Exercise 3

Last update January 27, 2025

This exercise sheet must be handed in via LearnIt

Your name must be part of the filename, e.g., `FP-03-<name>.fsx`. An example: `FP-03-MadsAndersen.fsx`.

You can only upload one file and it must be of type `fs` or `fsx`.

It is important that you annotate your own code with comments. It is also important that you apply a functional style, i.e., no loops and no mutable variables.

For this hand-in you also need to consider scenarios where your solutions should return an error, i.e., an exception. The requirement is, that no matter what input you pass to your function that fulfils the function type, then the function should return the intended answer or an exception. It is up to you to define the exceptions and whether they should carry extra information, like error messages.

Exercise 3.1 Write a function
`downTo:int->int list`
so that `downTo n` returns the `n`-element list `[n; n-1; ...; 1]`. You must use if-then-else expressions to define the function.

Secondly define the function `downTo2` having same semantics as `downTo`. This time you must use pattern matching.

Exercise 3.2 Write a function
`removeOddIdx:int list->int list`
so that `removeOddIdx xs` removes the odd-indexed elements from the list `xs`:

```
removeOddIdx [x0; x1; x2; x3; x4; ...] = [x0; x2; x4; ...]
removeOddIdx [] = []
removeOddIdx [x0] = [x0]
```

Exercise 3.3 Write a function
`combinePair:int list->(int*int) list`
so that `combinePair xs` returns the list with elements from `xs` combined into pairs. If `xs` contains an odd number of elements, then the last element is thrown away:

```
combinePair [x1; x2; x3; x4] = [(x1,x2);(x3,x4)]
combinePair [x1; x2; x3] = [(x1,x2)]
combinePair [] = []
combinePair [x1] = []
```

Hint: Try use pattern matching.

Exercise 3.4 Solve HR, exercise 3.2.

Exercise 3.5 Solve HR, exercise 3.3.

Exercise 3.6 Solve HR, exercise 4.4