Solution Types and Polymorphism

- Exercises are given every week on the PL page of the SCG website (http://scg.unibe.ch/teaching/pl)
- Solutions to each assignment must be sent to **mohammadreza.hazhirpasand@inf.unibe.ch**
- The solutions of the assignments are to be delivered before every Thursday at 11 PM. Solutions handed in later than the specified time will not be accepted. In case of serious reasons send an e-mail to **mohammadreza.hazhirpasand@inf.unibe.ch**

Exercise (6 points)

1. Infer types of the functions `factors`, `isPerfect`, `mH` , and `insert` and say whether they are monomorphic or polymorphic functions. Please justify your answer. (3 pts)

```
mod :: Int -> Int -> Int
factors n = [x | x <- [1..n-1], mod n x == 0 ]
isPerfect n = sum (factors n) == n

insert _ n [] = [n]
insert 0 n l = n:l
insert i n (x:xs) = x : insert (i-1) n xs

mH (a, b, c) = c
```

Answer:

`factors :: Int -> [Int]`
since both `n` and `x` are arguments of the function `mod` which accepts only the `Int` arguments

`isPerfect :: Int -> Bool`
since `n` is an argument of the function `factors` which accepts only the `Int` arguments,
and `== :: Eq a => a -> a -> Bool`

Both functions are monomorphic.

`insert :: Int -> a -> [a] -> [a]`
since
`insert _ n l = [n] => insert :: a->b->c->[b]`
`insert 0 n l = n:l => insert :: Int->b->[b]->[b]`
The `insert` function is polymorphic.

`mH (a, b, c) = c`

`mH :: (x, y, z) -> z`
mH is polymorphic since the three elements a, b, and c may be of any type.

2. Some functions can be used with different argument types, but rarely with any type. Consider the square function, and identify which type is invalid. Please justify your answer. (1 pts)

```
square :: Int -> Int
square :: Float -> Float
square :: Char -> Char
square :: Double -> Double
```

Answer:

Invalid type for square is
square :: Char - Char
The compiler doesn't know how to multiply two Chars.
Multiplication is needed to calculate the square.

3. Define a function to calculate the circumference of a circle and the area of a rectangle. It is required to define **a union type** containing the necessary parameters of a circle and a rectangle. (2 pts)

```
Hints :
   Circumference of a circle : 2 * 3.14 * r
   Area of a rectangle : l  * w
```

Answer:

```
data Shapes = Circle Float | Rectangle Float Float
calit ::  Shapes -> Float
calit (Circle r) = 2 * pi * r
calit (Rectangle a b) = a * b
```