# COMP 302 Winter 2019
## Mid-term Examination: Solutions

**Question 1**[30 points]
In this question you have to write a *higher-order function* that takes a function $f$ and a non-negative integer $n$ as arguments and returns the function that applies $f$, $n$ times. Here are some examples

```
# let rec repeated (f,n) = .... <code removed>...
      val repeated : ('a -> 'a) * int -> 'a -> 'a = <fun>
# let inc n = n + 1;;
val inc : int -> int = <fun>
# repeated(inc, 5) 0;;
- : int = 5
# let addfoo s = "foo" ^ s;;
val addfoo : string -> string = <fun>
# repeated(addfoo, 3) "bar";;
- : string = "foofoofoobar"
```

The way the expressions are nested is *crucial* because it shows whether you have understood the types. We will ruthlessly cut marks for improper nesting of expressions.

Solution

```
let rec repeated (f,n) =
  if (n = 0) then fun x -> x
  else
    fun x -> f ((repeated (f,n-1)) x);;

# let inc n = n + 1;;
val inc : int -> int = <fun>
# repeated(inc, 5) 0;;
- : int = 5
# let addfoo s = "foo" ^ s;;
val addfoo : string -> string = <fun>
# repeated(addfoo, 3) "bar";;
- : string = "foofoofoobar"
```

**Question 2**[40 points]
A matrix can be represented as a list of lists. For example we can write

```
# let m1 = [[1;2;3];[4;5;6];[7;8;9]];;
val m1 : int list list = [[1; 2; 3]; [4; 5; 6]; [7; 8; 9]]
```

This represents the $3 \times 3$ matrix $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$. We say that a matrix is *proper* if every row has the same length and *square* if it is proper *and* the length of each row is equal to the number of rows. The example above is a (proper) square matrix. Write a program to test whether a given list of lists is a square matrix.

Solution to Q2:

```
let square m =
  match m with
  | [] -> true
  | _ -> List.for_all (fun r -> (List.length m) = List.length r) m;;
```

Of course, there are many ways of doing this.

**Question 3**[30 points]
Consider the following nested let expression. Draw the environment diagram when the body of the function $f$ is about to be evaluated; this is *after* $f$ has been applied to $x$ and the evaluation of $u + x + y$ is about to start. Are all bindings used?

```
let x = 1 in
    let y = x in
      let x = 2 in
        let f u = u + x + y in
          let x = 4 in
            f x
```

Yes all binding are used. See me if you want to see the picture. The most common mistake was leaving out the frame with $u$ bound to 4.