# COMP3400
# Assignment 2 Written

Paul Vrbik

April 1, 2023

In addition to this written work there are *four* coding questions. The written work is worth 35 points and the coding questions are worth 60 points totalling 95 points.

## Tail Recursion

The *mean* of a collection of observations $x_1, x_2, \ldots, x_n$ is given by

$$\bar{x}_n = \frac{1}{n} \sum_{k=1}^{n} x_k.$$

n* mean(x)n = sumFromk(1,n)xk

At first glance it does not seem that $\bar{x}_{n+1}$ is related to $\bar{x}_n$ but notice

$$\bar{x}_{n+1} = \underline{\frac{1}{n+1} \sum_{k=1}^{n+1} x_k} = \frac{1}{n+1} \left( x_{n+1} + \boxed{\sum_{k=1}^{n} x_k} \right) = \frac{1}{n+1} \left( x_{n+1} + n\,\bar{x}_n \right)$$

and thereby

$$\bar{x}_{n+1} = \frac{x_{n+1} + n\,\bar{x}_n}{n+1}.$$

We have shown how to compute $\bar{x}$ *iteratively*. In particular we compute $\bar{x}_{n+1}$ from *only* $(n, \bar{x}_n, x_{n+1})$.

**Question 1.** [2 MARKS]

The definition of *variance* (specifically sample variance) of a collection of observations $x_1, x_2, \ldots, x_n$ is

$$S_n^2 = \frac{1}{n} \sum_{k=1}^{n} (x_k - \bar{x}_n)^2. \tag{1}$$

Take for granted (though it is easy to show) that

$$S_n^2 = \left( \frac{1}{n} \sum_{k=1}^{n} x_k^2 \right) - \bar{x}_n^2 \tag{2}$$

and use (1) and (2) to derive an iterative definition for variance. That is, produce an equation which computes $S_{n+1}^2$ from *only* $(n, S_n^2, \bar{x}_n, \bar{x}_{n+1}, x_{n+1})$.

**Question 2.** [2 MARKS]

Define a *linear recursive*

```
lcVariance :: [Float] -> Float
```

that computes the variance $S_n^2$ of a list. Your function must use `lcVariance xs` to compute `lcVariance (x:xs)`. Note, `fromIntegral.length` is compatible with `Float`.

**Question 3.** [4 MARKS]

Define a *tail recursive* helper function with type:

```
trVariance :: Float -> Float -> Float -> [Float] -> Float
```

that finds the variance of the list.

Remember your function may *only* be equal to

1. a call to itself with different inputs, or
2. one of the inputs.

**Question 4.** [1 MARK]

Define

```
variance :: [Float] -> Float
```

via a single call to `trVariance`.

**Question 5.** [1 MARK]

Define an iteration invariant for `trVariance` that proves the correctness of `variance`.

**Question 6.** [5 MARKS]

Prove `trVariance` satisfies your iteration invariant.

**Question 7.** [1 MARK]

State the bound value for `trVariance`.

**Question 8.** [2 MARKS]

Prove your bound value is always non-negative and decreasing.

**Question 9.** [2 MARKS]

Define *two* distinct quick-checks for `variance` that *both* use lists from `Arbitrary [Float]`. In particular, your quick-checks should be for lists of *arbitrary length*.

# Induction

Consider the following definitions for implementing addition on natural numbers.

```
data Nat = Zero | Succ Nat deriving Show

plus :: Nat -> Nat -> Nat
plus m Zero = m
plus m (Succ(n)) = Succ (plus m n)
```

**Question 10.** [7 MARKS]

*Using induction prove*:

`plus Zero n = n`

for any `n :: Nat`.

**Question 11.** [8 MARKS]

*Using induction prove* that plus is commutative, that is,

`plus m n = plus n m`

for any `n,m :: Nat`.

*Hint:* Keep `m` fixed and do induction over `n`.

*Note:* There are style marks for this question so be sure to justify each of your lines and include all necessary components of induction.