Exercise 4.2

#1

Compute the sum of the numbers from 1000 down to 1. Do this by defining a function sum n that computes the sum n + (n − 1) + · · · + 2 + 1. (Use straightforward summation, no clever tricks).

Solution:

```
let sum n = if n=1 then 1 else n + (sum (n-1))
in sum 1000 end
```

#2

Compute the number 3^8 , that is, 3 raised to the power 8. Again, use a recursive function.

Solution:

```
let threeToPow n = if n=0 then 1 else 3*(threeToPow (n-1))
in threeToPow 8 end
```

#3

Compute $3^0 + 3^1 + \cdots + 3^{10} + 3^{11}$, using a recursive function (or two, if you prefer).

Solution using function from #2:

```
let threeToPow n = if n=0 then 1 else 3*(threeToPow (n-1)) in
let sumPowerOfThrees n = if n=0 then 1 else (threeToPow n)+(sumPowerOfThrees (n-1))
in sumPowerOfThrees 11 end
```

This should give the correct result, but is computing it in the reverse order of the example.

#4

Compute $1^8 + 2^8 + \cdots + 10^8$, again using a recursive function (or two).

Solution:

```
let sumPowerOfEight n = if n=1 then 1 else n*n*n*n*n*n*n*n + sumPowerOfEight (n-1)
in sumPowerOfEight 10 end
```

I'ts a bit hacky, but it works.