Example Breakdown: Lambda Calculus Reductions

Problem 1: Reducing (λx. x x) (λy. y)

Step 1: Understanding the Expression

- We have a function **λx. x x** applied to another function **λy. y**.
- In lambda calculus, function application means substituting the argument into the function.

Step 2: Applying Beta Reduction

- The function **λx. x x** expects an input and applies it to itself.
- We substitute **(λy. y)** for **x** in **x x**:

(λx. x x) (λy. y) → (λy. y) (λy. y)

Step 3: Evaluating the New Expression

- Now we have **(λy. y) (λy. y)**.
- The function **λy. y** is the identity function, meaning it returns whatever is given to it.
- Applying it to **λy. y** results in:

(λy. y) → λy. y

- The expression reduces to **λy. y**, which is the simplest form (normal form).

Key Concepts in This Reduction:

- **Beta Reduction:** Replacing a bound variable with its argument.
- **Identity Function: λy. y** always returns its input.
- **Normal Form:** When no further reductions can be applied.

Problem 2: Reducing (λx. x (λz. z)) (λy. y y)

Step 1: Understanding the Expression

- The function **λx. x (λz. z)** is applied to **λy. y y**.
- This means **x** will be replaced with **(λy. y y)** in the expression **x (λz. z)**.

Step 2: Applying Beta Reduction

- Substitute **(λy. y y)** for **x** in **x (λz. z)**:

(λx. x (λz. z)) (λy. y y) → (λy. y y) (λz. z)

Step 3: Evaluating the New Expression

- Now we have **(λy. y y) (λz. z)**.
- **λy. y y** applies itself to its argument, which is **λz. z**.
- This results in:

(λz. z) (λz. z)

- The function **λz. z** is the identity function, so it just returns **λz. z**.
- The final result is **λz. z**, which is the normal form.

Key Concepts in This Reduction:

- **Function Substitution:** Replacing **x** with its argument.
- **Self-Application: λy. y y** applies itself to an argument.
- **Normal Form:** The simplest form after reduction.

General Notes for Understanding Lambda Calculus:

- **Alpha Conversion:** Renaming bound variables to avoid confusion.
- **Beta Reduction:** The main step of computation in lambda calculus.
- **Eta Reduction:** Simplifying functions when possible (e.g., **λx. f x → f** if **x** does not appear in **f**).

This document should help you practice breaking down lambda expressions step by step!