Homework 1
COSE212, Fall 2018

Hakjoo Oh

Due: 9/30, 24:00

Academic Integrity / Assignment Policy

- All assignments must be your own work.
- Discussion with fellow students is encouraged including how to approach the problem. However, your code must be your own.
  - Discussion must be limited to general discussion and must not involve details of how to write code.
  - You must write your code by yourself and must not look at someone else's code (including ones on the web).
  - Do not allow other students to copy your code.
  - Do not post your code on the public web.
- **Violating above rules gets you 0 points for the entire HW score.**

Problem 1 (5pts) Write a function

```
prime: int -> bool
```

that checks whether a number is prime (n is prime if and only if n is its own smallest divisor except for 1). For example,

```
prime 2 = true
prime 3 = true
prime 4 = false
prime 17 = true
```

Problem 2 (5pts) Write a function

```
range : int -> int -> int list
```

that takes two integers n and m, and creates a list of integers from n to m. For example, `range 3 7` produces `[3;4;5;6;7]`. Assume that $n \leq m$.

Problem 3 (10pts) Write a function

```
dfact : int -> int
```

that computes double-factorials. Given a non-negative integer n, its double-factorial, denoted $n!!$, is the product of all the integers of the same parity as n from 1 to n. That is, when n is even

$$n!! = \prod_{k=1}^{n/2} (2k) = n \cdot (n-2) \cdot (n-4) \cdots 4 \cdot 2$$

and when n is odd,

$$n!! = \prod_{k=1}^{(n+1)/2} (2k-1) = n \cdot (n-2) \cdot (n-4) \cdots 3 \cdot 1$$

For example, $7!! = 1 \times 3 \times 5 \times 7 = 105$ and $6!! = 2 * 4 * 6 = 48$.

Problem 4 (10pts) Define the function `iter`:

```
iter : int * (int -> int) -> (int -> int)
```

such that

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

When $n = 0$, $\texttt{iter}(n, f)$ is defined to be the identity function. When $n > 0$, $\texttt{iter}(n, f)$ is the function that applies f repeatedly n times. For instance,

```
iter(n, fun x -> 2+x) 0
```

evaluates to $2 \times n$.

Problem 5 (10pts) Natural numbers are defined inductively:

$$\frac{}{0} \qquad \frac{n}{n+1}$$

In OCaml, the inductive definition can be defined by the following a data type:

```
type nat = ZERO | SUCC of nat
```

For instance, `SUCC ZERO` denotes 1 and `SUCC (SUCC ZERO)` denotes 2. Write two functions that add and multiply natural numbers:

```
natadd : nat -> nat -> nat
natmul : nat -> nat -> nat
```

For example,

```
# let two = SUCC (SUCC ZERO);;
val two : nat = SUCC (SUCC ZERO)
# let three = SUCC (SUCC (SUCC ZERO));;
val three : nat = SUCC (SUCC (SUCC ZERO))
# natmul two three;;
- : nat = SUCC (SUCC (SUCC (SUCC (SUCC (SUCC ZERO)))))
# natadd two three;;
- : nat = SUCC (SUCC (SUCC (SUCC (SUCC ZERO))))
```

Problem 6 (10pts) Consider the inductive definition of binary trees:

$$\overline{n}\ n \in \mathbb{Z} \qquad \frac{t}{(t, \mathbf{nil})} \qquad \frac{t}{(\mathbf{nil}, t)} \qquad \frac{t_1 \quad t_2}{(t_1, t_2)}$$

which can be defined in OCaml as follows:

```
type btree =
  | Leaf of int
  | Left of btree
  | Right of btree
  | LeftRight of btree * btree
```

For example, binary tree $((1, 2), \mathbf{nil})$ is represented by

```
Left (LeftRight (Leaf 1, Leaf 2))
```

Write a function that exchanges the left and right subtrees all the ways down. For example, mirroring the tree $((1, 2), \mathbf{nil})$ produces $(\mathbf{nil}, (2, 1))$; that is,

```
mirror (Left (LeftRight (Leaf 1, Leaf 2)))
```

evaluates to

```
Right (LeftRight (Leaf 2, Leaf 1)).
```

Problem 7 (10pts) Consider the following propositional formula:

```
type formula =
 | True
 | False
 | Not of formula
 | AndAlso of formula * formula
 | OrElse of formula * formula
 | Imply of formula * formula
 | Equal of exp * exp
and exp =
 | Num of int
 | Plus of exp * exp
 | Minus of exp * exp
```

Write the function

```
eval : formula -> bool
```

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

```
eval (Imply (Imply (True,False), True))
```

evaluates to *true*, and

```
eval (Equal (Num 1, Plus (Num 1, Num 2)))
```

evaluates to *false*.

Problem 8 (10pts) Write a higher-order function

```
all : ('a -> bool) -> 'a list -> bool
```

which decides if all elements of a list satisfy a predicate. For example,

```
all (fun x -> x mod 2 = 0) [1;2;3]
```

evaluates to false while

```
all (fun x -> x > 5) [7;8;9]
```

is true.

Problem 9 (10pts) Write a higher-order function

```
drop : ('a -> bool) -> 'a list -> 'a list
```

which removes elements of a list while they satisfy a predicate. For example,

```
drop (fun x -> x mod 2 = 1) [1;3;5;6;7]
```

evaluates to `[6;7]` and

```
drop (fun x-> x > 5) [1;3;7]
```

evaluates to `[1;3;7]`.

Problem 10 (10pts) Write a function

```
lst2int : int list -> int
```

which converts a list of integers to an integer. For example;

```
lst2int [2;3;4;5] = 2345.
```

Problem 11 (10pts) Write a function

```
concat: 'a list list -> 'a list
```

which makes a list consisting of all the elements of a list of lists. For example,

```
concat [[1;2];[3;4;5]] = [1;2;3;4;5]
```