450 Programming Project #2

Due Date: Saturday, October 16, 2010 (10pm)

Overview

Write solutions to the problems below using F#.

1. You have probably had to determine the greatest common divisor, or GCD, of a pair of integers at some time. That is, you were supposed to take two numbers (integers) and find the largest integer that would evenly divide both of the numbers. Write a recursive function named, `gcd`, that returns the GCD of two integers.

   Sample run:

   ```
   > gcd 25 30;;
   val it : int = 5
   > gcd 3 9;;
   val it : int = 3
   ```

2. Write a program named `convertDate` that takes in a date as a string in MM/DD/YYYY format and returns the date as a long date format: Thursday, September 30, 2010

   Sample run:

   ```
   > convertDate "01/02/2010";;
   val it : string = "Saturday, January 02, 2010"
   > convertDate "02/25/1950";;
   val it : string = "Saturday, February 25, 1950"
   ```

3. Write a program that given a string, we count the number of vowels in the string (a, e, i, o, u).

   Sample run:

   ```
   > countVowels "The quick brown fox jumps over the lazy dog";;
   val it: int * int * int * int * int = (1, 3, 1, 4, 2)
   ```

4. Suppose we represented a matrix in f# as a list of lists. For example, [[1; 2]; [3; 4]] would represent a 2x2 matrix whose first row contains the elements 1 and 2, and whose second row has the elements 3 and 4. Write a recursive function that takes a matrix as input, and outputs its transpose.

   Sample run:

   ```
   > transpose [[1; 2];[3; 4]];;
   val it : int list list = [[1; 3]; [2; 4]]
   ```

5. Write a function named, `in2post`, which takes as its argument an algebraic infix notation string. The result of your function is to be a postfix version of the expression, without parentheses. A recommended approach is to use the shunting-yard algorithm, though a recursive decent parser could also work.

   Sample run:

   ```
   > in2post "(a + b) * (c + d))";;
   val it: string = "a b + c d + *"
   ```

Style

Write clean, modular, well-structured, and well-documented code.

Here are some general guidelines:

1. Make sure that your name appears somewhere in each source file.
2. You do not have to document every single line of code you write. Provide enough documentation so someone who knows Lisp/Scheme can understand what your program is doing
3. Names should be indicative of their purpose.

Submitting Your Work

All of your functions should be in a single f# file / module. I will be taking your file and loading it into f# interactive. As such, you only need to submit the f# file. If you would like to include a file explaining design choices you made, be sure to include them in a single file. You may also zip up the entire directory and submit that, if you do not wish to submit just the .fs file. In any event, your files should be ZIP'ed up and submitted to the myCourses dropbox.