CIS 343 - Structure of Programming Languages

Nathan Bowman

Slides by Ira Woodring

Functional Practice

Practice

Logon to EOS. Start mit-scheme.

Remember to rlwrap it!

```
rlwrap mit-scheme
```

Practice

Problem 1 (the classic):

Write a function called fact that takes a single parameter. The function should return the value of the factorial of the parameter.

Don't look this up. Struggle with it for a few minutes. Try things and see what happens.

```
(define (fact n)
  (cond
    ((<= n 1) 1)
    (else (* n (fact (- n 1))))
  )
)
```

Practice

Problem 2;

Write a function that reverses a list.

```
(define (r a_list)
  (cond
    ((null? a_list) #t)
    (else (r (cdr a_list))(write (car a_list))#t)
  )
)
```

However, there does exist a `reverse` function.

Scheme does allow the use of strings. Strings can be created in a variety of ways, the simplest being

```
"ira"
```

Scheme views characters in a string as ASCII values, and represents them as such:

```
#\i #\r #\a
```

We can make a string into a list, or a list into a string via the `list->string` or `string->list` functions:

```
(define name "ira")
(string->list name)
;Value 40: (#\i #\r #\a)

(define l (list #\i #\r #\a))
(list->string l)
;Value 43: "ira"
```

Practice

Problem 3:

Knowing what you now know, write a function that reverses a string.

```
(define (flip str)
  (list->string (reverse (string->list str))))
```

Alternatively you could have called your own reverse function.

Much like Java we can't compare strings with the `eq?` function. Instead we need to use `string=?`.

```
1 ]=> (eq? "ira" "ira")

;Value: #f

1 ]=> (string=? "ira" "ira")

;Value: #t
```

Practice

Problem 4:

Write a function that checks if a string is a palindrome or not.

It should return either `#t` or `#f`.

Make use of what we already have.

```
(define (pal str)
  (string=? str (flip str))
)
```

Practice

Problem 5:

Write a function that returns the *i*th value in a list.

```scheme
(define (findi my_list i)
  (cond
    ((eq? i 0) (car my_list))
    (else (findi (cdr my_list) (- i 1)))
  )
)
```