Exercise 3.5

Give two different definitions of the nAnd function
`nAnd :: Bool -> Bool -> Bool`
which returns the result `True` except when both arguments are `True`

Exercise 3.8

Explain the effect of the function defined here:

```
mystery :: Integer -> Integer -> Integer -> Bool
mystery m n p = not((m==n) && (n==p))
```

Exercise 3.14

Give definitions of the functions `min2` and `minThree` which calculate the minimum of two and three integers, respectively. (Note: `min` is a built-in function from the `Prelude`, threfore we choose the name `min2`)

```
min2 :: Int -> Int -> Int
minThree :: Int -> Int -> Int -> Int
```

Exercise 3.17

Define the function `charToNum` which converts a digit like '8' to its value 8. The value of non-digits should be taken to be 0.

Exercise 3.22

Write a function `numberNDroots` that given the coefficients of the quadratic `a`, `b` and `c`, will return how many (real) roots the equation has. You may assume that `a` is non-zero.

Exercise 3.23

Using your answer to the last question, write a function
`numberRoots :: Float -> Float -> Float -> Integer`
that given the coefficients of the quadratic `a`, `b` and `c`, will return how many (real) roots the equation has. In the case that equation has every number a root you should return the result 3.

Exercise 3.24

The formula for the roots of a quadratic is

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Write definitions of the functions

```
smallerRoot :: Float -> Float -> Float -> Float
largerRoot :: Float -> Float -> Float -> Float
```

which return the smaller and larger real roots of the quadratic. In the case that the equation has no real roots or has all values as roots you should return zero as result of each of the functions.

Exercise 4.17

Define the function `rangeProduct` which when given the natural numbers `m` and `n` returns the product `m*(m+1)*...*(n-1)*n`. The function should return 0 when `n` is smaller than `m`.

Exercise 4.18

As `fac` is a special case of `rangeProduct`, write a definition of `fac` which uses `rangeProduct`

Exercise 4.32

Suppose we have to raise 2 to the power `n`. If `n` is even, `2*m` say, then

$$2^n = 2^{2m} = (2^m)^2 .$$

If `n` is odd, `2*m+1 say`, then

$$2^n = 2^{2m+1} = (2^m)^2 \cdot 2.$$

Give a recursive function to compute 2^n which uses these insights.

Exercise 5.1

Give a definition of the function
`maxOccurs :: Integer -> Integer -> (Integer, Integer)`
which returns the maximum of two integers, together with the number of times it occurs. Using this, define the function
`maxThreeOccurs :: Integer -> Integer -> Integer -> (Integer, Integer, Integer)`
which does a similar thing for three arguments.

Exercise 5.18

Give a definition of the function
`doubleAll :: [Integer] -> [Integer]`
which doubles all the elements of a list of integers.

Exercise 5.21

Define the function
`matches :: Integer -> [Integer] -> [Integer]`
which picks out all occurrences of an integer `n` in a list. For instance:
`matches 1 [1,2,1,4,5,1]` $\rightsquigarrow$`[1,1,1]`
`matches 1 [2,3,4,6]` $\rightsquigarrow$`[]`
Next, use it to implement the function `isElementOf n xs` which returns `True` if `n` occurs in the list `xs`, and `False` otherwise.