BU CAS CS 320 (Spring Semester, 2002)
Concepts of Programming Languages

Assignment 1

Out: Monday, 14 January 2002
Due: Monday, 28 January 2002

Academic Integrity We pledge strict adherence to the university guidelines.

- All work you turn in must be *your own* unless specified otherwise.
- You are allowed to discuss problems with your classmates but you must write your own code and solutions.
- Please always remeber that every student deserves a chance to achieve a fair grade!

Total: 250 points

Exercise 1 *(100 points) The following is a description of* Game of 24.

Given four natural numbers n_1, n_2, n_3 and n_4, one chooses two of them and generate a rational *number r_1 using either addition, subtraction, multiplication or division; one mixes r_1 with the remaining two numbers and chooses two of them to generate a rational number r_2 using either addition, subtraction, multiplication or division; one then takes r_2 and the last remaining number to get a rational number r_3 using addition, subtraction, multiplication, or division; if there is a way to make r_3 equal to 24, then we say that (n_1, n_2, n_3, n_4) is a good quad. For instance, $(10, 10, 4, 4)$ is a good quad since we have*

$$(10 * 10 - 4)/4 = 24$$

Similarly, $(5, 7, 7, 11)$ is a good quad since we have

$$(5 - 11/7) * 7 = 24$$

Game of 24 is a game that determines whether four given natural numbers are a good quad.

Please implement a program in your favorite programming language that takes four given natural numbers and returns 1 or 0 according to whether the four natural numbers are a good quad; if they are a good quad, the program should also print out an arithmetic expression that attests to their being a good quad.

Exercise 2 *(10 points) Please translate the following expression into prefix form*

$$\frac{5 + 4 + (2 - (3 - (6 + \frac{4}{5})))}{3(6 - 2)(2 - 7)}$$

Exercise 3 *(20 points) Alyssa P. Hacker doesn't see why* `if` *needs to be provided as a special form. "Why can't I define as an ordinary procedure in terms of* `cond`*" she asks. Alyssa's friend Eval Lu Ator claims this can indeed be done, and she defines a new version of* `if`*:*

```
(define (new-if predicate then-clause else-clause)
   (cond (predicate then-clause)
         (else else-clause)))
```

Delighted, Alyssa uses `new-if` *to rewrite the square-root program:*

```
(define (sqrt-iter guess x)
  (new-if (good-enough? guess x)
      guess
      (sqrt-iter (improve guess x) x)))
```

What happens when Alyssa attempts to use this to compute square roots? Please explain.

Exercise 4 *(30 points) Please implement a procedure that takes three numbers as its arguments and returns true or false according to whether these numbers can be the sides of a triangle. Note that three given real numbers can be the sides of a triangle if and only if the sum of any two of these numbers is greater than the third.*

Exercise 5 *(40 points) Newton's method for cube roots is based on the fact that if y is an approximation to the cube root of x, then the following value*

$$\frac{x/y^2 + 2y}{3}$$

yields a better approximation. Under this formula please implement a cube-root procedure that is analogous to the square-root procedure in the textbook.

Exercise 6 *(20 points) It is said that there is only one natural number n such that $n-1$ is a square and $n+1$ is a cube, that is, $n-1 = x^2$ and $n+1 = y^3$ for some natural numbers x and y. Please implement a procedure in Scheme to find n.*

Exercise 7 *(30 points) McCarthy's 91-function is defined as follows.*

$$f(x) = \begin{cases} f(f(x+11)) & \text{if } x < 101; \\ x - 10 & \text{otherwise.} \end{cases}$$

Please implement a procedure g such that for a given integer x, $g(x)$ returns the number of calls to f during the computation of $f(x)$. For instance, $g(101)$ should return 1 since the computation of $f(101)$ calls f once; $g(100)$ should return 3 since three calls to f are required as shown below.

$$f(100) \to f(f(111)) \to f(101) \to 91$$