CSE 461: Programming Languages Concepts

Prof. G. Tan
Spring 2018

Homework 4: Scheme warming up
Due on Mar 12th before class (12:20pm) in Canvas.

Submission: you should submit via Canvas a DrRacket file with Scheme code in. Please clearly mark your answers using comments so that we can tell the correspondence between your code and questions.

1. (3 points) Go through a brief tutorial about the DrRacket programming environment at `http://docs.racket-lang.org/drracket/index.html`. Be sure to change the language to R5RS.

   Code the Fibonacci function in DrRacket, which is defined as follows:

$$fib(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ fib(n-1) + fib(n-2) & \text{if } n > 1 \end{cases}$$

   To check your result, use DrRacket to compute the result of $fib(20)$, which should be 6765. You only need to submit your Scheme code for the Fibonacci function.

2. (3 points) Hofstadter Female and Male sequences are a pair of mutually recursive functions defined as follows:

$$\begin{aligned} &F(0) = 1 \\ &M(0) = 0 \\ &F(n) = n - M(F(n-1)), \text{when } n > 0 \\ &M(n) = n - F(M(n-1)), \text{when } n > 0 \end{aligned}$$

   Write these two functions in Scheme.

3. (3 points) The Euclid's algorithm for computing the greatest common divisor of two numbers is one of the oldest algorithms. It can be defined recursively as follows (if you are not familiar with how it works, you should use the internet to find out):

$$\text{gcd}(x, y) = \begin{cases} x & \text{if } y = 0 \\ \text{gcd}(y, \text{remainder(x, y)}) & \text{if } y > 0 \end{cases}$$

Turn the gcd function to Scheme code. Scheme has a built-in remainder function, which you can use for this problem. Also, Scheme already has a built-in gcd function, which you shouldn't use. To avoid name conflicts, you should rename your gcd function to be something else such as mygcd.

4. (3 points) Write a higher-order function `ncall` that takes three parameters: n, f, x; it returns x when $n = 0$, returns $f(x)$ when $n = 1$, returns $f(f(x))$ when $n = 2$, etc. That is, it returns the result of calling f on x, for n times. For instance, invoking `ncall` with $n = 4$, the add-one function, and $x = 2$ should return 6.