CS 135 Winter 2018
Tutorial 3: Stepping and Structures
Goals of this tutorial

You should be able to...

- correctly apply **stepping rules**.
- understand and use **structures**.
Review: Stepping Rules

- Always evaluate the **topmost, leftmost** unsimplified expression first.

- **Application of built-in functions:** \((f \, v_1 \ldots \, v_n) \Rightarrow v\) where \(f\) is a built-in function and \(v\) is the value of \(f(v_1, \ldots, v_n)\).

- **Substitution of Constants:** \(id \Rightarrow val\), where \((\text{define id val})\) occurs previously.
Review: Stepping Rules

- Application of user-defined functions:
  
  The general substitution rule is: \((f \ v_1 \ldots \ v_n) \Rightarrow \text{exp}'\)

  where \((\text{define } (f \ x_1 \ldots \ x_n) \ \text{exp})\) occurs previously, and \(\text{exp}'\) is obtained by substituting all occurrences of the formal parameter \(x_i\) replaced by the value \(v_i\) (for \(i\) from 1 to \(n\)) into the expression.
Group Problem - Stepping **mystery-quadratic**

Suppose the following definitions have been fully processed in the Beginning Student language:

```
(define q 13)

(define (mystery-quadratic a b c x)
  (max (− (sqr b) (* 4 a c)) q
       (+ (* a (sqr x)) (* b x) c)))
```

Step through the following:

```
(mystery-quadratic 2 −14 20 3)
```
Clicker Question: Stepping Rules

What are the next three substitution steps for the following code?

\[(\text{define } x \ 7)\]

\[(\text{define } (\text{foo } a \ b))\]

\[\ (\text{+ } a \ b \ x \ (\text{sqr } a))\)]

\[(\text{foo } 3 \ x)\]

A \ \Rightarrow \ (\text{+ } 3 \ 7 \ 7 \ (\text{sqr } 3)) \Rightarrow (\text{+ } 3 \ 7 \ 7 \ 9) \Rightarrow 26

B \ \Rightarrow \ (\text{+ } 3 \ 7 \ x \ (\text{sqr } 3)) \Rightarrow (\text{+ } 3 \ 7 \ 7 \ (\text{sqr } 3)) \Rightarrow (\text{+ } 3 \ 7 \ 7 \ 9)

C \ \Rightarrow \ (\text{foo } 3 \ 7) \Rightarrow (\text{+ } 3 \ 7 \ x \ (\text{sqr } 3)) \Rightarrow (\text{+ } 3 \ 7 \ 7 \ (\text{sqr } 3))

D \ \Rightarrow \ (\text{foo } 3 \ 7) \Rightarrow (\text{+ } 3 \ 7 \ 7 \ (\text{sqr } 3)) \Rightarrow (\text{+ } 3 \ 7 \ 7 \ 9)

E \ \Rightarrow \ (\text{foo } 3 \ 7) \Rightarrow (\text{+ } 3 \ 7 \ x \ (\text{sqr } a)) \Rightarrow (\text{+ } 3 \ 7 \ 7 \ (\text{sqr } a))
Review: Stepping Rules

Simplification Rules for and and or

The simplification rules we use for Boolean expressions involving and and or differ from the ones the Stepper in DrRacket uses.

\[(\text{and} \ \text{false} \ldots) \Rightarrow \text{false}\]

\[(\text{and} \ \text{true} \ldots) \Rightarrow (\text{and} \ldots)\]

\[(\text{and}) \Rightarrow \text{true}\]

\[(\text{or} \ \text{true} \ldots) \Rightarrow \text{true}\]

\[(\text{or} \ \text{false} \ldots) \Rightarrow (\text{or} \ldots)\]

\[(\text{or}) \Rightarrow \text{false}\]
Group Problem - Stepping and

The following definitions have been fully processed in the Beginning Student language:

\[
\begin{align*}
(\text{define } x & \ 2) \\
(\text{define } y & \ (+ \ x \ 4)) \\
(\text{define } z & \ (/ \ y \ 2))
\end{align*}
\]

Step through the following:

\[
(\text{and } (\text{not } (= \ x \ 4)) \ (< \ (/ \ y \ x) \ z))
\]
Review: Stepping Rules
Substitution in cond expressions

There are three rules: when the first expression is \texttt{false}, when it is \texttt{true}, and when it is \texttt{else}.

\[(\text{cond} [\text{false exp}] \ldots) \Rightarrow (\text{cond} \ldots)\]

\[(\text{cond} [\text{true exp}] \ldots) \Rightarrow \text{exp}\]

\[(\text{cond} [\text{else exp}]) \Rightarrow \text{exp}\]

These suffice to simplify any \texttt{cond} expression.

\[(\text{cond})\text{ (with no question/answer pairs) will produce an error. At least one of the questions must be \texttt{true} (if there is no \texttt{else} clause).}\]
Group Problem - Stepping cond

Suppose the following definitions have been fully processed in the Beginning Student language:

(define a 1)
(define b 3)
(define c (+ a b))

Step through the following:

(cond
  [(and (= c 0) (>= b 3)) 'first]
  [(or (even? a) (odd? b)) (* c b 8)]
  [else (/ c a)])
User-defined Structures: uwstudent

Students at UW can be represented with a user-defined structure. Assume that individual grades are integers between 0 and 100, inclusive. A uwstudent is a structure, of type UWStudent with the following fields:

- questid, a non-empty string that is no more than 8 characters long.
- average, a number representing the student’s academic average.
- courses, the number of courses the student has taken. You may assume that each student has taken at least one course.
- coop?, which is true if the student is in co-op, and false otherwise. A student must have an average of at least 60% to be in co-op.

Write a structure definition for uwstudent and a data definition for UWStudent.
User-defined Structures: uwstudent

Once the uwstudent structure has been defined, we get the following constructor and selector functions:

;; Constructor:
(define uwstudent1 (make-uwstudent "t24kwan" 66.5 14 true))
(define uwstudent2 (make-uwstudent "n3hoell" 92.875 32 false))

;; Selectors:
(uwstudent-questid student1) ⇒ "t24kwan"
(uwstudent-average student2) ⇒ 92.875
(uwstudent-courses student1) ⇒ 14
(uwstudent-coop? student2) ⇒ false
User-defined Structures: uwstudent

- An expression such as `(make-uwstudent "t24kwan" 66.5 14 true)` is a **value**, which will not be simplified further by our stepping rules.

- Racket does not enforce data definitions. For example, `(make-uwstudent 'Zainab 159.6 "twenty" 'yes)` will **not** produce an error.

- Defining the uwstudent structure also gives the uwstudent? predicate:

  ```scheme
  ;; uwstudent?: Any → Bool
  (uwstudent? (make-uwstudent "n3hoell" 92.875 32 false)) ⇒ true
  (uwstudent? (make-uwstudent 'Zainab 159.6 "twenty" 'yes)) ⇒ true
  (uwstudent? "Jimmy") ⇒ false
  ```
Group Problem: add-grade

Write a function **add-grade** that consumes a **UWStudent** and a new grade (between 0 and 100 inclusive), and produces the consumed **UWStudent**, with the new grade added to their record. If the student’s new average is less than 60%, they will no longer be in the coop program.

Examples:

```lisp
(add-grade (make-uwstudent "v9huang" 90 19 true) 95)
⇒ (make-uwstudent "v9huang" 90.25 20 true)

(add-grade (make-uwstudent "zkalsuda" 61 24 true) 26)
⇒ (make-uwstudent "zkalsuda" 59.6 25 false)
```
Clicker Question: Nested Structures

(define-struct line (start end))

;; A Line is a (make-line Posn Posn)

(define line1 (make-line (make-posn 2 -8) (make-posn -6 13)))

Using only the selector functions for a line and posn, and the identifier line1, which of the following would produce the value -6?

A (line1 (posn-x end))
B (posn-x (line-end line1))
C (line-end (posn-x line1))
D (posn-x (end line1))
E (line-end-x line1)