COMP 3958: Lab 1

Put your implementation in a file named `lab1.ml`. Be sure to test your functions in `utop`. Note that you may need to implement helper functions. Provide comments to indicate what each function does. Your file must compile. If it does not, you may receive no credit for this lab exercise. Maximum score: 13.

1. Implement the following functions with the given signatures using recursion and without calling any function from the library. Provide tail-recursive versions if possible. You may implement additional helper functions if needed.

   (a) `val drop :  int -> 'a list -> 'a list`

   `drop n lst` returns `lst` with the first `n` elements dropped. For example,

   `drop 3 [4;2;6;7;6;8;1]` returns `[7;6;8;1]`
   `drop (-1) [3; 2; 7]` returns `[3;2;7]`
   `drop 4 [3;2;7]` returns `[]`

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

   `zip lst1 lst2` returns a list consisting of pairs of corresponding elements of `lst1` and `lst2`. If `lst1` and `lst2` are of different lengths, the function stops "zipping" when the shorter list ends. For example,

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

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

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

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

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

   `dedup lst` returns a list where all consecutive duplicated elements in `lst` are collapsed into a single element. (Assume that the type of elements supports comparison using the = operator.) 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 `exp` with signature

   `val exp :  int -> float -> float`

   so that `exp 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 precondtion for the function is that $n \geq 0$.

   Evaluate `exp 20 1.0` to find an approximate value of e which is defined as $e = \exp(1)$.

   Do not use any library function except for `float_of_int`. If possible, try to implement `exp` in a way that you do not need to calculate the factorial from scratch for every term.