Midterm Review

Gilbert Chou
cs3-ta@imail.eecs.berkeley.edu

Navigating Text

C-v – move one screen forward
M-v – move one screen back
C-s/C-r – prompts for text to search. 's' searches forward. 'r' searches back
C-g – cancels a command
C-a – goes to front of the line
C-e – goes to back of the line
M-g – prompts for a line number then goes to it

Cut, Copy, Paste, Undo

C-k – kill the rest of the line
C-w – kill region
'Killing' text is the equivalent of cut

M-w – copy region
C-y – paste
M-y – pressing this after C-y will bring back earlier cuts
C-_ – undo last command

Arithmetic

Addition	+
Subtraction	-
Multiplication	*
Division	/*(decimal)*, quotient*(integerdivision)*
Remainder	remainder
Square root	sqrt

Sample Question #1

Write a procedure that will evaluate expressions like '(1 + 1), '(1 * 1), '(1 / 1), '(1 - 1) '(1 % 1).

Usage: (calc '(1 + 1))

It does not need to evaluate longer expressions like '(1 + 1 + 1).

Sample Question #1 (Sol.)

```
(define (calc expr)
    ((get-procedure expr) (first expr) (last expr)))
(define (second sent) (first (butfirst sent)))

;; NOTE: the word + is not the same as the
;; procedure +
(define (get-procedure expr)
    (cond
        (equal? (second expr) '+) +)
        (equal? (second expr) '-) -)
        (equal? (second expr) '/) /)
        (equal? (second expr) '*) *)))
```

Words and Sentences

- Joiners: word, sentence
- Accessors: first, last, butfirst, butlast
- Misc: item, count

Input and Output

(last (first (butfirst (butlast '(123 456 789)))))

→ 6

(butfirst (last (butfirst '(123 456 789))))

→ 89

(butfirst (butfirst (butfirst '123)))

→ ""

Quote Confusion

```
(define w 'hello)
(sentence (quote (sentence w w)) w)
    → (sentence w w hello)
(sentence (word w w))
    → (hellohello)
(word unbound-variable)
    → error
(quote unbound-variable)
    → unbound-variable
(quote 'unbound-variable))
    → (quote unbound-variable)
```

Sample Question #2

Write procedure that given a word returns the piglatin.

Usage:
(piglatin 'hello) → ellohay
(piglatin 'apple) → appleway

Sample Question #2 (Sol.)

Piglatin is more complicated than I orignally thought, but this is the solution for the version of piglatin I was thinking of.

```
(define (piglatin w)
   (if (member? (first w) '(a e i o u))
         (word w 'way)
         (word (butfirst w) (first w) 'ay)))
```

Conditionals

- Any value that is not #f is considered true.
- 'and' stops executing when it finds the first false value and returns false. Otherwise it evaluates each argument.
- 'or' stops executing when it finds the first true value then returns it. Otherwise it evaluates each argument.

What do these calls return?

(and 1 2 3) → 3

(or #f 'hello unbound-variable) → hello

Predicates

- equal?
- =, <, >, <=, >=
- member?
- and, or, not

Conventions for user defined predicates

'if' and 'cond' Syntax

```
(if <predicate>
    <execute-when-true> <execute-when-false>)


(cond
  (<predicate> <body>)
  (<predicate> <body>)
  ....
  (else <body>))
```

Sample Question #3

```
(define (mystery a b c)
        (cond
                (c (a b c))
                ((and a b c) 'hello)
                ((or a b c) (- (a 3 b)))
                (else
                        (and (not (and a b)) c)) ) )
```

Write a procedure call to mystery that will produce the output:

1

-1

#f

hello

Sample Question #3 (Sol.)

There's a lot of ways to do this problem. Here are some solutions.

(mystery – 4 #f) → 1
(mystery + -2 #f) → -1
(mystery word 'h 'ello) → hello
(mystery #f #f #f) → #f

The second condition is impossible to get to and it is only there to throw you off.

Data Abstraction

- Hides implementation from the user
- Provide accessors to retrieve information rather than having the user write it
- Allows the implementation to be changed without changing the use of the data

Case Study

- Differences between version 1 and version 2
- Thought process
- Data abstraction