Instructions: You have eighty minutes to complete this open-book, open-note, closed-interpreter exam. Please write all answers in the provided space, plus the back of the exam if necessary.

Note on actual exam: The exam may refer to the env.rkt, lambda.rkt, and store-with.rkt interpreters. If you need the interpreters for reference to answer the questions, please bring a copy (paper or electronic) with you.

1) [15 pts] Given the following grammar:

\[
\langle \text{weed} \rangle = \text{leaf} \\
| \ (\text{branch} \ (\text{weed}) \ (\text{weed})) \\
| \ (\text{stem} \ (\text{weed}))
\]

Provide a define-type declaration for Weed that is a suitable representation for (weed)s.
2) [25 pts] Implement the function \texttt{weed-forks}, takes a \texttt{Weed} and returns the number of branches that it contains. Your implementation must follow the shape of the data definition, and \textbf{it must include tests}. 
3) [20 pts] For each of the following expressions, show the store that would be returned with the program’s value when using the store-with.rkt interpreter. Instead of nested override-stores, you can show the store as a list of cells. Recall that locations are allocated starting at 1.

a) \{box \{+ 1 2\}\}

b) \{let \{[[b \{box \{+ 1 2\}\}]]\}
   \{begin
       \{set-box! b 4\}
       \{box 5\}\}\}

c) \{let \{[[f \{lambda \{x\}
      \{box x\}\}]]\}
   \{set-box! \{f 0\} \{f 1\}\}\}

d) \{let \{[[f \{lambda \{x\}
      \{box x\}\}]]\}
   \{let \{[[b \{f 0\}]]\}
   \{set-box! b b\}\}\}
4) [40 pts] The following expression is evaluated using the \texttt{lambda.rkt} interpreter:

\[
\begin{align*}
\text{let } & \{\{g \text{ lambda } \{x\} \text{ lambda } \{y\} \{+ y x\}\}\}\} \\
\text{let } & \{[x 13]\} \\
\text{let } & \{[f \{g 6\}\]} \\
\text{f x}\}
\end{align*}
\]

(Note: the actual exam will also use \texttt{lambda.rkt}.) Describe a trace of the evaluation in terms of arguments to an \texttt{interp} function for every call. (There will be 15 calls.) The \texttt{interp} function takes two arguments — an expression and an environment — so show both for each call. Put each call to \texttt{interp} and in a rectangle, and show recursive calls within the same rectangle by using nested rectangles; show the \texttt{interp} result at the end of each rectangle. Number the calls to \texttt{interp} to reflect the actual order of the calls when running the interpreter.

You can omit the rectangle around the first call and everything else, leaving it implicit. If you can’t get all of a box’s content on one page, you can write a placeholder for a nested box and write the actual box separately (possibly on a separate page). It’s ok to leave out calls to \texttt{parse} and just quote concrete syntax. Use the following abbreviations to save time and space:

\[
\begin{align*}
X_0 & = \text{the whole expression, quoted} \\
X_1 & = \text{`\{lambda } x\text{ } \text{ lambda } y \text{ } \{+ y x\}\}'} \\
X_2 & = \text{`\{let } \{x 13\} \text{ } \text{ let } \{[f \{g 6\}\]} \text{ } \{f x\}\}'} \\
X_3 & = \text{`\{let } \{[f \{g 6\}\]} \text{ } \{f x\}\}'}
\end{align*}
\]
Answers

1) [15 pts]

\[
\text{(define-type Weed}\n\text{(leaf)}\n\text{(stem [rest : Weed])}\n\text{(branch [left : Weed]\n\text{[right : Weed]])})\n\]

2) [25 pts]

\[
\text{(define (weed-forks [w : Weed]) : Number}\n\text{(type-case Weed w}\n\text{[(leaf) 0]}\n\text{[(stem rest) (weed-forks rest)]}\n\text{[(branch l r) (+ 1}\n\text{(+ (weed-forks l)}\n\text{(weed-forks r)))]}}\n\]

(test (weed-forks (leaf)) 0)
(test (weed-forks (stem (leaf))) 0)
(test (weed-forks (stem (branch (leaf) (leaf)))) 1)
(test (weed-forks (branch (branch (leaf) (leaf)) (leaf))) 2)

3) [20 pts]

a) (list (cell 1 (numV 3)))
b) (list (cell 2 (numV 5)) (cell 1 (numV 4)) (cell 1 (numV 3)))
c) (list (cell 1 (boxV 2)) (cell 2 (numV 1)) (cell 1 (numV 0)))
d) (list (cell 1 (boxV 1)) (cell 1 (numV 0)))

4) [40 pts]
1 (interp X₀
   mt-env)

2 (interp X₁
   mt-env)
result is \( C₁ = \text{closV} 'x \{\lambda y \{+ y x\}\} \) mt-env

3 (interp X₂
   \( E₁ = \text{extend-env} (\text{bind} 'g \ C₁) \) mt-env)

4 (interp '₁₃
   \( E₁ \))
result is (numV 13)

5 (interp X₁
   \( E₂ = \text{extend-env} (\text{bind} 'x \ (\text{numV} 13)) E₁)\)

6 (interp `g ₆
   \( E₂ \))

7 (interp `g
   \( E₂ \))
result is C₁

8 (interp `₆
   \( E₂ \))
result is (numV 6)

9 (interp `\{\lambda y \{+ y x\}\}
   \( E₃ = \text{extend-env} (\text{bind} 'x \ (\text{numV} 6)) \) mt-env)
result is \( C₂ = \text{closV} 'y \{+ y x\} E₃ \)

10 (interp `\{f x\}
    \( E₄ = \text{extend-env} (\text{bind} 'f \ C₂) E₂)\)

11 (interp `f
    \( E₄ \))
result is C₂

12 (interp `x
    \( E₄ \))
result is (numV 13)

13 (interp `\{+ y x\}
    \( E₃ = \text{extend-env} (\text{bind} 'y \ (\text{numV} 13)) E₃\))

14 (interp `y
    \( E₃ \))
result is (numV 13)

15 (interp `x
    \( E₃ \))
result is (numV 6)
result is (numV 19)
result is (numV 19)
result is (numV 19)
result is (numV 19)
result is (numV 19)
An alternate form of the same solution that fits more easily on multiple pages:

1. \((\text{interp } X_0 \text{ mt-env})\)

2. \((\text{interp } X_1 \text{ mt-env})\)
   
   result is \(C_1 = (\text{closV } 'x \cdot \{\lambda y \{ + y \cdot x\}\} \text{ mt-env})\)

3. \((\text{interp } X_2 \text{ mt-env})\)

   \[E_1 = (\text{extend-env } (\text{bind } 'g \ C_1) \text{ mt-env})\]

4. \((\text{interp } '13 \text{ E}_1)\)
   
   result is \((\text{numV } 13)\)

5. \((\text{interp } X_3 \text{ mt-env})\)

   \[E_2 = (\text{extend-env } (\text{bind } 'x \ (\text{numV } 13)) \ E_1)\]

   BOX A

   BOX B

   result is \((\text{numV } 19)\)

   result is \((\text{numV } 19)\)

   result is \((\text{numV } 19)\)

   BOX A =

   6. \((\text{interp } '{g \ 6} \text{ E}_2)\)

   7. \((\text{interp } 'g \text{ E}_2)\)

   result is \(C_1\)

   8. \((\text{interp } '6 \text{ E}_2)\)

   result is \((\text{numV } 6)\)

   9. \((\text{interp } '{\lambda y \{ + y \cdot x\}} \text{ E}_3 = (\text{extend-env } (\text{bind } 'x \ (\text{numV } 6)) \ E_2)\)

   result is \(C_2 = (\text{closV } 'y \cdot ('y \ E_3)\)

   result is \(C_2\)

   BOX B =

   10. \((\text{interp } '{f \ x} \text{ E}_4 = (\text{extend-env } (\text{bind } 'f \ C_2) \ E_2))\)

   11. \((\text{interp } 'f \text{ E}_4)\)

   result is \(C_2\)

   12. \((\text{interp } 'x \text{ E}_4)\)

   result is \((\text{numV } 13)\)

   13. \((\text{interp } '{+ \ y \ x} \text{ E}_5 = (\text{extend-env } (\text{bind } 'y \ (\text{numV } 13)) \ E_3))\)

   14. \((\text{interp } 'y \text{ E}_5)\)

   result is \((\text{numV } 13)\)

   15. \((\text{interp } 'x \text{ E}_5)\)

   result is \((\text{numV } 6)\)

   result is \((\text{numV } 19)\)

   result is \((\text{numV } 19)\)