

# Assignment 1

Deadline for solutions: 18.06.2022

---

## Exercise 1 Running Summs

(13 Points)

Recall the definition of running sums from the lecture:

```
open import bool
open import nat
open import nat-thms
open import eq

runningSum : (N → N) → N → N
runningSum σ 0 = σ 0
runningSum σ (suc n) = (runningSum σ n) + σ (suc n)
```

Prove the following property in Agda.

```
sum-nat-cubes : ∀ (n : N) → runningSum cube n ≡ square (runningSum id n)
```

where

```
cube : N → N
cube x = x * x * x
```

```
id : N → N
id n = n
```

**Hint:** Consider proving the following lemmas first

- `sq-binomial` :  $\forall (x\ y : \mathbb{N}) \rightarrow \text{square } (x + y) \equiv \text{square } x + 2 * x * y + \text{square } y$
- `2sum-nats'` :  $\forall (n : \mathbb{N}) \rightarrow 2 * (\text{runningSum id } n) \equiv n * n + n$

## Exercise 2 Fibonacci Numbers

(8 Points)

Using the following definition of *Fibonacci numbers*:

```
fib : N → N
fib 0 = 0
fib 1 = 1
fib (suc (suc n)) = fib (suc n) + fib n
```

prove the following property in Agda:

```
fib-squares : ∀ (n : N) → runningSum (λ n → square (fib n)) n ≡ (fib n) * fib (suc n)
```

### Exercise 3 Binary Number Presentation

(9 Points)

Agda represents natural numbers in the *unary numeral system*, that is the size of the data to store a number is proportional to that number. A more efficient representation uses a binary rather than a unary system. We represent a number as a *bitstring*:

```
data Bin : Set where
  ⟨⟩ : Bin
  _0 : Bin → Bin
  _1 : Bin → Bin
```

For instance, the bitstring 1011 standing for the number eleven is encoded as  $\langle \rangle$  1 0 1 1. Representations are not unique due to leading zeros. Hence, eleven is also represented by  $\langle \rangle$  0 1 0 1 1.

(a) Define a function

```
inc : Bin → Bin
```

that converts a bitstring to the bitstring for the next higher number. For example, since 1011 encodes twelve, we should have:  $\text{inc } (\langle \rangle \text{ 1 0 1 1}) \equiv \langle \rangle \text{ 1 1 0 0}$ . Confirm that this gives the correct answer for the bitstrings encoding zero through four.

(b) Using the above, define a pair of functions to convert between the two representations.

```
to   : ℕ → Bin
from : Bin → ℕ
```

For the former, choose the bitstring to have no leading zeros if it represents a positive natural, and represent zero by  $\langle \rangle$  0.

(c) Consider the following laws:

```
bin-stmt1 : ∀ (b : Bin) → from (inc b) ≡ suc (from b)
bin-stmt2 : ∀ (b : Bin) → to (from b) ≡ b
bin-stmt3 : ∀ (n : ℕ) → from (to n) ≡ n
```

For each law: if it holds, prove; if not, give a counterexample.