Announcements / Schedule

- Quiz 2 is being held in Walker (50-340) from 7:30-9:30pm
- Old quizzes are generally a good study guide
- 2 pages of notes are allowed (8.5 × 11)
- Everything up to 6 April is fair game
- Key topics: symbols, tagged data, mutation, trees, environment model, procedures with local state, message passing procedures.

Procedures with State

Suppose we want a procedure prev with the property that it returns the previous value it was applied to. For example:

(prev 1) ==> undef
(prev 2) ==> 1
(prev 3) ==> 2

Which of the following attempts to implement prev works correctly? (you probably want to draw an environment diagram for each)

(define prev1
  (let ((x 'undef))
    (lambda (y)
      (let ((z x))
        (set! x y)
        z))))

(define prev2
  (let ((x (list 'undef)))
    (lambda (y)
      (let ((z x))
        (set-car! x y)
        (car z))))))

(define prev3
  (let ((x (cons 'undef 'undef)))
    (lambda (y)
      (set-car! x (cdr x))
      (set-cdr! x y)
      (car x))))))

Suppose we have a correctly implemented version of prev. What is the result of evaluating the expressions below? Assume a fresh evaluation of our prev definition for each.

(let ((foo (prev prev)))
  (((foo +) foo) 1 2))

Answer: ___

(let* ((g (prev prev))
        (f (prev prev))
        ((f +) f) 1 2))

Answer: ___

Environmental Matching

Match each of the following code blocks to one of the environment models. Assume that the topmost frame is the global environment.

A

B

C

D

None

(define f (lambda () ...))
(let ()
  (f f))

Environment diagram: ___

(define f
  (let ()
    (lambda (f) ...)))
(f f)

Environment diagram: ___

(define f
  (let ()
    (let ((f (lambda () ...))) f)))

Environment diagram: ___

(define f
  (let ()
    (define f f))
  ...))
(f)

Environment diagram: ___

(define f
  (let ()
    (lambda () ...)))

Environment diagram: ___

(let ((x (prev prev)))
  (((prev +) prev) 1 2))

Answer: ___
Wiring
Consider the following expressions:

```scheme
(define (buffer init)
  (let ((prev init))
    (lambda (x)
      (let ((old prev))
        (set! prev x)
        old)))))
(define b (buffer 10))
(b 20)
```

Here are the parts of the environment diagram that results from evaluating these expressions:

```
GE  b  buffer
P1  p: (old)  b: (set! ...)...
P2  p: (x)    b: (let ((old...))...)
P3  p: (init) b: (let ((prev...))...)
P4  p: (prev) b: (lambda (x)...)
```

For each of the following frames, indicate the lowest frame of the enclosing environment, choosing one of GE, E1, E2, E3, E4, none, or not shown.

<table>
<thead>
<tr>
<th>Frame</th>
<th>Enclosing Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>GE</td>
<td></td>
</tr>
<tr>
<td>E1</td>
<td></td>
</tr>
<tr>
<td>E2</td>
<td></td>
</tr>
<tr>
<td>E3</td>
<td></td>
</tr>
<tr>
<td>E4</td>
<td></td>
</tr>
</tbody>
</table>

For each of the following procedure objects, indicate the lowest frame of the enclosing environment, choosing one of GE, E1, E2, E3, E4, none, or not shown.

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Enclosing Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td></td>
</tr>
</tbody>
</table>

For each of the following variable names, indicate the value to which it is bound in the specified environment at the end of the evaluation of the expressions. Indicate the value by choosing one of GE, E1, E2, E3, E4, E5 or P1, P2, P3 or a symbol, a number, a list of numbers, or a boolean value.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Environment</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>GE</td>
<td></td>
</tr>
<tr>
<td>buffer</td>
<td>GE</td>
<td></td>
</tr>
<tr>
<td>prev</td>
<td>E1</td>
<td></td>
</tr>
<tr>
<td>old</td>
<td>E2</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>E3</td>
<td></td>
</tr>
<tr>
<td>init</td>
<td>E4</td>
<td></td>
</tr>
</tbody>
</table>
Solutions

Announcements / Schedule

- Quiz 2 is being held in Walker (50-340) from 7:30-9:30pm
- Old quizzes are generally a good study guide (http://web.mit.edu/amdragon/www/6.001-quizzes/)
- 2 pages of notes are allowed (8.5 \times 11).
- Everything up to 6 April is fair game
- Key topics: symbols, tagged data, mutation, trees, environment model, procedures with local state, message passing procedures.

Procedures with State

Suppose we want a procedure \texttt{prev} with the property that it returns the previous value it was applied to. For example:

\begin{verbatim}
(prev 1) => undef
(prev 2) => 1
(prev 3) => 2
\end{verbatim}

Which of the following attempts to implement \texttt{prev} works correctly? (you probably want to draw an environment diagram for each)

\begin{itemize}
  \item \texttt{prev1} and \texttt{prev3} work as desired. \texttt{prev2} doesn’t work because the \texttt{set-car}! affects both \texttt{x} and \texttt{z}.
  \item \texttt{(define prev1}
    \begin{verbatim}
    (let ((x 'undef))
      (lambda (y)
        (let ((z x))
          (set! x y)
          z)))
    \end{verbatim}
  \end{itemize}

\begin{itemize}
  \item \texttt{(define prev2}
    \begin{verbatim}
    (let ((x (list 'undef)))
      (lambda (y)
        (let ((z x))
          (set-car! x y)
          (car z))))
    \end{verbatim}
  \end{itemize}

\begin{itemize}
  \item \texttt{(define prev3}
    \begin{verbatim}
    (let ((x (cons 'undef 'undef)))
      (lambda (y)
        (set-car! x (cdr x))
        (set-cdr! x y)
        (car x))))
    \end{verbatim}
  \end{itemize}

Suppose we have a correctly implemented version of \texttt{prev}. What is the result of evaluating the expressions below? Assume a fresh evaluation of our \texttt{prev} definition for each.

\begin{verbatim}
(let* ((g (prev prev))
       (f (prev prev)))
  (((f +) f) 1 2))
\end{verbatim}

Answer: \texttt{error}

\begin{verbatim}
(let* ((g (prev prev))
       (f (prev prev)))
  (((f +) f) 1 2))
\end{verbatim}

Answer: \texttt{3}

Environmental Matching

Match each of the following code blocks to one of the environment models. Assume that the topmost frame is the global environment.

A \hspace{1cm} B \hspace{1cm} C \hspace{1cm} D

\begin{itemize}
  \item \texttt{(define f (lambda () ...))}\n    \begin{verbatim}
    (let () (lambda (f) ...))
    (f f)
    \end{verbatim}
  \end{itemize}

Environment diagram: \texttt{A}

\begin{itemize}
  \item \texttt{(define f}
    \begin{verbatim}
    (let () (lambda (f) ...))
    (f f)
    \end{verbatim}
  \end{itemize}

Environment diagram: \texttt{C}

\begin{itemize}
  \item \texttt{(define f}
    \begin{verbatim}
    (let () (lambda (f) ...))
    (f f)
    \end{verbatim}
  \end{itemize}

Environment diagram: \texttt{C}

\begin{itemize}
  \item \texttt{(define f}
    \begin{verbatim}
    (let () (lambda () ...
      (define f f)...
    )
    \end{verbatim}
  \end{itemize}

Environment diagram: \texttt{C}

\begin{itemize}
  \item \texttt{(define f}
    \begin{verbatim}
    (let () (lambda () ...
      (lambda () ...
    )
    \end{verbatim}
  \end{itemize}

Environment diagram: \texttt{B}

\begin{itemize}
  \item \texttt{(define f}
    \begin{verbatim}
    (let () (lambda () ...
      (lambda () ...
    )
    \end{verbatim}
  \end{itemize}

Environment diagram: \texttt{B}
Wiring
Consider the following expressions:

\[
\text{define (buffer init)} \\
\text{let ((prev init))} \\
\text{(lambda (x)} \\
\text{(let ((old prev))} \\
\text{(set! prev x)} \\
\text{old))})\
\text{(define b (buffer 10))} \\
\text{(b 20)}
\]

Here are the parts of the environment diagram that results from evaluating these expressions:

<table>
<thead>
<tr>
<th>Frame</th>
<th>Enclosing Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>GE</td>
<td>none</td>
</tr>
<tr>
<td>E1</td>
<td>E4</td>
</tr>
<tr>
<td>E2</td>
<td>E3</td>
</tr>
<tr>
<td>E3</td>
<td>E1</td>
</tr>
<tr>
<td>E4</td>
<td>GE</td>
</tr>
</tbody>
</table>

For each of the following procedure objects, indicate the lowest frame of the enclosing environment, choosing one of GE, E1, E2, E3, E4, none, or not shown.

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Enclosing Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>E3</td>
</tr>
<tr>
<td>P2</td>
<td>E1</td>
</tr>
<tr>
<td>P3</td>
<td>GE</td>
</tr>
<tr>
<td>P4</td>
<td>E4</td>
</tr>
</tbody>
</table>

For each of the following variable names, indicate the value to which it is bound in the specified environment at the end of the evaluation of the expressions. Indicate the value by choosing one of GE, E1, E2, E3, E4, or a symbol, a number, a list of numbers, or a boolean value.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Environment</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>GE</td>
<td>P2</td>
</tr>
<tr>
<td>buffer</td>
<td>GE</td>
<td>P3</td>
</tr>
<tr>
<td>prev</td>
<td>E1</td>
<td>20</td>
</tr>
<tr>
<td>old</td>
<td>E2</td>
<td>10</td>
</tr>
<tr>
<td>x</td>
<td>E3</td>
<td>20</td>
</tr>
<tr>
<td>init</td>
<td>E4</td>
<td>10</td>
</tr>
</tbody>
</table>