COMP 3958: Lab 1

Submit a zip file named `lab1.zip` containing your 2 sources files: & `expo.ml` Put your implementation in a file named `lab1.ml`. Your files must compile without warnings or errors. If it does not, it is possible that you may receive no credit for this lab. Maximum score: 15.

1. Implement each of the following functions using recursion without calling external functions except those in the `Stdlib` module and the `List.rev` function. For each function, provide both tail-recursive and non-tail-recursive implementations. The name of the tail-recursive version should end in `_tr`, e.g., the two "zip" functions should be named `zip` and `zip_tr`. You may implement additional helper functions if necessary. Provide at least 3 tests for each implementation. Also provide a `run_all_tests` function that runs all the tests when invoked.

   (a) `val zip :  'a list -> 'b list -> ('a * 'b) list`

   `zip lst1 lst2` is the list formed by pairing corresponding elements of `lst1` and `lst2` into a tuple. If `lst1` and `lst2` are of different lengths, pairing stops when the shorter list ends. For example,

   `zip [1; 2; 3] ['a'; 'b'; 'c'; 'd']` returns `[(1,'a'); (2,'b'); (3,'c')]`.

   (b) `val unzip :  ('a * 'b) list -> 'a list * 'b list`

   `unzip lst`, where `lst` is a list of pairs, is a pair of lists where the first list consists of the first component of each pair in `lst` and the second list consists of the second component of each pair in `lst`. For example,

   `unzip [(1,'a'); (2,'b'); (3,'c')]` returns `([1; 2; 3], ['a'; 'b'; 'c'])`

   (c) `val dedup:  'a list -> 'a list`

   `dedup lst` is the list formed from `lst` by collapsing consecutive duplicated elements into a single element. For example,

   `dedup [1; 1; 2; 3; 3; 3; 2; 1; 1]` returns `[1; 2; 3; 2; 1]`

2. The exponential function can be defined by the infinite series

$$\exp(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$$

   Implement from basics a function `expo` with signature

   ```
   val expo :  int -> float -> float
   ```

   so that `expo n x` returns the sum of the first `n` terms of the above series, i.e., it returns the sum

$$1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^{n-1}}{(n-1)!}$$

   Note that the function requires $n \geq 1$. (We call our function `expo` because there is an `exp` in `Stdlib`.)

   You are only allowed to use the arithmetic operator +, -, *, /, +., -., *., /., and `float_of_int`. Try to implement `expo` in such a way that you do not need to calculate the factorial from scratch for every term.

   Check by evaluating `expo 20 1.0` to find an approximate value of e which is defined as $e = \exp(1)$ and compare it to `exp 1.0`.