HW 01

CS 321 – Özyeğin University

Due None. This is for exercise.

Note: There may be multiple ways to solve a problem. Your function definitions may not need to be exactly the same as the solution. However, the types of the functions should *exactly* match the types of the functions in the solution. You are strongly encouraged to write several test cases to check the correctness of your functions.

The questions in the assignments are designed so that there are easy, medium level, and hard questions. It is normal and in fact expected that not everybody solves all the problems. If you can't find the solution to a problem, don't worry; that may happen to anybody. Study harder and in the next assignment you'll do better.

Make sure that your file compiles without any errors. The types of the functions should *exactly* match the types of the functions in the solution. Submission with compilation errors will receive 0 points. There is an old saying: *"Derdinizi önce bilgisayara, sonra bana anlatın"*. Poorly written code (e.g. bad indentation) will be penalized as well.

Getting started: Check out the exercise files from the SVN repository. To do this, run the following command in your terminal, but replace each occurrence of `donald.duck` with your username.

```
svn co --username donald.duck https://srl.ozyegin.edu.tr/cs321students/donald.duck/hw1
```

Now, change your directory to `hw1` and there you have all the files you need.

Submission: To submit, run the following command while your are inside the `hw1` directory:

```
make commit
```

You may submit as many times as you want.

Problems

In all problems below, you must use explicit recursion (i.e. no library functions such as `map`, `fold`, and `foldBack`).

1. (5 pts) Write a function `stringy : string list -> (string * int) list` that associates each string in its input with the length of the string. You may use `String.length` to find the length of a string.

```
> stringy ["a"; "bbb"; "cc"; "ddddd"];;
val it : (string * int) list = [("a", 1); ("bbb", 3); ("cc", 2); ("ddddd", 5)]
```

2. (5 pts) Write a function `positivesOf : int list -> int list` that returns the positive numbers in its input.

```
> positivesOf [-4; 9; 2; -8; -3; 1; 0];;
val it : int list = [9; 2; 1]
```

3. (5 pts) Write a function `gotcha : ('a -> bool) -> 'a list -> 'a` that takes a predicate function `p` and a list `lst`, and returns the first element `x` of `lst` for which `p(x)` is true. If there is no such element, the function should fail with the error message "No soup for you!".

```
> gotcha (fun n -> n > 5) [3; 4; 1; 2; 8; 4; 9; -8];;
val it : int = 8
> gotcha (fun n -> n > 15) [3; 4; 1; 2; 8; 4; 9; -8];;
System.Exception: No soup for you!
...
```

To make the program fail in the error case, use the (`failwith "No soup for you!"`) expression.

4. (5 pts) Write a function `allUntil : ('a -> bool) -> 'a list -> 'a list` that takes a predicate function `p`, a list `lst`, and returns all the elements of `lst` up to the first element that does not satisfy `p`.

```
> allUntil (fun n -> n < 5) [3; 4; 1; 2; 8; 4; 9; -8];;
val it : int list = [3; 4; 1; 2]
> allUntil (fun n -> n > 5) [3; 4; 1; 2; 8; 4; 9; -8];;
val it : int list = []
> allUntil (fun n -> n < 15) [3; 4; 1; 2; 8; 4; 9; -8];;
val it : int list = [3; 4; 1; 2; 8; 4; 9; -8]
> allUntil (fun s -> String.length(s) < 4) ["aa"; "bbb"; "c"; "dddd"; "eeeeeeee"; "fff"]
val it : string list = ["aa"; "bbb"; "c"]
```

5. (10 pts) Write a function `interleave : 'a list -> 'a list -> 'a list * 'a list` that mixes its inputs by interleaving their elements. In this question, you may assume that the inputs will always have the same length; that is, I won't test your function with naughty inputs.

```
> interleave [1;2;3;4;5] [6;7;8;9;10];;
val it : int list * int list = ([6; 2; 8; 4; 10], [1; 7; 3; 9; 5])
> interleave [2;3;4;5] [7;8;9;10];;
val it : int list * int list = ([7; 3; 9; 5], [2; 8; 4; 10])
```

6. (10 pts) Write a function `enumerate : 'a list -> ('a * int) list` that enumerates the elements of its input with their index. The first element in a list is considered to be at index 0. You will want to write a helper function for this problem.

```
> enumerate ['a';'b';'c';'d';'e'];;
val it : (char * int) list = [('a',0);('b',1);('c',2);('d',3);('e',4)]
```

In all problems below, you must NOT use explicit recursion;use the library functions `map`, `fold`, and `foldBack`.

7. (5 pts) Write a function `stringyWithMap` that is exactly the same as `stringy`, but this time use `map`.

8. (5 pts) Write a function `stringyWithFoldBack` that is exactly the same as `stringy`, but this time use `foldBack`.

9. (5 pts) Write a function `stringyWithFold` that is exactly the same as `stringy`, but this time use `fold`.

10. (5 pts) Write a function `positivesOfWithFoldBack` that is exactly the same as `positivesOf`, but this time use `foldBack`.

11. (5 pts) Write a function `positivesOfWithFold` that is exactly the same as `positivesOf`, but this time use `fold`.

12. (5 pts) Write a function `enumerateWithFold` that is exactly the same as `enumerate`, but this time use `fold`.