

CS-257L

Nonimperative Programming: Scheme!

Instructor:

Joel Castellanos

e-mail: joel@unm.edu

Web: <http://cs.unm.edu/~joel/>

Office: Farris Engineering
Center (FEC) room 321

```
(define f
  (lambda (n)
    (cond
      ((= n 1) 1)
      ((= n 2) 1)
      (else (+ (f (- n 1)) (f (- n 2))))))
  )
)
```

Homework - Tuesday Night at Midnight

- Due Tuesday Night at Midnight (2/5/2008)
- Create a DrScheme definition file (.scm) that will define the function: prime-factors
 - Takes 1 numeric argument.
 - Returns a list of the prime factors of the argument.
 - Example: (prime-factors 20) returns (2 2 5).
- Use comments.
- Report error if argument is not a positive integer.
- Cannot use library procedures gcd, lcm, nor others not covered in class or in chapters 1 & 2.
- Turn-in via WebCT.

Grading of HW-4

10 pts:

- Runs correctly in DrScheme on all positive integers less than 300 million.
- Correctly reports an error when the input is not a positive integer.
- Is efficient.
- Is well commented.
- Is neat and easy to read.

9 pts:

- All of above, but fails in being one of: efficient, well commented or easy to read.

Grading of HW-4 - continued

8 pts:

- All of above, but fails in more than one of: efficient, well commented or easy to read.

7 pts:

- Runs correctly in DrScheme on all positive integers less than 300 million.
- Does not correctly report an error when the input is not a positive integer.

5 pts:

- Works correctly in most cases, but fails in some special cases (i.e. repeated roots or for $n=1$).

Library Procedures: `quotient`, `remainder`, `modulo`

`(quotient 7 2)` → 3

`(quotient 1 3)` → 0

`(quotient -10 3)` → -3

`(remainder 7 2)` → 1

`(remainder 7 2)` → 1

`(remainder -7 2)` → -1

`(modulo 7 2)` → 1

`(modulo -7 2)` → 1

Library Procedures: `floor`, `ceiling`, `truncate`, `round`

`(floor 3.5)` → 3.0

`(ceiling 3.5)` → 4.0

`(truncate 3.5)` → 3.0

`(round 3.5)` → 4.0

`(ceiling 3.01)` → 4.0

`(floor -4.3)` → -5.0

`(ceiling -4.3)` → -4.0

`(truncate -4.3)` → -4.0

`(round -4.3)` → -4.0

Library Procedures: Algebra and Trig

`(sqrt 81)` → 9

`(sqrt 2)` → 1.4142135623730951

`(sqrt -1)` → 0+1i

`(exp 1)` → 2.718281828459045

`(exp 100)` →
2.6881171418161356e+043

`(log (exp 1))` → 1.0

`(expt 2 9)` → 512

`(sin (/ 3.14159265358979 2))` → 1.0

(l and 1), (1 and l), (1 and 1), (*l and l*)

The textbook has a number of examples where it uses the letter *l* and the number 1.

However;

Please avoid using as a function or list name in the homework.

Michael Murphy - What does `test?` do?

```
(define test?  
  (lambda (x)  
    (cond  
      ((list? x) (size? (cdr x) 1))  
      (else #f)  
    )  
  ) )
```

```
(define size?  
  (lambda (x i)  
    (cond  
      ((> i 5) #t)  
      ((null? x) #f)  
      (else (size? (cdr x) (+ i 1)))  
    )  
  ) )
```

Answer:

- Returns true if given a list of 6 or more elements; otherwise returns false.

Annette Hatch – Forms of and

What is:

<code>(and (= 2 2) (= 2 2))</code>	<code>#t</code>
<code>(and (not (= 2 2)) (= 2 2))</code>	<code>#f</code>
<code>(not (and (= 2 2) (= 2 2)))</code>	<code>#f</code>
<code>(and not (= 2 2) (= 2 2))</code>	<code>#t</code> - undefined
<code>(and 2 2)</code>	<code>2</code> - undefined
<code>(and 4 8)</code>	<code>8</code> - undefined
<code>(and 8 4)</code>	<code>4</code> - undefined

Annette Hatch – Implicit Lambda

What is the output to the following code?

```
(define (incrVar x) (+ x 1))  
(incrVar 2)
```

Answer: 3

This user-defined function increments the input by 1. Lambda is implicit in this form.

Samuel Hopkins – `member?` List in List?

What is the output to the following code?

```
(define haha '( (abc) ))  
(define blah '( (abc) ))  
(member? (car blah) haha)  
(member? (car haha) haha)
```

No Answer, because `member?` works off of `eq?` and `eq?` does not work on lists.

DrScheme gives the inconsistent results:

```
#f
```

```
#t
```

Some will argue that the second is `#t` because both have the same address – but this is C/C++ thinking where the concept of address is a well defined part of the language.