Programming Assignment 2

Programming Languages
(SWE3006-41)
Spring, 2022

Instructor:

Sungjae Hwang

- jason.sungjae.hwang@gmail.com
- https://softsec-lab.github.io/

Introduction

- Deadline : 2022.05.11

- You have two days for late submission (~2022.05.13)
  - 25% deduction per day

- Write functions using OCaml!

- Submit source code (*.ml) for each exercise
  - You will not get any points if your source code does not compiles well.
  - Submit **"PA2_StudentID.zip"** through icampus
  - The zip file should contains:
    - ex1.ml, ex2.ml, ex3.ml, ex4.ml, ex5.ml, ex6.ml, ex7.ml

- Please leave the questions in the google sheet
  - https://docs.google.com/spreadsheets/d/1ob4m9kC8hhDPCYrWcG6M46HzZthrSscP7CjZnTrneAs/edit

Installing OCaml

- 참고: https://ocaml.org/docs/install.html#Ubuntu-Ubuntu-20-04
  - Hello World Example (Linux)

```
root@b06966b74d68:/# apt install ocaml
```

Installing Ocaml

```
print_string "Hello World!\n";
```

hello.ml file

```
root@b06966b74d68:/# ocamlc hello.ml
root@b06966b74d68:/# ./a.out
Hello World!
root@b06966b74d68:/#
```

Compiling and running

Exercise #1 (5pt)

- Write below function
  - **gcd : int -> int -> int**
  - The function returns the greatest common divisor (GCD) of two given non-negative integers.
  - Use the Euclidean algorithm based on the following principle (for two integers n an m such that n >= m):

$$\texttt{gcd}\ n\ m = \begin{cases} n & (m = 0) \\ \texttt{gcd}\ (n - m)\ m & \end{cases}$$

- Test cases
  - gcd 10 0 => 10
  - gcd 9 5 => 1
  - gcd 13 13 => 13
  - gcd 37 600 => 1
  - gcd 0 0 => 0

Exercise #2 (5pt)

- Write below function
  - **prime : int -> bool**
  - The function checks whether a number is prime.
  - N is prime if and only if n is its own smallest divisor except for 1.

- Test cases
  - prime 2 => true
  - prime 3 => true
  - prime 4 => false
  - prime 17 => true

Exercise #3 (10pt)

- Write below function
  - **sec_last : int list -> int**
  - The function returns the second last element of a list.

- Test cases
  - sec_last [1;2;3;4;5] => 4
  - sec_last [4;3;2;1] => 2
  - sec_last [] => 0
  - sec_last [1] => 0
  - sec_last [1,2] => 1

Exercise #4 (20pt)

- Write below function
  - **merge : int list -> int list -> int list**
  - The function takes two integer lists sorted in descending order.
  - The function returns a new sorted integer list that includes every element in the two given list.

- Test cases
  - merge [3;2;1] [5;4] => [5;4;3;2;1]
  - merge [5;3] [5;2] => [5;5;3;2]
  - merge [4;2] [] => [4;2]
  - merge [] [2;1] => [2;1]
  - merge [] [] => []
  - merge [0;0;0;0] [0;0;0;0;] => [0;0;0;0;0;0;0;0;]
  - merge [4;3;-2] [9;7;7] => [9;7;7;4;3;-2]
  - merge [-2; -999] [] => [-2, -999]

Exercise #5 (20pt)

- Write below function
  - **range : int -> int -> int list**
  - The function is invoked as "**range lower upper**".
  - It returns a sorted list of integers in the range [**lower** … **upper**].
  - If **lower** is greater than **upper**, then function returns the empty list.

- Test cases
  - range 10 15 => [10;11;12;13;14;15]
  - range (-2) 7 => [-2;-1;0;1;2;3;4;5;6;7]
  - range 9 3 => []
  - range 22 22 => [22]
  - range 55 (-12) => []

Exercise #6 (20pt)

- Write below function
  - **sigma : int * int * (int -> int) -> int**
  - Such that sigma(a,b,f) returns as follow:

$$\Sigma_{n=a}^{b} f(n)$$

- Test cases
  - sigma (10,10, (fun x -> x)) => 10
  - sigma (11,10,(fun x -> x)) => 0
  - sigma (10,5,(fun x -> x)) => 0
  - sigma (1,10,(fun x -> if x mod 2 = 0 then 1 else 0)) => 5
  - sigma (2,10,(fun x -> x + 10)) => 144
  - sigma (0,100,(fun x -> 0)) => 0
  - sigma (10,12,(fun x -> 2 * x)) => 66

Exercise #7 (20pt)

- The **fold** function for lists:
  - fold: (‘a -> ‘b -> ‘a) -> ‘a -> ‘b list -> ‘a
  - recombines then results of recursively processing its constituent parts, building up a return value through use of a given combining operation. For example,
  - fold f a [b1; …; bn] = f (…(f (f a b1) b2) …) bn.
  - Extend fold function so that it takes three lists. **Write below function:**
  - **fold3: (‘a -> ‘b -> ‘c -> ‘d -> ‘a) -> ‘a -> ‘b list -> ‘c list -> ‘d list -> ‘a**
  - of which means,
  - fold3 f a [b1;…;bn] [c1;…;cn] [d1;…;dn] = f (...(f (f a b1 c1 d1) b2 c2 d2)...) bn cn dn.
  - You may assume that all the given lists are of the same length.

- Test cases
  - fold3 (fun a b c d -> a + b + c + d) 10 [33;67;12;33] [10;23;84;57] [11;55;23;58] => 476
  - fold3 (fun a b c d -> (-a) + b + c + d) 4 [11;63;-45;22] [75;123;-44;1] [55;24;20;3] => 168
  - fold3 (fun a b c d -> a * b * c * d) 55 [] [] [] => 55
  - fold3 (fun a b c d -> (a * b * c + d) mod 7) 33 [12;33] [10;7] [5;12] => 5
  - fold3 (fun a b c d -> if b then a + c else a + d) 34 [true;false;false;true] [12;3;4;77] [11;23;6;100] => 152
  - fold3 (fun a b c d -> if b then a else c + d) 55 [true;true;false;false;true] [111;63;88;123;98] [0;23;778;34;6] => 157