Homework 1
ENE4014 Programming Languages, Spring 2021
Woosuk Lee
due: 4/12(Mon), 24:00

Exercise 1 Write a function

$$\texttt{gcd} : \texttt{int} \to \texttt{int} \to \texttt{int}$$

that 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 $n \geq m$):

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

□

Exercise 2 Write a function

$$\texttt{merge} : \texttt{int list} \to \texttt{int list} \to \texttt{int list}$$

that takes two integer lists sorted in descending order and returns a new sorted integer list that includes every element in the two given lists. For example,

$$\begin{aligned} \texttt{merge}\ [3;2;1]\ [5;4] &= [5;4;3;2;1] \\ \texttt{merge}\ [5;3]\ [5;2] &= [5;5;3;2] \\ \texttt{merge}\ [4;2]\ [] &= [4;2] \\ \texttt{merge}\ []\ [2;1] &= [2;1] \end{aligned}$$

□

Exercise 3 Write a function

$$\texttt{range} : \texttt{int} \to \texttt{int} \to \texttt{int list}$$

`range lower upper` generates a sorted list of integers in the range [`lower` ... `upper`]. If `lower` is greater than `upper`, the function returns the empty list. For example,

$$\begin{array}{rcl}
\texttt{range}\ 1\ 3 & = & [1;2;3] \\
\texttt{range}\ (-2)\ 2 & = & [-2;-1;0;1;2] \\
\texttt{range}\ 2\ 2 & = & [2] \\
\texttt{range}\ 2\ (-2) & = & []
\end{array}$$

$\square$

```
type formula = TRUE | FALSE
  | NOT of formula
  | ANDALSO of formula * formula
  | ORELSE of formula * formula
  | IMPLY of formula * formula
  | LESS of expr * expr
and expr = NUM of int
  | PLUS of expr * expr
  | MINUS of expr * expr
```

Exercise 4 Considering the above definition of propositional formula, write a function

$$\texttt{eval} : \texttt{formula} \rightarrow \texttt{bool}$$

that computes the truth value of a given formula. For example,

```
eval (IMPLY (IMPLY (TRUE, FALSE), TRUE))
```

evaluates to *true*, and

```
eval (LESS (NUM 5, PLUS (NUM 1, NUM 2)))
```

evaluates to *false*. $\square$

Exercise 5 Binary trees can be inductively defined as follows:

```
type btree = Empty | Node of int * btree * btree
```

For example, the following `t1` and `t2` are binary trees.

```
let t1 = Node (1, Empty, Empty)
let t2 = Node (1, Node (2, Empty, Empty), Node (3, Empty, Empty))
let t3 = Node (1, Node (2, Node (3, Empty, Empty), Empty), Empty)
```

Write a function

$$\texttt{height} : \texttt{btree} \rightarrow \texttt{int}$$

that computes the height of a given binary tree. The height of a given binary tree is inductively defined as follows:

$$\begin{array}{rcl}
\texttt{height Empty} & = & 0 \\
\texttt{height}\ (\texttt{Node}(\texttt{n},\texttt{l},\texttt{r})) & = & \begin{cases} (\texttt{height l}) + 1 & (\texttt{height l} > \texttt{height r}) \\ (\texttt{height r}) + 1 & (\text{otherwise}) \end{cases}
\end{array}$$

For example,

$$\begin{aligned} \texttt{height Empty} &= \texttt{0} \\ \texttt{height t1} &= \texttt{1} \\ \texttt{height t2} &= \texttt{2} \\ \texttt{height t3} &= \texttt{3} \end{aligned}$$

□

Exercise 6 Write a function

$$\texttt{balanced} : \texttt{btree} \to \texttt{bool}.$$

that determines if a given binary tree is *balanced*. A tree is balanced if the tree is empty or the following conditions are met.

- The left and right subtrees' heights differ by at most one, and
- The left subtree is balanced, and
- The right subtree is balanced.

For example,

$$\begin{aligned} \texttt{balanced Empty} &= \texttt{true} \\ \texttt{balanced t1} &= \texttt{true} \\ \texttt{balanced t2} &= \texttt{true} \\ \texttt{balanced t3} &= \texttt{false} \end{aligned}$$

where `t1`, `t2`, and `t3` are defined in Exercise 5. □

Exercise 7 The `fold` function for lists

$$\texttt{fold} : (\texttt{`a} \to \texttt{`b} \to \texttt{`a}) \to \texttt{`a} \to \texttt{`b list} \to \texttt{`a}$$

recombines the results of recursively processing its constituent parts, building up a return value through use of a given combining operation. For example,

$$\texttt{fold f a [b1; ...; bn]} = \texttt{f (...(f (f a b1) b2)...) bn}.$$

Extend the following function that takes three lists. Write a function

$$\texttt{fold3} : (\texttt{`a} \to \texttt{`b} \to \texttt{`c} \to \texttt{`d} \to \texttt{`a}) \to \texttt{`a} \to \texttt{`b list} \to \texttt{`c list} \to \texttt{`d list} \to \texttt{`a}$$

of which meaning is defined as follow:

$$\begin{gathered} \texttt{fold3 f a [b1; ...; bn] [c1; ...; cn] [d1; ...; dn]} \\ = \texttt{f (...(f (f a b1 c1 d1) b2 c2 d2)...) bn cn dn}. \end{gathered}$$

You may assume that all the given lists are of the same length. □

Exercise 8 Write a function

$$\texttt{(iter}\ n\ \ f\texttt{)} = \underbrace{f \circ \cdots \circ f}_{|n|}$$

The function returns the identity function (fun x $\rightarrow$ x) when $n = 0$.

For example,

$$\texttt{(iter}\ n\ \texttt{(fun x -> x + 2)) 0}$$

returns $2 \times n$. □

Exercise 9 Write a function

```
sigma :  int * int * (int -> int) -> int.
```

such that `sigma(a,b,f)` returns $\Sigma_{n=a}^{b} f(n)$. □

Exercise 10 Write a function

```
cartesian:  'a list -> 'b list -> ('a * 'b) list
```

that returns a list of from two lists. That is, for lists A and B, the Cartesian product $A \times B$ is the list of all ordered pairs (a, b) where $a \in A$ and $b \in B$. For example, if $A = [``a''; ``b''; ``c'']$ and $B = [1; 2; 3]$, $A \times B$ is defined to be

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

□