COMP 3007 - Assignment #1

Getting started with Scheme.

Due: Friday January 24th @ 11:55pm

Question 1

Rewrite the following expressions as Scheme expressions:

a. [2 marks] `1 + -2 - 3 + 4 + -5 + 6`
b. [2 marks] `20-1+((26/2 + 2)*(20/5 - 2))`
c. [2 marks] `(63/9*(12/4-14/(13+2))+12)/4`
d. [2 marks] `((50*20)*2)+((17/4)+3.85)+((30*2)-48)`

Question 2

For this question you will define a number of functions and illustrate their evaluation.

a. [1 mark] Create a procedure `(cube x)` that computes x^3.
b. [2 marks] Create a procedure that computes the following function:

$$f(x) = 3x^2 + 4$$

c. [2 marks] Create a procedure that computes the following function:

```
g(x) = f(2x) - 2x^3
```

d. [2 marks] Create a procedure that computes the following function:

```
h(x) = 2f(x/2) + g(x)
```

e. [2 marks] Provide the substitution model using **applicative order** for `(h (* 2 3))`.
f. [4 marks] Provide the substitution model using **normal order** for `(h (* 2 3))`.

Question 3

a. [4 marks] The formula for the **Quadratic Equation** is as follows:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Create the procedure `(quadratic a b c)` to calculate the roots of a quadratic formula with coefficient a, b, and c. For simplicity (since we have not yet covered pairs or lists), use only the + version of the equation in place of the ±. The procedure should return false (#f) if there are no real roots (this occurs when the discriminant (b^2 - 4ac) is negative, or if a is 0).

b. [4 marks] Write a procedure called `convert` that takes as arguments: an amount (number), and two strings representing the input and output units respectively. The procedure should support conversions to or from *any* combination of Bytes ("B"), KiloBytes ("KB"), and **KibiBytes** ("KiB"). Any other unit argument should result in the return of an error string.
For a discussion on the difference between KB and KiB see: **wikipedia**, **wolfram**, or **google**
(Note: to compare strings in Scheme you can use the predicate `(equal? str1 str2)`)
E.g. `(convert 42 "KB" "B") → 42000`
E.g. `(convert 42 "KiB" "KB") → 43.008`
E.g. `(convert 42 "KiBapples" "Bytes") → "Could not convert from KiBapples to Bytes"`

Question 4

For this problem you will construct a small program over several separate procedures. You should re-use your own functions wherever possible. Some other built-in functions you may find useful (in addition to those covered in the lecture notes): **(read)**, **(log num)**, **(floor num)**, **(expt num1 num2)**, **(number->string num)**, **(string-append str1 str2)**. You may ignore any minor rounding errors that may occur in this question.

a. [2 marks] Define a procedure called `user-num` that takes a single string representing a prompt as argument. The procedure should display the prompt and read a single value from the user. Finally, ensure that the user entered a valid number and return a default value of 0 if they did not.
E.g. `(user-num "Enter a number: ") → Enter a number: 123 → 123`
E.g. `(user-num "Hello! ") → Hello! hi! → 0`

b. [3 marks] Define a procedure called `sci-exponent` that takes a number as argument and returns the exponent that should be used to display that number in scientific notation. For the purposes of this problem the exponent for the number zero should be 0 (ie 0 = 0x10^0).
E.g. `(sci-exponent 1.234) → 0`
E.g. `(sci-exponent 12345) → 4`
E.g. `(sci-exponent 0.001234) → -3`
E.g. `(sci-exponent -12345) → 4`

c. [2 marks] Define a procedure called `sci-coefficient` that takes a number as argument and returns the coeffiecient that should be used to display that number in scientific notation.
E.g. `(sci-coefficient 1.234) → 1.234`
E.g. `(sci-coefficient 12345) → 1.2345`
E.g. `(sci-coefficient 0.001234) → 1.234`

d. [3 marks] Define a procedure called `sci-num` that takes no arguments. This procedure should first prompt the user to enter a number, and then return a string representing that number in scientific notation.
E.g. `(sci-num) → Enter a number: 12345 → "1.2345x10^4"`
E.g. `(sci-num) → Enter a number: 1.234 → "1.234x10^0"`
E.g. `(sci-num) → Enter a number: 0.001234 → "1.234x10^-3"`
E.g. `(sci-num) → Enter a number: hello → "0x10^0"`

Question 5

[4 marks] The following program can be used to determine if a given interpreter is using applicative-order or normal-order evaluation:

```
(define (test x y)
    (if (= x 0)
        x
        y))

(test 0 (/ 3 0))
```

a. What will be the behaviour of this code on an interpreter that uses applicative-order evaluation? **Explain why.**
b. What behaviour will be observed with an interpreter that uses normal-order evaluation? **Explain why.**

Question 6

[3 marks] Observe that Scheme's model of evaluation allows for combinations where the operator is itself a combination. Use this observation to describe the behaviour of the following procedure:

```
(define (foo a b)
        ((cond ((> b 0) +)((= b 0) *)(else /)) a b))
```

Your answer should describe what happens for all integer values of a and b. Illustrate your answer using the substitution model.

Documentation & Testing [10 marks]

Documentation

- Ensure that your name and student number are in comments at the top of all files.
- Document the purpose of each function including its expected inputs (parameters) and output (return).
- Ensure that your code is well-formatted and easily readable; a happy TA is a generous TA.

Testing

- You are required to include testing runs of every function in your submission.
- The specific tests required depend on the question at hand, but should cover all valid inputs and all possible branches of your code.
- The example runs provided in the guidelines above may not be sufficient.
- Unless otherwise specified, you may assume inputs supplied are of the correct type.
- Fabricated test outputs will result in 0 marks for a question.
- For best practices: Comment your testing as to what you are testing and why, giving expected output as well as observed output and explanations for any differences.

An example submission including documentation and testing can be found here: primes.scm

Submission

- Any files that are not runnable (in DrRacket using R5RS) will result in a mark of 0 for that question.
- Do not use set! (or any procedure with !) in your solutions for this assignment.
- Combine all files into a single .zip file for your submission.
- Submit your assignment using cuLearn before the due date.
- Marks will be deducted for late submissions.
- You are responsible for submitting all files and ENSURING that the submission is completed successfully.
- If you are having issues with submission, contact me **before** the due date, afterwards late deductions will apply.
- Please see the course outline for all submission guidelines.