Elements of Programming/ Introduction to Scheme

September 2008

Course Information

- Instructor: Prof. Kathy McCoy (mccoy@cis.udel.edu)
- Place: 209 Smith

Home page:
http://www.cis.udel.edu/~mccoy/courses/cisc280-08f

Course Syllabus

Computational Processes

- Abstract entity on a computer that manipulates data
- Program – a pattern of precise rules that direct programs
- Programming – creating the rules to create computational processes that presumably perform some task
  - Correct
  - Fast
  - Modular (and able to be debugged)

Why Scheme?

Language features make it excellent for the study of programming constructs/data structures

- Procedure syntax = data syntax
- Simple syntax
- Untyped
- No explicit pointer notation
- Automatic memory management
- Focus on higher-level concepts/symbol manipulation
- Interpreted language
- Ability to easily capture/explain recursive processes

Extending the language

Concept
- Primitive expression
- Combination expression
- Abstraction (treat combination like a primitive)

In Scheme
- Numbers, symbols
- Lists, e.g., (fun arg1 arg2 ...)
- Define operator

Scheme's top level

- Read one expression
- Evaluate that expression
- Print the value of that expression
- Repeat

(the read-eval-print loop)

Evaluation of primitive expressions

- Numbers, strings, etc., evaluate to themselves
- Symbols evaluate to values that have been assigned to them (usually by define)

```
(define x 12)
(define y "this is a string")
```


Evaluation of compound expressions (lists/combinations)

- Evaluate first element of list to obtain a procedure
- If normal procedure: evaluate remaining elements of list, then apply procedure to their values
- If special procedure: apply procedure to rest of list unevaluated

NOTE: the recursive definition of evaluation!

Observations

- Arithmetic operators are normal procedures
- The define operator is a special procedure
- Special procedures are called special forms (also called syntactic forms)
- Note: no "reserved" words. Any symbol (collection of characters not containing a blank) can be defined as a function.
- Names of built-in procedures not special. Can redefine them if you want.

Practice Evaluating Expressions (Combinations) in Scheme

- Note – uses prefix notation, standard arithmetic operations are predefined: +,-,*,/….
- (* 5 99)
- (*5 99)
- (+5 99)
- (*(5 99))
- (* (+ 2 2) 5)
- (* (+ 2 2) (5))
- (*(+(2 2) 5))
- (*(+ 2 2)5)
- (5 * 4)
- (5 * (2 + 2))
- ((+ 2 3))

Procedure Definitions

(define (<name> <param1> <param2> ...)
<body>)

```
(define (square x) (* x x))
(define (fourth-power x)
  (square (square x)))
(define (quadratic a b c x)
  (+ a (* b x) (* c (square x))))
```