Programming Languages (COP 4020/6021) [Spring 2016]

Assignment II

Objectives

1. To understand and use higher-order functions, including map and fold.
2. To continue practicing writing functional programs, using ML features like Currying.

Due Date: Sunday, February 7, 2016 (at 11:59pm).

Machine Details: Complete this assignment by yourself on the following CSEE network computers: c4lab01, c4lab02, ..., c4lab20. These machines are physically located in ENB 220. Do not use any server machines like grad, babbage, sunblast, etc. You can connect to the C4 machines from home using SSH. (Example: Host name: *c4lab01.csee.usf.edu* Login ID and Password: <your NetID username and password>) You are responsible for ensuring that your programs compile and execute properly on these machines.

Assignment Description

(0) Read Sections 5.1, 5.3-5.6, and 6.1-6.3 of the *Elements of ML Programming* textbook.

(1) Create a file called *sets2.sml*. In that file, re-implement all ten undergraduate-level functions from Assignment I (i.e., `cardinality`, `elementOf`, `subset`, `equals`, `union`, `funion`, `intersection`, `fintersection`, `complement`, and `cartesian`), without defining any extra top-level functions.

For this second assignment, your functions must:

a) Never call built-in/library functions, *except* `map`, `foldr`, and `foldl`.
b) Never be recursive. All recursion must occur within `map` and `fold`s.
c) Never contain let-expressions. The keyword "let" should never appear in *sets2.sml*.
d) Never cause side effects (such as I/O or pointer/array operations) to occur.
e) Match the types shown on the next page (*note that arguments must be Curried*), without defining any extra helper functions. As in Assignment I, any function *f* defined after function *g* may invoke *g* (e.g., `funion` may invoke `union`).
f) Be commented and formatted properly (again please use spaces instead of tabs); as part of this restriction, limit line widths to 100 characters.
g) Use ML constructs like pattern matching and anonymous variables when appropriate.
h) Make the same data-structure-invariant assumptions as on Assignment I (e.g., assume that incoming set arguments are valid and guarantee that returned sets are valid).
i) Compile and execute on the C4 machines with no errors or warnings.
j) Not be significantly more complicated than necessary.
k) Be reasonably efficient.

Extra Credit for Undergraduates

The `cartesian` function is worth 10% extra credit for undergraduates.

Function Types

```
- use "sets2.sml";
[opening sets2.sml]
val cardinality = fn : int list * bool list -> int
val elementOf = fn : int -> int list * bool list -> bool
val subset = fn : int list * bool list -> int list * bool list -> bool
val equals = fn : int list * bool list -> int list * bool list -> bool
val union = fn
  : int list * bool list -> int list * bool list -> int list * bool list
val funion = fn : (int list * bool list) list -> int list * bool list
val intersection = fn
  : int list * bool list -> int list * bool list -> int list * bool list
val fintersection = fn : (int list * bool list) list -> int list * bool list
val complement = fn : int list * bool list -> int list * bool list
val cartesian = fn
  : int list * bool list
    -> int list * bool list -> (int * int) list * bool list
val it = () : unit
```

Hints

My *sets2.sml* is 45 lines of code (not counting comments/whitespace) and took me 1-2 hours to write and test.

To select the i^{th} element of a tuple T, use `#i T`. For example, `#2(3,4,5,6)` returns 4. This selection operation may be useful if, for example, you want to fold through a list to build a tuple, but then just return one element of the final tuple value.

You can test your implementations on the examples shown in the Assignment I handout, though for this assignment arguments need to be curried.

The complement function can (and should ☺) be implemented quite elegantly.

Submission Notes

- Type the following pledge as an initial comment in your *sets2.sml* file: "I pledge my Honor that I have not cheated, and will not cheat, on this assignment." Type your name after the pledge. Not including this pledge will lower your grade 50%.
- Upload and submit your *sets2.sml* file in Canvas.
- You may submit your assignment in Canvas as many times as you like; we will grade your latest submission.
- For every day that your assignment is late (up to 3 days), your grade reduces 10%.