Exercises
Tail Recursive Functions

Exercise 1

In the previous exercise set, you wrote the code for:

```
fibonacci :: Int -> Int
```

that calculates the Fibonacci number as per the following definition (note for non-negative integers)
$F_0 = 0, F_1 = 1$
$F_n = F_{n-1} + F_{n-2}$ Now, write this fibonacci using tail recursion.

```
fibT :: Int -> Int
```

Exercise 2

Noting our definition of add from the previous lecture (this is actually tail recursive)

```
myAdd :: Int -> Int -> Int
myAdd x 0 = x
myAdd 0 y = y
myAdd x y = myAdd (x-1) (y+1)
```

write a function

```
myMult :: Int -> Int -> Int
```

which takes two positive integers and returns the product of the two numbers. This calcluation can only use + and −. Ensure that the solution is tail recursive

Exercise 3

Given the recursive function

```
reverse_ :: [a] -> [a]
reverse_ [] = []
reverse_ (x:xs) = reverse_ xs ++ [x]
```

that returns the reverse of a list, but this time, using tail recursion.