Lab 9

Due Wednesday, April 23 by 11:59pm

Coding Guidelines

Each question in this assignment should go into the appropriate project directory. For example, the solution to question 1 should be in a folder called "q1". When a solution is a program, one should be able to cd into the question directory and then run your program by typing the command "dotnet run", with additional arguments depending on the question.

Every program should be split into two pieces: a "Program.fs" file that contains the main method and associated program-startup helpers (if needed), and another "Library.fs" file that contains the function(s) of interest in the question. Library code should be contained within a module named "CS334". Be sure to provide usage output (defined in main) for all programs that require arguments. For full credit, your program should both build and run correctly.

_____ Turn-In Instructions _____

Turn in your work using your assigned git repository. The name of your repository will have the form https: //aslan.barowy.net/cs334-s25/cs334-lab09-<USERNAME>.git. For example, if your CS username is abc1, the repository would be https://aslan.barowy.net/cs334-s25/cs334-lab09-abc1.git.

You should have received an invite to commit to the repository via email. If you did not receive an email, please contact me right away!

_____ Group Programming Assignment _____

This is a <u>partner lab</u>. You may work with another classmate if you wish, and you may co-develop solutions. Remember: although you can work on code together, you must each independently write up and submit your solution. No code copying is allowed. Tell me who your partner is by committing a collaborators.txt file to your repository. Be sure to commit this file whether you worked with a partner or not. If you worked by yourself, collaborators.txt should contain something like "I worked by myself." (5 points)

This assignment is due on Wednesday, April 23 by 11:59pm.

_____ Reading _____

- 1. (Required) "Evaluation"
- 2. (Recommended) "Appendix A: Original SML Specification"
- 3. (Recommended) "A Logo Primer"

Problems

Q1. (45 points) Evaluation

In this problem, you will build a complete programming language that calculates the derivative of a polynomial expression. In Backus-Naur form, a polynomial is defined as follows.

where \Box is a mandatory space character. For example, both $3x^4 + 2x^2$ and $1x^2$ are valid polynomials in this language, but x^5 and 3y are not. A term must always include a coefficient and an exponent, and the variable must always be x. Observe that polynomial expressions are strictly defined by term addition. To represent subtraction of terms, like $x^2 - 1$, we must rewrite the expression in the form above with a negative coefficient, like $1x^2 + -1x^0$.

Your interpreter should use only the following AST types.

```
type Term = { coeff: int; exp: int }
type Polynomial = Term list
```

Above, we are using an F# feature called a <u>record</u> in our definition of **Term**. A record works much like a tuple, except that the name associated with a record's element is meaningful, unlike with tuples. For example, we can instantiate a **Term** like so,

let t = { coeff = 3; exp = 5 }

and we can access one of the Term's elements like so

printfn "%d" t.coeff

Observe that our Term type does not explicitly represent the variable in a term. You should assume that all valid Terms are always implicitly defined in terms of x. Consequently, a term that consists only of a constant must include the exponent 0.

(a) Start by defining a function with the following signature

evalTerm: Term -> Term

that computes the derivative of a single term using the power rule. Let $f(x) = cx^e$ where $c \in \mathbb{Z}$ and $e \in \mathbb{Z}$. The power rule states that the derivative of \overline{f} is

$$f'(x) = cex^{e-1}$$

For example, if evalTerm is given the Term with the value { coeff = 4; exp = 5 }, evalTerm returns { coeff = 20; exp = 4 }. Note that the above definition implies that when e = 0, the derivative is 0. In this case, evalTerm should return { coeff = 0; exp = 1 }, which we will use as our standard representation of the constant zero.

(b) Next, define the function

```
evalPoly: Polynomial -> Polynomial
```

which returns the derivative of an entire polynomial expression by calling evalTerm as appropriate. The polynomial returned by evalPoly should also order its terms from the highest to the lowest degree. To sort, use the List.sortByDescending function.

(c) Define a parser function

parse : string -> Polynomial option

where the function returns Some AST after a successful parse and None when the input is invalid.

(d) Define a pretty-print function

prettyPrint: Polynomial -> string

that prints out the string representation of a polynomial. This function should have the property that for any valid polynomial expression string s,

prettyPrint (parse s) = s

when ignoring the inconvenient fact that **parse** returns a **Polynomial option** instead of a **Polynomial**. In other words, in principle, we can compute a repeated derivative by pasting the output of our interpreter back into the interpreter's input.

(e) Finally, define a main method that reads a polynomial expression from the command line, parses it, and then prints the pretty-printed derivative. As usual, your program should not throw exceptions when the user supplies bad input. You may use the following usage function if you wish.

```
let usage() =
    printfn "Usage: dotnet run <polynomial>"
    printfn "\twhere <polynomial> has the form <c_1>x^<e_1> + ... + <c_n>x^<e_n>,"
    printfn "\tfor example, \"3x^5 + -5x^2\""
    exit 1
```

When you run your interpreter with the input below, you should see output like the following:

\$ dotnet run "2x^5 + -1x^1" 10x^4 + -1x^0

The project directory for this question should be called "q1". You should be able to run your program on the command line by typing, for example, "dotnet run " $3x^5 + -5x^2$ "".

Q2. (50 points) Project Description

For this question, you will start describing your final project: a programming language of your own design. You must submit your specification as a LATEX document.

Many programming language specifications begin as informal documents, and this is the template that we will follow in this class. With each stage of your project, you will revisit this document, making it clearer and more precise as you work. It will be a "living document." By the end of the semester, your final specification will include a formal syntax and an informal, but precise, description of your language's semantics. It should clearly document your software artifact, which will be an interpreter for the language. For now, we will start informally, and you should feel free to borrow text from your brainstorm.

Goals

To help you see where we are going, here are the goals of the final project. Your language need not be (and I discourage you from trying to build) a Turing-complete programming language. Instead, focus on solving a single kind of problem. In other words, design a *domain-specific programming language*.

By the end of the semester, your project must achieve all of the following objectives:

- (a) It should have a grammar capable of expressing either an infinite or practically-infinite number of possible programs.
- (b) It should have a parser that recognizes a syntatically-correct program, outputting the corresponding abstract syntax tree.
- (c) It should have an evaluator capable of interpreting any valid AST.
- (d) The language should do some computational work.

If it is convenient to solve your problem by reducing to or extending a lambda calculus interpreter, you may propose such a solution. However, in most cases, it will likely be easier to design a purpose-built interpreter.

Structure of the Specification

Version 1.0 of your specification should explicitly include the following sections. The purpose of this document is to convince yourself that your language implementation is possible. If you aren't convinced, add more detail to convince yourself!

(a) <u>Introduction</u>

2+ paragraphs. What problem does your language solve? What makes you think that this problem should have its own programming language?

(b) Design Principles

1+ paragraphs. Languages can solve problems in many ways. What are the aesthetic or technical ideas that guide its design?

(c) Examples

3+ examples. Keeping in mind that your syntax is still informal, sketch out 3 or more sample programs in your language.

(d) Language Concepts

1+ paragraphs. What are the core concepts a user needs to understand in order to write programs? Think in terms of both "primitives" and "combining forms." What are the key ideas and how are they combined?

(e) Syntax

As much as is needed. Sketch out the syntax of the language. For now, this can be an English description of the key syntactical elements and how they fit together. Examples are fine. We will eventually transform this into a formal syntax section written in Backus-Naur Form (BNF). If you prefer to cut to the chase and provide BNF now, you are welcome to do so, but if that feels like a big leap right now, try to describe the syntax in plain language.

 $(f) \ \underline{\rm Semantics}$

As much as is needed. How is your program interpreted? This need not be formal yet, however, you should demonstrate that you've thought about how your program will be represented and evaluated on a computer. It should answer the following questions.

- i. What are the primitive kinds of values in your system? For example, a primitive might be a number, a string, a shape, or a sound. Every primitive should be an idea that a user can explicitly state in a program written in your language.
- ii. What are the combining forms in your language? In other words, how are values combined in a program? For example, your system might combine primitive "numbers" using an operation like "plus." Or perhaps a user can arrange primitive "notes" within a "sequence."
- iii. How is your program evaluated? In particular,
 - A. Do programs in your language read any input?
 - B. What is the effect (output) of evaluating a program? Does the language produce a file or print something to the screen? Use one of your example programs to illustrate what you expect as output.

How to Organize

The project directory for this question should be called "q2". You must use LATEX for your specification.