Summary of higher order procedures

Rights and privileges of first class citizens

• May be named by variables
• May be passed as arguments to procedures
• May be returned as results of procedures
• May be included as parts of data structures

Procedures in Scheme are + first class citizens!

Repeated power

(define composef
  (lambda (f g)
    (lambda (x)
      (f (g x)))))

rather than:

(define fourth-power (composef square square))
(define eight-power (composef square (composef square square)))

let’s use...

(define fourth-power (repeated square 2))
(define eight-power (repeated square 3))

Recursive repeated

(define (repeated proc n)
  (if (= n 0)
      (lambda (x) x)
      (composef proc (repeated proc (- n 1)))))

(define fourth-power (repeated square 2))
(define eight-power (repeated square 2))

List HOP Common Pattern #1: Transforming a List

(define (square-list lst)
  (if (null? lst)
      nil
      (adjoin (square (first lst))
              (square-list (rest lst)))))

(define (double-list lst)
  (if (null? lst)
      nil
      (adjoin (* 2 (first lst))
              (double-list (rest lst)))))

(define (MAP proc lst)
  (if (null? lst)
      nil
      (adjoin (proc (first lst))
              (MAP proc (rest lst)))))


List HOP Common Pattern #2: Filtering a List

\[ \text{filter even? (list 1 2 3 4 5 6)} \]
\[ ; \text{Value: (2 4 6)} \]

\[ \text{(define (filter pred lst)} \]
\[ \text{ (cond ((null? lst) nil)} \]
\[ \text{ ((pred (first lst)) \}
\[ \text{ (adjoin (first lst)}} \]
\[ \text{ (filter pred (rest lst))))) \]
\[ \text{ (else (filter pred (rest lst)))))} \]

List HOP Common Pattern #3: Accumulating Results

\[ \text{(define (add-up lst)} \]
\[ \text{ (if (null? lst) 0)} \]
\[ \text{ (+ (first lst) \}
\[ \text{ (add-up (rest lst))))) \]
\[ \text{(define (mult-all lst) \}
\[ \text{ (if (null? lst) 1 \}
\[ \text{ (* (first lst) \}
\[ \text{ (mult-all (rest lst))))) \]
\[ \text{(define (fold-right op init lst) \}
\[ \text{ (if (null? lst) init \}
\[ \text{ (op (first lst) \}
\[ \text{ (fold-right op init (rest lst))))) \]

Lists and higher-order procedures

\[ x \to (()) \]
\[ y \to (1 2 3) \]
\[ z \to (1 (2 3) (4)) \]
\[ w \to (1 2 3 4 5) \]

\[ \text{(map (lambda (x) (cons x nil)) y)} \]
\[ \text{((1) (2) (3))} \]
\[ \text{(map inc w)} \]
\[ \text{((2 3 4 5 6))} \]
\[ \text{(filter odd? w)} \]
\[ \text{((1 3 5))} \]
\[ \text{(map inc (filter odd? w))} \]
\[ \text{((2 4 6))} \]
\[ \text{(fold-right op init lst)} \]
\[ \text{((fold-right + 0 lst))} \]

For lists, car has type List<A>->A, cdr has type List<A>->List<A>, and assume nil is an element of type List<A>...

What are the types of these procedures?

1. \text{length List<A>->number} \\
2. \text{list-ref List<A>,number->A} \\
3. \text{map (A->B),List<A>->List<B>} \\
4. \text{filter (A->boolean),List<A>->List<A>} \\
5. \text{fold-right (A,B->B),B,List<A>->B} \\

Uniqueness

Given

\[ \text{(define x (list 1 2 3 1 5 4 3 4))} \]

To select the unique elements from this list,

\[ \text{(unique x)} \]
\[ \text{((1 2 3 5 4))} \]

Start with function remove (e.g. (remove 3 (list 3 4 5)) \( x \) \( (4 5) \))

\[ \text{(define (remove el lst)} \]
\[ \text{ (filter (lambda (x) (not (= x el))) lst))} \]

Uniqueness

\[ \text{(define (unique lst)} \]
\[ \text{ (if (null? lst) nil \}
\[ \text{ (cons (car seq) \}
\[ \text{ (unique (remove (car lst) (cdr lst)))))} \]

### Remember...

- calendar...

<table>
<thead>
<tr>
<th></th>
<th>Orders of growth</th>
<th>Programming</th>
<th>Date allocation</th>
<th>Project 1 due</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>September 17</strong></td>
<td>11 Lecture 4: OOP Programming</td>
<td>19 Exercise 5</td>
<td>29 Exam 4</td>
<td>October 1: Exam 1*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>October 4</strong></td>
<td>15 Lecture 9: Strings and variables</td>
<td>22 Exercise 11</td>
<td>36 Exam 5</td>
<td>October 2: Exam 2*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>October 11</strong></td>
<td>15 Lecture 10: Date and exceptions</td>
<td>22 Exercise 11</td>
<td>36 Exam 5</td>
<td>October 3: Exam 3*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>October 18</strong></td>
<td>15 Lecture 12: Method overriding</td>
<td>22 Exercise 11</td>
<td>36 Exam 5</td>
<td>October 4: Exam 4*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Exams include midterms and finals.