PA5 - Environment Diagrams (Paper Assignment)

Due: Monday, Oct 27 at 11:59 PM

Overview

This assignment focuses on understanding environments, variable scoping, and the relationship between `let` expressions and procedure applications. Unlike previous assignments, **PA5 is completed on paper** (or drawn digitally) and submitted as a single PDF document.

The V5 Language

All questions reference the **V5 language** from PLCC, which includes:

- Basic arithmetic operations: `+`, `-`, `*`, `/`
- Let expressions: `let x = <exp> in <body>`
- Procedures: `proc(<params>) <body>`
- Procedure application: `.proc(<args>)`
- Recursive procedures: `letrec f = proc(<params>) <body> in <exp>`
- Conditionals: `if <test> then <consequent> else <alternative>`
- Built-in operations: `add1`, `sub1`, `zero?`

You can reference the V5 grammar and examples at:
`https://github.com/ourPLCC/languages/blob/main/src/V5`

Part 1: Drawing Diagrams (Questions 1-5)

For questions 1-5, **draw environment diagrams** showing all environments created during evaluation of the given expressions. Start with an empty initial environment.

Important Reminders

- **Only `let` expressions and `proc` applications create new environments**
- Primitive operations (like `+`, `sub1`) do NOT create new environments
- Show all bindings in each environment
- Draw arrows to parent environments to show environment chaining
- For procedure values, show the captured environment (closure)
- For recursive procedures (`letrec`), show the circular environment reference

---

Question 1

Draw a diagram of all environments created during evaluation of the following expression. Assume the initial environment is empty.

```
let
  x = 3
  y = 5
  z = 8
in
  +(x,+(y,z))
```

Question 2

Draw a diagram of all environments created during evaluation of the following expression.

```
let
  x = 3
in
  let
    y = 5
  in
    let
      z = 8
    in
      +(x,+(y,z))
```

Hint: This creates a different environment structure than Question 1, even though the result is the same.

Question 3

Draw a diagram of all environments created during evaluation of the following expression.

```
let
  x = 3
in
  let
    p = proc(t) +(t,x)
  in
    .p(5)
```

Important: Be sure to show:

- The environment captured by the procedure `p` (closure)
- The new environment created when `.p(5)` is applied

Question 4

Draw a diagram of all environments created during evaluation of the following expression.

```
let
  t = 3
in
  let
    f = let
          x = t
        in
          proc(t) +(t,x)
  in
    .f(5)
```

Important: Notice that the inner `let` evaluates to a `ProcVal`. Show:

- The environment where `x` is bound
- The environment captured by the procedure
- The environment created when `.f(5)` is applied

Question 5

Draw a diagram of all environments created during evaluation of the following expression.

```
letrec
  sumi = proc(x) {
    if x
    then +(x, .sumi(sub1(x)))
    else 0
  }
in
  .sumi(1)
```

Important: Show all environments created by procedure applications, including:

- The recursive environment created by `letrec`
- The environment for `.sumi(1)`
- The environment for the recursive call `.sumi(0)`
- How `sumi` refers back to itself (circular reference)

Hint: This will create more environments than you might expect. The recursive call creates a new environment even though it returns 0 without making further calls.

Part 2: Let-to-Procedure Transformation (Questions 6-7)

The V-languages demonstrate that `let` expressions are syntactic sugar for procedure applications. Specifically:

```
let x = <exp> in <body>
```

is equivalent to:

```
.proc(x) <body>(<exp>)
```

For questions 6-7, **transform the given expressions** by replacing all `let` expressions with procedure applications using the algorithm above. Make no other changes to the expressions.

Verification

You are **strongly encouraged** to test your transformations using the V4 REPL:

1. Evaluate the original expression and note the result
2. Evaluate your transformed expression
3. Verify both produce the same result

Question 6

Replace all `let` expressions in the following with procedure applications. Your answer should contain **no `let` keywords**.

```
let
  x = 3
  y = 5
  z = 8
in
  +(x,+(y,z))
```

Hint: Multiple bindings in a single `let` can be handled by nested procedure applications.

Question 7

Replace all `let` expressions in the following with procedure applications. Your answer should contain **no `let` keywords**.

```
let
  x = 3
in
  let
    y = 5
  in
    +(x,+(y,2))
```

Hint: Work from the inside out. Transform the innermost `let` first, then work outward.

Submission

To Submit:

1. Complete all 7 questions on paper or using a digital drawing tool
2. If drawing by hand, use clear, dark lines (consider tracing with a marker for legibility)
3. Ensure all text is readable and environments/bindings are clearly labeled

Submission Options (choose one):

- **Option A - Digital Submission**: Convert your work to a **single PDF file** named `pa5-YOURNAME.pdf` and upload to Kodiak by Monday, Oct 27 at 11:59 PM
- **Option B - Physical Submission**: Hand in a physical copy in class on Tuesday, Oct 27 (Midterm day)

Drawing Guidelines:

- Label all environments, bindings, and values clearly
- Draw arrows to show parent environment links
- Include question numbers with each answer
- For closures, clearly indicate the captured environment

Grading Criteria:

- **Submission (33.3%):** Environment diagrams show correct structure, all bindings, and proper parent links
- **Completeness (33.3%):** Let-to-proc transformations are correct with no `let` keywords remaining
- **Correctness (33.3%):** Legible diagrams, clear labels, organized layout, proper question numbering

Late Policy: 10% per day, maximum 5 days late

Course content developed by Declan Gray-Mullen for WNEU with Claude