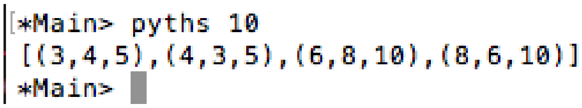
```
*Main> myReplicate 3 True
[True,True,True]
```

*** Exercise 13

A triple (x, y, z) of positive integers is called pythagorean if $x^2 + y^2 = z^2$. Using a list comprehension, define a function

```
pyths :: Int -> [(Int,Int,Int)]
```

that returns a list of all such triples whose components are at most a given limit. For example



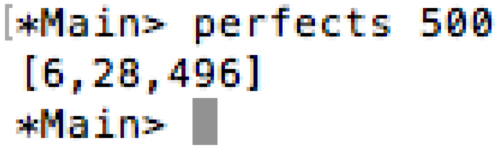
```
*Main> pyths 10
[(3,4,5),(4,3,5),(6,8,10),(8,6,10)]
*Main>
```

**** Exercise 14

A positive integer is perfect if it equals the sum of all of its factors, excluding the number itself. Using a list comprehension and the function **factors**, define a function

```
perfects :: Int -> [Int]
```

that returns the list of all perfect numbers up to a given limit. For example:



```
*Main> perfects 500
[6,28,496]
*Main>
```

Solutions

Solution to Exercise 1

```
cubes :: [Int]
cubes = [x^3 | x <- [1..10]]
```

Solution to Exercise 2

```
myConstFunc :: [(Int, Int)]
myConstFunc = [(x, 1)| x <- [1..5]]
```

Solution to Exercise 3

```
 l1 = [(1,4),(1,5),(2,4),(2,5),(3,4),(3,5)]
 l2 = [(1,4),(2,4),(3,4),(1,5),(2,5),(3,5)]
 l3 = [(4,1),(5,1),(4,2),(5,2),(4,3),(5,3)]
```

Solution to Exercise 4

```
[8]
```

Solution to Exercise 5

```
doubleAll :: [Integer] -> [Integer]
doubleAll ns = [n*2 | n<-ns]
```

Solution to Exercise 6

```
capitalize :: String -> String
capitalize xs = [toUpper(c) | c<- xs ]
```

Solution to Exercise 7

```
sigma :: Int
sigma = sum [x^2 | x <- [1..100]]
```

Solution to Exercise 8

```
sigma' :: Int-> Int
sigma' n = sum[x^2 | x <- [1..n]]
```

Solution to Exercise 9

```
elem':: Integer -> [Integer] -> Bool
elem' x xs = matches x xs  /= []
```