Computer Science 54

Assignment 0

Friday, September 2@ 5pm

LAST NIGHT I DRIFTED OFF WHILE READING A ~~LISP~~ SML BOOK.
HUH?
SUDDENLY, I WAS BATHED IN A SUFFUSION OF BLUE.
AT ONCE, JUST LIKE THEY SAID, I FELT A GREAT ENLIGHTENMENT. I SAW THE NAKED STRUCTURE OF ~~LISP~~ SML CODE UNFOLD BEFORE ME.
MY GOD
IT'S FULL OF 'car'S
THE PATTERNS AND METAPATTERNS DANCED. SYNTAX FADED, AND I SWAM IN THE PURITY OF QUANTIFIED CONCEPTION. OF IDEAS MANIFEST.
TRULY, THIS WAS THE LANGUAGE FROM WHICH THE GODS WROUGHT THE UNIVERSE.
NO, IT'S NOT.
IT'S NOT?
I MEAN, OSTENSIBLY, YES. HONESTLY, WE HACKED MOST OF IT TOGETHER WITH PERL.

slightly modified from: `https://xkcd.com/224/`

This assignment is mostly to check out that you have access to and can use SML. Start early, particularly on getting SML up and running on either your laptop or the lab computers (whichever you plan to use). Corey (our sysadmin) will have office hours in Edmunds 218 (upstairs): Wednesday, 1-2:30pm and 4-5pm (before and after class) if you have problems getting it installed.

Basic setup

- Install SML and get a text editor working. The course webpage has instructions on how to do this. If you're working on your laptop, it involves a few steps, so please start ASAP. If you plan to work in the lab, it just involves one small thing and should be fast.
- Setup a working directory for CS54.

  Open either Terminal (Mac) or cygwin (windows), create a directory for your CS54 assignments by typing:

  ```
  mkdir cs54
  ```

  (`mkdir` is short for "make directory").You can then move into this directory by typing:

  ```
  cd cs54
  ```

  (`cd` is short for "change directory").

Template files

Each assignment will have a template file to get you started. The main purpose of the file is to provide a standard format to ease our task of grading a large number of submissions. You will find the template file linked from the course web pages—near where you found the assignment. For this assignment the file is named

```
assign0-template.sml
```

Download and save this file into your “cs54” directory/folder and name it

```
assign0.sml
```

In Terminal/cygwin if you now type:

```
ls
```

it will list the contents of that folder/directory and you should now see the `assign0.sml` file there.

Open this file in your editor (probably, Sublime) and then put your solutions to the problems in this file. Make sure that you put your code in the appropriate place in the file (it’s critical for grading!) Here is a sampling from the template file for this assignment.

```
(*
 *          ⟵ Put your name and the date in the header comment here
 *
 * CS 54
 * Assignment 0
 *
 *)
(*** $-* ***)

(*** Problem 1   $+00_01 ***)
(* numList : int -> int list *)
{                                  } ⟵ Place your solution to Problem 1 in this area

(*** Problem 2   $-00_01 $+00_02 ***)
(* sumList : int -> int *)



(*** Problem 3   $-00_02 $+00_03 ***)
(* sumFormula : int -> int *)
```

Comments of the form `(*** ... ***)` are delimiters used by the grading scripts. **Do not change or delete them.** Place your entire solution to a given problem in the space between the delimiters, as shown above. You may include multiple functions as long as they appear between the problem delimeters.

The problems are at the end of this handout.

Check files

Most assignments will also come with a check file to give your submission a quick check before it is submitted. Like the template file, it is designed to give you early feedback (did you follow the basic instructions!) as well as help with grading. For this assignment the file is named

```
assign0-check.sml
```

You can find it in the same place as the assignment and the template file. Copy it to your cs54, and just before you turn in your assignment, check it by typing

```
sml assign0-check.sml
```

at the command line (not inside the SML system). If you see the message

```
Basic check successfully passed!
```

then your file has all the required functions and values, and they have the right types. If you see error messages, then you must go back and correct the assignment before submitting it.

Keep in mind that the check file verifies *only* the function names and types. It does not check functions for correctness.

Please follow the instructions for the template and check files accurately and carefully!

Get in the habit of running the check file immediately before submitting an assignment. Run it again after a quick last-minute correction—that is a time when many trivial errors slip into your code.

The assignment

The main goal of this assignment is to be able to create an SML file, correct errors in it, run it, and submit it.

Take the time to read each line of your submission carefully and understand the significance of the system's responses. Feel free to ask questions if you find anything confusing or you encounter problems.

Also, pay attention to the formatting and comments. Read the course's Style Guide in the document *A Brief Introduction to SML,* and make sure that your future submissions follow it.

Because this is a warm-up assignment, the solutions for most of the problems are provided to you (in blue). All you have to do is to copy them into your file `assign0.sml`. For the last few problems, you will need to write out your own solutions. A completed sample solution appears at the end of this assignment. Although you are not expected to understand all of the code completely at this point in the course, take the time to look carefully at them and try to understand the general ideas. We will discuss the details in our first few classes.

1. Write a function `numList` with the property that `numList n` is a list of integers from `n` down to 1.

```
numList : int -> int list
```

```
fun numList n = if 0 < n then
                   n::(numList (n-1))
                else
                   nil;
```

2. Write a function `sumList` with the property that `sumList n` is the sum of the integers from `n` down to 1.

```
sumList : int -> int
```

```
fun sumList n = if 0 < n then
                   n + sumList (n-1)
                else
                   0;
```

3. Write a function `sumFormula` with the property that `sumFormula n` gives the same result as `sumFormula n` but uses the mathematical formula $n + (n - 1) + \ldots + 1 = \frac{n(n+1)}{2}$.

```
sumFormula : int -> int
```

```
fun sumFormula n = n * (n+1) div 2;
```

4. Write a function `sameValues` that returns `true` when given two lists with the same lengths and the same values. Use list recursion—even though it would be easier to write simply `listA=listB`. We ask you to write it in the more complicated way here to illustrate list recursion.

```
sameValues : 'a list -> 'a list -> bool
```

```
fun sameValues nil     nil     = true
  | sameValues (x::xs) (y::ys) = x=y andalso sameValues xs ys
  | sameValues _       _       = false;
```

5. Verify that the two ways of computing the sum of the first `n` integers give the same results, at least for the values from 47 down to 1. First create a list of integers from 47 down to 1, and then create two lists by applying the two functions to each element of the list. Then see if the two lists are equal, using `sameValues`.

```
val baseList = numList 47;
val siValues = map sumList baseList;
val sfValues = map sumFormula  baseList;
sameValues siValues sfValues;
```

When you are finished with the above problems and run your program, you should see something like the following in the terminal window. The purple parts are what you type in the terminal.

```
assign0 28$ sml
Standard ML of New Jersey v110.78 [built: Sun Dec 21 16:30:08 2014]
- use "assign0.sml";
[opening asgt00.sml]
val numList = fn : int -> int list
val sumList = fn : int -> int
val sumFormula = fn : int -> int
asgt00.sml:37.40 Warning: calling polyEqual
val sameValues = fn : ''a list -> ''a list -> bool
val baseList = [47,46,45,44,43,42,41,40,39,38,37,36,...] : int list
val slValues = [1128,1081,1035,990,946,903,861,820,780,741,703,666,...]
  : int list
val sfValues = [1128,1081,1035,990,946,903,861,820,780,741,703,666,...]
  : int list
val it = true : bool
val it = () : unit
-
```

6. Write a function `isEven` that takes as input an integer and returns `true` if it is even and `false` otherwise. The `mod` operator calculates the remainder (e.g., `5 mod 3` would give 2 as the answer.

```
isEven : int -> bool
```

7. Write a function `circleArea` that takes as input a number representing a radius and returns the area of a circle with that radius.

Why are the input and return type real?

```
circleArea : real -> real
```

8. The previous function doesn't really do the right thing when we give it a value less than 0. Write a function `circleArea2` that return 0.0 if the radius is less than 0, otherwise, it calculates the circle area normally.

Remember that the comparison operators require that both things being compared are the same type (int or real).

```
circleArea2 : real -> real
```

When you're done

When you're ready to submit, run the assignment checker (see above). Make sure that you always run the checker before submitting.

Submit your assignment via Gradescope.

The sample program

```
(*
 * YourName
 * TheDate
 *
 * CS 54
 * Assignment 0
 *
 *)
(*** $-* ***)

(*** Problem 1   $+00_01 ***)
(* numList : int -> int list *)

fun numList n = if 0 < n then
                   n::(numList (n-1))
                else
                   nil;


(*** Problem 2   $-00_01 $+00_02 ***)
(* sumList : int -> int *)

fun sumList n = if 0 < n then
                   n + sumList (n-1)
                else
                   0;


(*** Problem 3   $-00_02 $+00_03 ***)
(* sumFormula : int -> int *)

fun sumFormula n = n * (n+1) div 2;


(*** Problem 4   $-00_03 $+00_04 ***)
(* sameValues : 'a list -> 'a list -> bool *)

fun sameValues nil     nil     = true
       | sameValues (x::xs) (y::ys) = x=y andalso sameValues xs ys
       | sameValues _       _       = false;


(*** Problem 5   $-00_04 $+00_05 ***)
(* Verification that the two computations give equal values *)

(*
 *  Create a list of integers from 47 down to 1, and create
 *  two lists by applying the two functions to each element
 *  of the list.  Then see if the two lists are equal.
 *
 *)
val baseList = numList 47;
val slValues = map sumList baseList;
val sfValues = map sumFormula  baseList;
sameValues slValues sfValues;

(*** Problem 6   $-00_05 $+00_06 ***)
(* isEven : int -> bool *)



(*** Problem 7   $-00_06 $+00_07 ***)
```

```
(* circleArea : real -> real *)



(*** Problem 8   $-00_07 $+00_08 ***)
(* circleArea2 : real -> real *)



(*** $-00_08 ***)
```