Programming Assignment 0: Proof of Access

Due: Saturday, January 28, 2023, 11:59 PM.

Write a file named '**p0.sml**' that contains a function named '**epoly**', which accepts as parameters a list of real values a_0 through a_n, and a single real value x. The list contains the coefficients of a polynomial of the form $a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$, where the real x used is the x parameter passed to your function. Your implementation must accept the list of coefficients as the first parameter and the value of x as the second. Your function must return the value of the polynomial specified by the parameters passed to it. Note that as this is a "proof of access" to the CS department's csx server and Canvas, the code for **epoly** is very simple when written efficiently as a recursive algorithm. Thus, you are not required to learn ML in great detail, only enough to write a simple algorithm and use the **sml** interpreter to load and test your code. ML will be discussed in sufficient detail in the first week for you to write the code needed.

Code that does not compile and run will receive no credit.

You must add proper documentation to the code in your submission, describing what is done and why. This an exercise in reading code. Over a career you may well be expected to understand programs written in languages you do not know well, if at all. This isn't as hard as it may sound. It just takes some practice in analysis and a bit of research. Programs that do not include proper documentation will be penalized.

Submit the file '**p0.sml**' to the **handin** folder for CS3363 on the department's csx server using the following command on csx.

```
handin  cs3363-rlchurc  program0  p0.sml
```

You must also submit a screenshot of the session on csx where you have run this code successfully. This must be a full-window image, not an image cropped to just a few lines on-screen. Submit this to the Canvas folder for this programming assignment.

Note: You may need to provide the path to the file you are submitting as part of the file name, if the current working directory does not contain that file.

Please see the information in the "Programming Languages" area of the Canvas / File folder for information about ML, and the "Using csx" area for how to run it on the department's csx server.

Additional instructor comments

First, you should view any programming assignment as you would view a task given to you by your supervisor when you are working professionally. When your boss gives you a specification for something you are to implement, you must satisfy the specification exactly. If you were to give your boss a completed product that did not match the specification given you, odds are you would have a serious problem. The piece you were given must fit in with the overall task your boss is responsible for. Your product must fit where it belongs in the larger scheme.

If you think there is something wrong with a specification given you, you should discuss the problem with your boss. In this case, your boss is your instructor. You cannot deviate from the specification without explicit approval to do so.

ML is a language with a strong affinity for recursive operations. This is true of functional languages. It may well help you to start thinking in the manner best suited to functional languages, as you will be programming in three functional languages this semester.

Also, you should keep in mind as you study recursion that there is a symmetry between correct recursive algorithms and their proofs by induction. The two are mirror twins.

In the case of this assignment, a recursive solution can be extremely simple and direct. The key to finding such a solution is to remember that there are several different ways to write a polynomial. The two ways you see most often are the following.

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n, or$$
$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0.$$

But, there is another way to write the same polynomial, as shown here.

$$f(x) = a_0 + x(a_1 + x(a_2 + x(\cdots)))$$

This third format can easily be translated into a recursive function that evaluates a polynomial.

This illustrates a point I often raise (and sometimes beat people over the head with) on a regular basis: Any question worth asking is worth rephrasing. The reverse is also true: Any answer worth having is worth rephrasing. In rephrasing answers and questions you can often discover their hidden features.

Recursive functions must have at least three characteristics.

1. A recursive function must call itself.
2. A recursive function must have at least one parameter that changes with each successive call to the function.
3. A recursive function must have a stopping condition.

Proper file-heading documentation must include the following at the top of the file.

- Assignment name
- Course and semester
- The instructor’s name
- Due date and time
- Date and time submitted
- Your name and csx user ID
- A description of what the submitted code does, how to compile or load for translation, and how to run the program/code correctly.

As stated above, proper documentation of code is more than just saying what the code does. It includes why it does what it does. Proper code documentation is required. Failure to document code properly entails a 20 percent penalty on the grade for this assignment.