Programming Languages (COP 4020/6021) [Fall 2017]

Assignment I

Objectives

1. To become acquainted with the SML/NJ compiler.
2. To understand basic ML constructs such as lists, functions, pattern matching, anonymous variables, and let-environments.
3. To gain experience defining recursive functions in a functional programming language.

Due Date: Monday, January 23, 2017 (at 5pm).

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 1-3.6.3 and 4.1 of the *Elements of ML Programming* textbook.

1) Let's represent two-variable polynomials as `int list lists`. These polynomials will be in terms of the variables *x* and *y*. The n^{th} item in a high-level list will represent all the polynomial terms containing y^n, and the m^{th} item in a low-level list will represent all the polynomial terms containing x^m (where *n* and *m* are natural numbers; recall that the natural numbers are 0, 1, 2, etc). The actual integers in low-level lists are the coefficients of the corresponding polynomial terms.

   For example, the list:

   ```
   [[1], [0,~2,0,0,3], [], [], [], [], [~5,0,7]]
   ```

   represents the polynomial: $1 - 2xy + 3x^4y - 5y^6 + 7x^2y^6$.

   In this assignment, we start counting items in lists at 0, so the 0^{th} item in the high-level list above is the low-level list `[1]`, which represents all polynomial terms containing y^0. The 0^{th} item in this low-level list is 1, so our polynomial contains the term $1x^0y^0 = 1$.

   Similarly, the 6^{th} item in the high-level list above is the low-level list `[~5,0,7]`, so this low-level list represents all polynomial terms containing y^6:

   - Because `~5` is the 0^{th} item in this low-level list, we have a polynomial term $-5x^0y^6 = -5y^6$.
   - Because 0 is the 1^{st} item in this low-level list, we have a polynomial term $0x^1y^6 = 0$.
   - Because 7 is the 2^{nd} item in this low-level list, we have a polynomial term $7x^2y^6$.

   Please note the empty lists in the example above. An empty list as the n^{th} item in a high-level polynomial list just indicates that there are no polynomial terms containing y^n.

   Using this representation of polynomials as lists of lists of integers, implement the following functions in a file named *xy.sml*:

(a) `printxy`
This function prints a 2-variable polynomial. For example:

```
printxy([[1], [0,~2,0,0,3], [], [], [], [], [~5,0,7]])
```

prints `1 - 2xy + 3x^4y - 5y^6 + 7x^2y^6` and returns a unit value `()`.

Notes:

- Exponents are preceded by the "caret" symbol ^.
- Variables having exponent-value 0 are not printed (nor are exponents of value 0).
- Exponents of value 1 are not printed.
- Only terms having nonzero coefficients are printed, unless the entire polynomial lacks nonzero coefficients, in which case `printxy` must still print a `0`. For example, calling `printxy` with argument `[[],[0,0,0],[],[0]]` causes a `0` to be printed.
- Coefficients of value 1 are not printed, unless both the x and y exponents are 0. For example $1x^0y^0$ gets printed as `1`, but $1x^2y^2$ gets printed as `x^2y^2`.
- Subtracted terms are printed with – signs (not ~ signs).
- The initial term, if positive, should not be preceded by a + sign.
- `printxy` must always print a newline (`\n`) after a polynomial.
- I believe `printxy` is the most intricate and challenging function this assignment asks you to implement. Nonetheless, I recommend trying to implement `printxy` first because printing polynomials will be helpful for testing and debugging your code in the rest of the assignment.

(b) `evalxy`
This function evaluates a 2-variable polynomial for given integer *x* and *y* values. For example:

```
evalxy([[1],[0,~2,0,0,3],[],[],[],[],[~5,0,7]], ~1, 1)
```

evaluates $1 - 2xy + 3x^4y - 5y^6 + 7x^2y^6$ with x = -1 and y = 1, returning the value 8.

(c) `multxy`
This function produces a new 2-variable polynomial by multiplying an existing 2-variable polynomial by an integer scalar. For example:

```
multxy([[1],[0,~2,0,0,3],[],[],[],[],[~5,0,7]], 4)
```

returns the value `[[4],[0,~8,0,0,12],[],[],[],[],[~20,0,28]]`, which represents the polynomial $4 - 8xy + 12x^4y - 20y^6 + 28x^2y^6$.

(d) `paddxy`
This function produces a new 2-varaiable polynomial by adding 2 existing 2-variable polynomials. For example:

```
paddxy([[1],[0,~2,0,0,3],[],[],[],[],[~5,0,7]],[[~1],[~1]])
```

returns the value `[[0],[~1,~2,0,0,3],[],[],[],[],[~5,0,7]]`, which represents the polynomial -y - 2xy + $3x^4y$ - $5y^6$ + $7x^2y^6$ (i.e., the sum of 1 - 2xy + $3x^4y$ - $5y^6$ + $7x^2y^6$ and -1 - y).

(e) `pmultxy` [Note: implementing this function is extra credit for undergraduates.]
This function produces a new 2-variable polynomial by multiplying 2 existing 2-variable polynomials. For example:

```
pmultxy([[1],[0,~2,0,0,3],[],[],[],[],[~5,0,7]],[[~1],[~1]])
```

returns the value

```
[[~1],[~1,2,0,0,~3],[0,2,0,0,~3],[],[],[],[5,0,~7],[5,0,~7]]
```

which represents the polynomial

$$-1 - y + 2xy - 3x^4y + 2xy^2 - 3x^4y^2 + 5y^6 - 7x^2y^6 + 5y^7 - 7x^2y^7$$

(i.e., the product of 1 - 2xy + $3x^4y$ - $5y^6$ + $7x^2y^6$ and -1 - y).

Sample Execution

```
> sml
Standard ML of New Jersey v110.67 [built: Mon Aug 11 10:54:32 2008]
- use "xy.sml";
[opening xy.sml]
[autoloading]
[library $SMLNJ-BASIS/basis.cm is stable]
[autoloading done]
val printxy = fn : int list list -> unit
val evalxy = fn : int list list * int * int -> int
val multxy = fn : int list list * int -> int list list
val paddxy = fn : int list list * int list list -> int list list
val pmultxy = fn : int list list * int list list -> int list list
val it = () : unit
- val P = [[1], [0,~2,0,0,3], [], [], [], [], [~5,0,7]];
val P = [[1],[0,~2,0,0,3],[],[],[],[],[~5,0,7]] : int list list
- val Q = [[~1],[~1]];
val Q = [[~1],[~1]] : int list list
- printxy(P);
1 - 2xy + 3x^4y - 5y^6 + 7x^2y^6
val it = () : unit
- printxy(Q);
-1 - y
val it = () : unit
- evalxy(P,~1,1);
val it = 8 : int
- evalxy(Q,~1,1);
val it = ~2 : int
- multxy(P,4);
val it = [[4],[0,~8,0,0,12],[],[],[],[],[~20,0,28]] : int list list
- printxy(multxy(P,4));
4 - 8xy + 12x^4y - 20y^6 + 28x^2y^6
val it = () : unit
- printxy(multxy(Q,~1));
1 + y
val it = () : unit
```

```
- paddxy(P,Q);
val it = [[0],[~1,~2,0,0,3],[],[],[],[],[~5,0,7]] : int list list
- printxy(paddxy(P,Q));
-y - 2xy + 3x^4y - 5y^6 + 7x^2y^6
val it = () : unit
- pmultxy(P,Q);
val it = [[~1],[~1,2,0,0,~3],[0,2,0,0,~3],[],[],[],[5,0,~7],[5,0,~7]]
  : int list list
- printxy(pmultxy(P,Q));
-1 - y + 2xy - 3x^4y + 2xy^2 - 3x^4y^2 + 5y^6 - 7x^2y^6 + 5y^7 - 7x^2y^7
val it = () : unit
```

Grading

For all students, Function (a) is worth 40% of the assignment grade. For undergraduates, Functions (b)-(d) are each worth 20% of the assignment grade, and Function (e) is worth 10% extra credit. For graduate students, Functions (b)-(e) are each worth 15% of the assignment grade.

For full credit, your implementation must:

- be commented and formatted appropriately. To make it easier for the TA to grade, use spaces instead of tabs for indentation.
- use anonymous variables, pattern matching, and let-environments when appropriate (e.g., define all helper functions in let-environments).
- compile on the C4 machines with no errors or warnings.
- not use any advanced ML features that cause *side effects* to occur (e.g., I/O or pointer use). The one exception is that your implementation of printxy is expected to invoke the built-in `print` function, which has the I/O side effect of printing a string.
- not use any built-in (i.e., predefined) functions besides `print` and `Int.toString`.
- not be significantly more complicated than is necessary.

Please note that we will test submissions on inputs not shown in the examples above.

Hints

It took me 4-5 hours to implement and test my *xy.sml*, which is 60 lines of code (not counting whitespace/comments). If, after completely reading Sections 1-3.6.3 and 4.1 of the textbook, you find yourself spending a significant amount of time (e.g., more than 12 hours) on this assignment, please visit or email the teaching assistant to ask for help with whatever problems you are having.

Submission Notes

- Type the following pledge as an initial comment in your *xy.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 *xy.sml* file in the Canvas folder for this assignment.
- You may submit your assignment in Canvas as many times as you like; we will grade your latest submission.
- You may submit your assignment late (between 5pm on 1/23 and 5pm on 1/25) with a 15% penalty.