Contents

F# Scripts/Files

You can only evaluate expressions that will fit on a single line at the prompt. If you want to write more complex expressions, or declare your own functions, you need to put the code in a file and then load that file into the F# Interactive. **Script files** can be run from a command line using **F#** Interactive (FSI).

1.2.1 –Create F# Script

Open VS code and create a .fsx script file, PclExercise1.fsx. You can use any text editor you prefer, but Visual Studio Code (Dev Container) is recommended. Now declare some values in the file, for example:

```
let x = 23
let myName = "Your Name"
let age = 25
let country = "Denmark"
let y = 6 + 6
```

(Note that you don't have to insert ";;" after each declaration in the file: this is only necessary in the interactive environment, to signal that a line of input is ready to interpret.)

Enter the following declarations into the file and evaluate. What values will **b** and **c** hold, and why?

```
let a = 5
let b = let a = 10 in a + 5
let c = a + b
```

1.2.2 – Function Declaration I

Declare the following functions inside the script file. Select and Alt + Enter to test it as you work on it:

- Define a function `addNum1` that adds 1 to a number.
- Define a function `addNum10`, which takes an integer argument, adds 10 to it, and returns the result.
- Define a function `addNum20`, which uses `addNum10` to add 20 to a given integer.

1.2.3 – Function Declaration II

Declare the following functions:

- Define the function max2 that takes two integers as arguments and returns the largest of them.
- Define a function evenOrOdd that takes an integer and prints out "even number" if the given integer is even otherwise it prints out "odd number".
- Define a function addXY that takes two integers and prints out the two integers before adding them.

1.2.4 – Function Declaration - Extra

Declare the following functions:

- Define a function `addNum_j k` that takes two integers `j`, `k` as arguments and returns `j + 10*k`. For instance, `addNum_jk 3 5 = 3 + 10*5 = 53`. You are, however, not allowed to use addition and multiplication directly: instead, you must write a recursive solution that calls `addNum10` defined in Exercise 1.2.3 above. (So `addNum_jk 3 5` should be computed as `3 + 10 + 10 + 10 + 10 + 10`.)