

Scheme Examples

September 29, 2008

1 Vector Usage

```
#lang scheme
```

```
; Equivalent to (define games #("Super Mario" "Ping Pong" "Hockey"))  
(define games (vector "Super Mario" "Ping Pong" "Hockey"))
```

```
; How to determine the size of the vector  
(printf "I have ~a favourite games\n" (vector-length games))
```

```
; How to iterate through the vector and print the values  
; Note that the for-each function accepts a function to apply to each element  
(printf "They are:\n")  
(for-each  
  (lambda (x) (printf "~a\n" x))  
  (vector->list games))
```

```
; How to get an individual element  
(printf "The best game ever is: ~a\n" (vector-ref games 0))
```

```
; How to change an element  
(vector-set! games 1 "Tennis")  
(printf "Another decent game is: ~a\n" (vector-ref games 1))
```

2 HashMap Usage

```
#lang scheme
```

```

; This example demonstrates using the hashmap object in Scheme

; First, create a hashmap
(define students (make-hash))

; Add some students and their marks in CS 241 to the hashmap
(hash-set! students "Albert" 80)
(hash-set! students "John" 61)
(hash-set! students "Calvin" 50)
(hash-set! students "Dave" 94)
(hash-set! students "Ken" 75)

; Check if a certain key is in the hashmap
(if (number? (hash-ref students "Dan" false))
    (printf "Dan is in CS 241.\n")
    (printf "Dan is not in CS 241.\n"))

; Check the size of the hashmap
(printf "There are ~a students in CS 241.\n" (hash-count students))

; Iterate over the list of students and print their names and marks
; and also calculate the average
(define average 0)
(hash-for-each
 students
 (lambda (x y) (begin
                 (printf "~a has a mark of ~a%.\n" x y)
                 (set! average (+ average y))))))
(set! average (/ average (hash-count students)))
(printf "The class average is ~a%.\n" average)

```

3 Binary Output

```
#lang scheme
```

```

(write-byte #x48) ; H
(write-byte #x65) ; e
(write-byte #x31) ; l
(write-byte #x31) ; l
(write-byte #x6f) ; o
(write-byte #x20) ; (space character)
(write-byte #x57) ; W
(write-byte #x6f) ; o
(write-byte #x72) ; r
(write-byte #x31) ; l
(write-byte #x64) ; d
(write-byte #x21) ; !
(write-byte #x0a) ; (newline character)

```

4 Outputting an Instruction

```
#lang scheme
```

```

; Write the "jr $31" instruction in binary to standard out
; i.e. 0x03e00008
(define jr #x03e00008)

```

```

(write-byte (bitwise-and (arithmetic-shift jr -24) #xFF)) ; #x00000003
(write-byte (bitwise-and (arithmetic-shift jr -16) #xFF)) ; #x000000e0
(write-byte (bitwise-and (arithmetic-shift jr -8) #xFF)) ; #x00000000
(write-byte (bitwise-and jr #xFF)) ; #x00000008

```