# COMP 3031 Assignment 1
# SML programming
# Fall 2020

**Due: 5PM on Oct 5, 2020**

## Instructions

- There are **five** questions in this assignment. Each question counts for 2 points. The total number of points is 10.
- This is an individual assignment. You can discuss with others and search online resources, but your submission should be your own code.
- Write your functions exactly the same as defined in the problem description (name, type, and functionality). In addition, you can write any helper functions as needed and call any built-in SML functions available in the lab machine.
- Put your entire solution in a single text file called "*ass1.ml*". In this file, put down your **name, ITSC account, and student ID** as a ***comment*** (surrounded by "(*" and "*)") on the first line.
- Submit your file through Canvas before the deadline.
- Your submission will be tested under the SML interpreter on a lab machine by issuing the following command:

*- use "ass1.ml";*

- **No** late submissions will be accepted under usual circumstances.

**NOTE:** We will perform code similarity checks. In case a submission has confirmed code similarity issues, we will deduct partial marks or full marks on a case-by-case basis.

**PART I:**

## Question 1. Sum of digits

Write a function `sumDigits` that takes an integer `n` as input and returns the sum of all digits of the integer n.

```
- sumDigits;
val sumDigits = fn : int -> int
```

Examples:

```
- sumDigits(1000);
val it = 1 : int

- sumDigits(54321);
val it = 15 : int
```

## Question 2. Prefix sum of number of occurrences

The prefix sum of an array `array` is an array `prefix_sum` such that the $i^{th}$ element of `prefix_sum` is the sum of the first through the $i^{th}$ element of `array`: `prefix_sum[i-1] = array[0] + array[1] + … + array[i-1].`

Write a function `frequencyPrefixSum` that takes an integer list `lst` and an integer `n` as input and returns the prefix sum of number of times of `n` occurring in `lst`.

```
- frequencyPrefixSum;
val frequencyPrefixSum = fn : int list * int -> int list
```

Examples:

```
- frequencyPrefixSum ([1, 2, 2, 4, 5], 2);
val it = [0,1,2,2,2] : int list

- frequencyPrefixSum ([1, 2, 2, 4, 5], 3);
val it = [0,0,0,0,0] : int list

- frequencyPrefixSum ([], 2);
val it = [] : int list
```

**PART II:**

We define the following nested list data type to be used throughout PART II:

```
datatype 'a llist = LList of 'a llist list| Elem of 'a;
```

A nested list consists of an element of a polymorphic type, or a list of nested lists. The following are some examples:

```
Elem(1);
LList [];
LList([Elem(1), LList([Elem(2), LList([Elem 1, Elem(3)]), Elem(4)])]);
```

## Question 3. Flatten a nested list.

Write a function `flatten` that takes a nested list as input and returns a flat list of all elements in the nested list. Note that the elements in the resulting list are in the same order as in the nested list.

```
- flatten;
val flatten = fn : 'a llist -> 'a list
```

Examples:

```
- flatten(Elem(3));
val it = [3] : int list

- flatten(LList([]));
val it = [] : ?.X1 list

- flatten(LList([Elem(1),LList([Elem(2),LList([]),Elem(3)]),Elem(4)]
));
val it = [1,2,3,4] : int list
```

## Question 4. Depth of a nested list.

Write a function `depth` that takes a nested list as input and returns the highest level of nesting in the list.

```
- depth;
val it = fn : 'a llist -> int
```

For example:

```
- depth(Elem(1));
val it = 0 : int

- depth(LList([]));
val it = 1 : int

-  depth(LList([Elem(1),LList([Elem(2),LList([]),Elem(3)]),Elem(4)]));
val it = 3 : int
```

## Question 5. Equality of two nested lists.

Write a function `equal` that takes two nested lists as input and returns true if they are equal on all corresponding pairs of elements.

```
- equal;

val it = fn : ''a llist * ''a llist -> bool
```

For example:

```
- val test_1 = Elem(3);
- val test_2 = Elem(9);
- equal(test_1, test_2);
val it = false : bool

- val test_1 = Elem(3 + 6);
- val test_2 = Elem(9);
- equal(test_1, test_2);
val it = true : bool

- val test_3 = LList([Elem(#"1")]);
- val test_4 = LList([Elem(#"1")]);
- equal(test_3, test_4);
val it = true : bool

- val test_5 = LList([Elem(1), LList([Elem(2), LList([Elem(5),
Elem(6)]), Elem(3)]), Elem(4)]);
- val test_6 = LList([Elem(1), LList([Elem(2), LList([Elem(5),
LList([Elem(6)])]), Elem(3)]), Elem(4)]);
- equal(test_5, test_6);
val it = false : bool
```