CMSC330 - Organization of Programming Languages
Fall 2024 - Exam 2

CMSC330 Course Staff
University of Maryland
Department of Computer Science

Name: _____

UID: _____

I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination

Signature: _____

Ground Rules

- Please write legibly. **If we cannot read your answer you will not receive credit.**
- You may use anything on the accompanying reference sheet anywhere on this exam
- Please remove the reference sheet from the exam
- The back of the reference sheet has some scratch space on it. If you use it, you must turn in your scratch work
- You may not leave the room or hand in your exam within the last 10 minutes of the exam
- If anything is unclear, ask a proctor. If you are still confused, write down your assumptions in the margin

Question	Points
P1.	10
P2.	6
P3	8
P4.	4
P5.	18
P6.	16
P7.	20
P8.	18
Total	100

Problem 1: Language Concepts

[Total 10 pts]

	True	False
CFGs and regular expressions can be used interchangably.	T	**F**
An expression's grammatical correctness is checked in both the parsing and evaluating phase.	T	**F**
A turing complete language can simulate any turing machine.	**T**	F
A language that uses dynamic typing will raise any type errors during compile time, not runtime	T	**F**
The type of a variable in a dynamically typed language is determined at run-time, when the variable is last assigned.	**T**	F
$\{a : \{a : Int, b : Int\}, b : Int\}$ is a subtype of $\{a : \{a : Int, b : Int\}\}$	**T**	F
Inference rules can be used to specify whether a program is well typed.	**T**	F
A lambda calculus term is in beta normal form if it cannot be reduced any further using beta reduction.	**T**	F
Lambda calculus is not Turing-complete.	T	**F**
A type-safe language is one in which for every program, well-defined $\longrightarrow$ well-typed.	T	**F**
There exist regular expressions that cannot be represented by Context Free Grammars.	T	**F**
An expression's grammatical correctness is checked in both the lexing and parsing phase.	T	**F**
Turing complete languages can be used to describe any mathematical expression.	T	**F**
A language that uses dynamic typing will raise any type errors during runtime, not compile time	**T**	F
If an Operational semantics proof cannot be completed, it is still possible for the program to be correct.	T	**F**
If B and C are subtypes of A then B must also be a subtype of C	T	**F**
The typing rules of a language can be determined by the language's Operational semantics rules alone.	T	**F**
The first step of a lambda calculus evaluation will always be distinct between eager and lazy evaluation methods.	T	**F**
When talking about type systems, soundness and completeness can be used interchangeably.	T	**F**
All regular expressions can be represented by Context Free Grammars.	**T**	F
A turing complete language can solve all problems given unlimited memory .	T	**F**

	True	False
If B and C are subtypes of A and A is a subtype of D, then C and B must also be a subtype of D.	T	F
The typing rules of a language cannot be determined by the language's Operational semantics rules alone.	T	F
When talking about type systems, soundness and completeness cannot be used interchangeably.	T	F
A type system can be sound, complete, and decidable.	T	F

Problem 2: Context Free Grammars - Acceptance

[Total 6 pts]

Which of the following strings can be derived using CFG below? Select all that apply.

$$S \rightarrow \quad greater?\ M\ N \mid N\ ?\ M\ :\ M \mid M$$
$$M \rightarrow \quad add\ M\ M \mid sub\ M\ M \mid N$$
$$N \rightarrow \quad n \mid b \mid (S)$$

Note: $n \in \mathbb{Z}, b \in \{true, false\}$

- (A) `(sub (greater? true 2) 3)`
- (B) `()`
- (C) `greater? 2 3`
- (D) `(((add 6 7)))`
- (E) `(add 1 2 ? (sub 3 4) : 8)`
- (F) `(true 2)`

$$S \rightarrow \quad equal?\ M\ N \mid N\ ?\ M\ :\ M \mid M$$
$$M \rightarrow \quad mult\ M\ M \mid div\ M\ M \mid N$$
$$N \rightarrow \quad n \mid b \mid (S)$$

Note: $n \in \mathbb{Z}, b \in \{true, false\}$

- (A) `(div (equal? true 2) 3)`
- (B) `equals? true false`
- (C) `(1 8)`
- (D) `(((mult 6 7)))`
- (E) `(true? (mult 2 (div 3 4)) : 8)`
- (F) `()`

$$S \rightarrow \quad equal?\ M\ N \mid N\ ?\ M\ :\ M \mid M$$
$$M \rightarrow \quad add\ M\ M \mid sub\ M\ M \mid N$$
$$N \rightarrow \quad n \mid b \mid (S)$$

Note: $n \in \mathbb{Z}, b \in \{true, false\}$

- (A) `(sub (equal? true 2) 3)`
- (B) `equals? true false`
- (C) `(1 8)`
- (D) `(((add 6 7)))`
- (E) `(true ? (add 2 (sub 3 4)) : 8)`
- (F) `()`

$$S \rightarrow \quad equal?\ M\ N \mid N\ ?\ M\ :\ M \mid M$$
$$M \rightarrow \quad mult\ M\ M \mid div\ M\ M \mid N$$
$$N \rightarrow \quad n \mid b \mid (S)$$

Note: $n \in \mathbb{Z}, b \in \{true, false\}$

- (A) `(div (equals? true 2) 3)`
- (B) `equal? true false`
- (C) `(((mult 6 7)))`
- (D) `((1 8))`
- (E) `(true ? (div 2 (mult 3 4)) : 8)`
- (F) `()`

Problem 3: Context Free Grammars - Derivations

[Total 8 pts]

$$
\begin{aligned}
S \rightarrow &\quad greater?\ M\ N \mid N\ ?\ M\ :\ M \mid M \\
M \rightarrow &\quad add\ M\ M \mid sub\ M\ M \mid N \\
N \rightarrow &\quad n \mid b \mid (S) \\
&\textbf{Note: } n \in \mathbb{Z}, b \in \{true, false\}
\end{aligned}
$$

Using only **left-most** derivation, and the above grammar, derive the string "true ? 4 : (add 2 1)" **(do not draw a tree)**.

S $\rightarrow$
$N\ ?\ M : M \rightarrow$
$true\ ?\ M : M \rightarrow$
$true\ ?\ N : M \rightarrow$
$true\ ?\ 4 : M \rightarrow$
$true\ ?\ 4 : N \rightarrow$
$true\ ?\ 4 : (S) \rightarrow$
$true\ ?\ 4 : (M) \rightarrow$
$true\ ?\ 4 : (add\ M\ M) \rightarrow$
$true\ ?\ 4 : (add\ N\ M) \rightarrow$
$true\ ?\ 4 : (add\ 2\ M) \rightarrow$
$true\ ?\ 4 : (add\ 2\ N) \rightarrow$
$true\ ?\ 4 : (add\ 2\ 1)$

$$
\begin{aligned}
S \rightarrow &\quad equal?\ M\ N \mid N\ ?\ M\ :\ M \mid M \\
M \rightarrow &\quad mult\ M\ M \mid div\ M\ M \mid N \\
N \rightarrow &\quad n \mid b \mid (S) \\
&\textbf{Note: } n \in \mathbb{Z}, b \in \{true, false\}
\end{aligned}
$$

Using only a **left-most** derivation, and the above grammar, derive the string "true ? (div 3 1) : 4" **(do not draw a tree)**.

S $\rightarrow$
$N\ ?\ M : M \rightarrow$
$true\ ?\ M : M \rightarrow$
$true\ ?\ N : M \rightarrow$
$true\ ?\ (S) : M \rightarrow$
$true\ ?\ (M) : M \rightarrow$
$true\ ?\ (divMM) : M \rightarrow$
$true\ ?\ (divNM) : M \rightarrow$
$true\ ?\ (div3M) : M \rightarrow$
$true\ ?\ (div3N) : M \rightarrow$
$true\ ?\ (div31) : M \rightarrow$
$true\ ?\ (div31) : N \rightarrow$
$true\ ?\ (div31) : 4$

$$
\begin{aligned}
S \rightarrow &\quad equal?\ M\ N \mid N\ ?\ M\ :\ M \mid M \\
M \rightarrow &\quad add\ M\ M \mid sub\ M\ M \mid N \\
N \rightarrow &\quad n \mid b \mid (S) \\
&\textbf{Note: } n \in \mathbb{Z}, b \in \{true, false\}
\end{aligned}
$$

Using only a **left-most** derivation, and the above grammar, derive the string "true ? (add 3 1) : 4" **(do not draw a tree)**.

S $\rightarrow$
$N\ ?\ M : M \rightarrow$
$true\ ?\ M : M \rightarrow$
$true\ ?\ N : M \rightarrow$
$true\ ?\ (S) : M \rightarrow$
$true\ ?\ (M) : M \rightarrow$
$true\ ?\ (addMM) : M \rightarrow$
$true\ ?\ (addNM) : M \rightarrow$
$true\ ?\ (add3M) : M \rightarrow$
$true\ ?\ (add3N) : M \rightarrow$
$true\ ?\ (add31) : M \rightarrow$
$true\ ?\ (add31) : N \rightarrow$
$true\ ?\ (add31) : 4$

$$
\begin{aligned}
S \rightarrow &\quad equal?\ M\ N \mid N\ ?\ M\ :\ M \mid M \\
M \rightarrow &\quad mult\ M\ M \mid div\ M\ M \mid N \\
N \rightarrow &\quad n \mid b \mid (S) \\
&\textbf{Note: } n \in \mathbb{Z}, b \in \{true, false\}
\end{aligned}
$$

Using only a **left-most** derivation, and the above grammar, derive the string "equal? (mult 2 4) 8" **(do not draw a tree)**.

S $\rightarrow$
$equal\ ?\ M\ N \rightarrow$
$equal\ ?\ N\ N \rightarrow$
$equal\ ?\ (S)\ N \rightarrow$
$equal\ ?\ (M)\ N \rightarrow$
$equal\ ?\ (mult\ M\ M)\ M \rightarrow$
$equal\ ?\ (mult\ N\ M)\ M \rightarrow$
$equal\ ?\ (mult\ 2\ M)\ M \rightarrow$
$equal\ ?\ (mult\ 2\ N)\ M \rightarrow$
$equal\ ?\ (mult\ 2\ 4)\ M \rightarrow$
$equal\ ?\ (mult\ 2\ 4)\ N \rightarrow$
$equal\ ?\ (mult\ 2\ 4)\ 8$

Problem 4: Context Free Grammars - Creation

[Total 4 pts]

Design a Context Free Grammar using the alphabet {x, y, z}.

- Accepted strings must be of length 0 or more
- Accepted strings must have an equal count of both 'x's and 'z's, with an even number of 'y's allowed in between 'x's and 'z's.
- The above rule can also be represented as: $x^a y^b z^a$ where a is a whole number and b is an **even** whole number.
- Examples of accepted strings: "xyyz", "xxzz", "yy", "xxxyyzzz", ""
- Note: Whole numbers are all positive integers including 0.

(A) $S \rightarrow xxYzz \mid \epsilon$
$Y \rightarrow yY \mid \epsilon$

(B) $S \rightarrow xSz \mid Y \mid \epsilon$
$Y \rightarrow y \mid \epsilon$

(C) $S \rightarrow xSz \mid Y \mid \epsilon$
$Y \rightarrow yyY \mid \epsilon$

(D) $S \rightarrow xSz \mid Y$
$Y \rightarrow yY \mid \epsilon$

Design a Context Free Grammar using the alphabet {x,y,z}.

- Accepted strings must be of length 0 or more
- Accepted strings must have an equal, even count of both 'x's and 'z's, with any number of 'y's allowed in between 'x's and 'z's.
- The above rule can also be represented as: $x^a y^b z^a$ where a is an **even** whole number and b is a whole number.
- Examples of accepted strings: "xxyyzz", "xxzz", "y", "xxxxyyyzzzz", ""
- Note: Whole numbers are all positive integers including 0.

(A) $S \rightarrow xSz|\epsilon$
$Y \rightarrow y|\epsilon$

(B) $S \rightarrow xxSzz|Y|\epsilon$
$Y \rightarrow yY|\epsilon$

(C) $S \rightarrow xxYzz|\epsilon$
$Y \rightarrow yY|\epsilon$

(D) $S \rightarrow xxSzz|Y$
$Y \rightarrow yY|\epsilon$

Design a Context Free Grammar using the alphabet {a,b,c}.

- Accepted strings must be of length 0 or more
- Accepted strings must have an equal, even count of both 'a's and 'c's, with any number of 'b's allowed in between 'a's and 'c's.
- The above rule can also be represented as: $a^x b^y c^x$ where x is an **even** whole number and y is a whole number.
- Examples of accepted strings: "aabbcc", "aacc", "b", "aaaabbbcccc", ""
- Note: Whole numbers are all positive integers including 0.

(A) $S \rightarrow aSc|\epsilon$
$B \rightarrow b|\epsilon$

(B) $S \rightarrow aaScc|B$
$B \rightarrow bB|\epsilon$

(C) $S \rightarrow aaBcc|\epsilon$
$B \rightarrow bB|\epsilon$

(D) $S \rightarrow aaScc|B|\epsilon$
$B \rightarrow bB|\epsilon$

Design a Context Free Grammar using the alphabet $\{a,b,c\}$.

- Accepted strings must be of length 0 or more
- Accepted strings must start with 'a's and end with 'c's, where the number of 'c's is double the number of 'a's. Any number of 'b's can appear between the 'a's and 'c's
- The above rule can also be represented as: $a^x b^y c^{2x}$ where x and y are both whole numbers.
- Examples of accepted strings: "acc", "aabcccc", "b", "aaabbcccccc", ""
- Note: Whole numbers are all positive integers including 0.

(A) $S \rightarrow aSc|\epsilon$
$B \rightarrow b|\epsilon$

(B) $S \rightarrow aBcc|\epsilon$
$B \rightarrow bB|\epsilon$

(C) $S \rightarrow aaScc|B$
$B \rightarrow bB|\epsilon$

(D) $S \rightarrow aScc|B|\epsilon$
$B \rightarrow bB|\epsilon$

Problem 5: Lexing, Parsing, and Evaluating

[Total 18 pts]

Given the following CFG, and assuming the **Ocaml** type system and semantics, at what stage of language processing would each expression **fail**? Mark **'Valid'** if the expression would be accepted by the grammar and evaluate successfully. Assume the only symbols allowed are those found in the grammar.

$$E \rightarrow \texttt{if not } M \textit{ then } E \textit{ else } E \mid M$$
$$M \rightarrow N > M \mid N < M \mid N$$
$$N \rightarrow 1 \mid 2 \mid 3 \mid 4 \mid \texttt{true} \mid \texttt{false} \mid (E)$$

	Lexer	Parser	Evaluator	Valid
`(true > (false)`	(L)	(P)	(E)	(V)
`if true then 4 else 1`	(L)	(P)	(E)	(V)
`2 < 3`	(L)	(P)	(E)	(V)
`if not 2 < 2 then true`	(L)	(P)	(E)	(V)
`(((2)))`	(L)	(P)	(E)	(V)
`(if not true < false then 2 < 3 else 2)`	(L)	(P)	(E)	(V)

$$E \rightarrow \texttt{if}\ M\ \textit{then}\ E\ \textit{else}\ E \mid M$$
$$M \rightarrow N = M \mid N <> M \mid N$$
$$N \rightarrow 1 \mid 2 \mid 3 \mid 4 \mid \texttt{true} \mid \texttt{false} \mid (E)$$

	Lexer	Parser	Evaluator	Valid
`5 = 5`	**L**	P	E	V
`(if true <> false then 2 = 2 else 2)`	L	P	**E**	V
`if true then 2`	L	**P**	E	V
`(if 2 = 2 then (2 <> 3) else (2 = 2)`	L	**P**	E	V
`(((true)))`	L	P	E	**V**
`if true then 2 else true`	L	P	**E**	V

$$E \rightarrow \texttt{if}\ M\ \textit{then}\ E\ \textit{else}\ E \mid M$$
$$M \rightarrow N = M \mid N <> M \mid N$$
$$N \rightarrow 1 \mid 2 \mid 3 \mid 4 \mid \texttt{true} \mid \texttt{false} \mid (E)$$

	Lexer	Parser	Evaluator	Valid
`(true = (false)`	L	**P**	E	V
`2 <> 3`	L	P	E	**V**
`if 2 = 2 then true`	L	**P**	E	V
`if true then 5 else 1`	**L**	P	E	V
`(if true <> false then 2 = 2 else 2)`	L	P	**E**	V
`(((2)))`	L	P	E	**V**

$$E \rightarrow \texttt{let}\, V\ =\ E\ in\ E \mid M$$
$$M \rightarrow N = M \mid N <> M \mid N$$
$$N \rightarrow 1 \mid 2 \mid 3 \mid 4 \mid \texttt{true} \mid \texttt{false} \mid (E)$$
$$V \rightarrow x \mid y \mid z$$

	Lexer	Parser	Evaluator	Valid
`x = 3`	L	**P**	E	V
`let a = 3 in a`	**L**	P	E	V
`let x = 5`	**L**	P	E	V
`((true))`	L	P	E	**V**
`true <> (false)`	L	P	E	**V**
`let x = 1 in x = true`	L	**P**	E	V

Problem 6: Coding and Debugging

[Total 16 pts]

Recall the interpreter code done in discussion/project 4/lecture. Given the following operational semantics rules, write a function that will return the the final value of the expression. Whenever there is an issue of incorrect typing, raise an "Unexpected Type" error by doing `raise (UnexpectedType "unexpected type")`.

$$\frac{}{A; \text{true} \rightarrow \text{true}} \qquad \frac{}{A; \text{false} \rightarrow false}$$

$$\frac{}{A; n \rightarrow n} \qquad \frac{A; e_1 \rightarrow v1 \quad A; \vdash e_2 \rightarrow v_2 \quad v_3 \; is \; v1 \; \&\& \; not \; v2}{A; \; e_1 \; e_2 \; op1 \; \rightarrow v_3}$$

$$\frac{A; e_1 \rightarrow v_1 \quad A; e_2 \rightarrow v_2 \quad v_3 \; is \; v1 < v2}{A; e_1 < e_2 \rightarrow v_3} \qquad \frac{A; e_1 \rightarrow v_1 \quad A; e_2 \rightarrow v_2 \quad v_3 = v_1 * v_2}{A; e_1 \; e_2 \; \text{op2} \rightarrow v_3}$$

```
type ast = Op1 of ast * ast | Op2 of ast * ast
          | LT of ast * ast | Int of int | Bool of bool
type expr = Int of int | Bool of bool

Examples:
evaluate (Op1(Bool(true), LT(Int(5), Int(2)))) = Bool true
evaluate (Op2(Int(2), Int(4))) = Int 8
evaluate (Int(2)) = Int 2
evaluate (Bool(false)) = Bool false
```

```
let rec evaluate (ast: ast) : expr  = match ast with
    | Int x -> Int x
    | Bool b -> Bool b
    | LT(e1, e2) -> let res1 = evaluate e1 in
                    let res2 = evaluate e2 in
                    (match res1, res2 with
                        | Int n1, Int n2 -> Bool (n1 < n2)
                        | Bool b1, Bool b2 -> Bool (b1 < b2)
                        | _, _ -> raise (UnexpectedType "unexpected type"))
    | Op1(e1, e2) -> let res1 = evaluate e1 in
                     let res2 = evaluate e2 in
                     (match res1, res2 with
                        | Bool b1, Bool b2 -> Bool (b1 && (not b2))
                        | _, _ -> raise (UnexpectedType "unexpected type"))
    | Op2(e1, e2) -> let res1 = evaluate e1 in
                     let res2 = evaluate e2 in
                     (match res1, res2 with
                        | Int n1, Int n2 -> Int (n1 * n2)
                        | _, _ -> raise (UnexpectedType "unexpected type"))
```

Recall the interpreter code done in discussion/project 4/lecture. Given the following type checking rules, write a function that will return the type rather than the value of an expression. Whenever there is an issue of incorrect typing, raise an "Unexpected Type" error by doing `raise (UnexpectedType "unexpected type")`.

$$\frac{}{G \vdash \text{true} : \text{bool}} \qquad \frac{}{G \vdash \text{false} : \text{bool}}$$

$$\frac{}{G \vdash n : int} \qquad \frac{G \vdash e_1 : int \quad G \vdash e_2 : int \quad + : int \to int \to int}{G \vdash e_1 + e_2 : int}$$

$$\frac{G \vdash e_1 : t \quad G \vdash e_2 : t \quad < : t \to t \to bool}{G \vdash e_1 < e_2 : bool} \qquad \frac{G \vdash e_1 : bool \quad G \vdash e_2 : t \quad G \vdash e_3 : t}{A; if\ e_1\ then\ e_2\ else\ e_3 : t}$$

```
type ast = If of ast * ast * ast | Add of ast * ast
          | LT of ast * ast | Int of int | Bool of bool
type tipe = TInt | TBool

Examples:
typecheck (If(Bool true, Int 4, Add(Int 4, Int 4))) = TInt
typecheck (Bool true < Bool false) = TBool
typecheck (Int 4) = TInt
typecheck (Bool true) = TBool
```

```
    let rec typecheck (ast: ast) : tipe  =
        | Int x -> TInt
        | Bool b -> TBool
        | LT(e1, e2) -> let res1 = typecheck e1 in
                        let res2 = typecheck e2 in
                        (match res1, res2 with
                            | TInt, TInt -> TBool
                            | TBool, TBool -> TBool
                            | _, _ -> raise (UnexpectedType "unexpected type"))
        | Add(e1, e2) -> let res1 = typecheck e1 in
                         let res2 = typecheck e2 in
                         (match res1, res2 with
                            | TInt, TInt -> TInt
                            | _, _ -> raise (UnexpectedType "unexpected type"))
        | If(e1, e2, e3) -> let res1 = typecheck e1 in
                            let res2 = typecheck e2 in
                            let res3 = typecheck e3 in
                             (match res1, res2, res3 with
                                | TBool, t1, t2 -> if t1 = t2 then
                                                       t1
                                                   else
                                                       raise (UnexpectedType "unexpected type"))
                                | _, _, _ -> raise (UnexpectedType "unexpected type"))
```

Problem 7: Lambda Calculus [Total 20 pts]

(a) Reduce [10 pts]

Reduce the following lambda expression to beta normal form using eager evaluation. Show every step, including alpha conversions, if you used any.

$$(\lambda x.\ y\ (x\ \lambda y.\ x)\ z)\ ((\lambda z.\ z)\ a)$$

$(\lambda x.\ y\ (x\ \lambda y.\ x)\ z)\ ((\lambda z.\ z)\ a)$

$(\lambda x.\ y\ (x\ \lambda y.\ x)\ z)\ a$

$(y\ (a\ \lambda y.\ a)\ z)$

$$(\lambda x.\ (x\lambda y.\ x)\ y)((\lambda a.\ a)\ b)$$

$(\lambda x.\ (x\lambda y.\ x)\ y)\ ((\lambda a.\ a)\ b)$

$(\lambda x.\ (x\lambda y.\ x)\ y)\ b$

$(b\ \lambda y.\ b)\ y)$

$$(\lambda x.\ (x\lambda y.\ x)\ y)((\lambda a.\ a\ c)\ b)$$

$(\lambda x.\ (x\lambda y.\ x)\ y)\ ((\lambda a.\ a\ c)\ b)$

$(\lambda x.\ (x\lambda y.\ x)\ y)\ (b\ c)$

$((b\ c)\ \lambda y.\ (b\ c)\ y)$

(b) Free Variables: [6 pts]

Circle the free variables in the expression below:

$$(\lambda a.(\lambda y.a\ \text{x})\ \text{y}\ \text{y})\ \text{a}\ ((\lambda z.\text{x}\ (\lambda z.z))\ \text{z})$$

$$(\lambda z.(\lambda y.z\ y)\ \text{y})\ \text{z}\ (\lambda x.x\ (\lambda y.\text{c}\ y)\ \text{z}\ \text{y}))\ \text{a}$$

$$(\text{x}\ \lambda a.(\lambda y.a\ \text{x})\ \text{y})\ \text{a}\ (\lambda z.\text{x}\ (\lambda z.z)\ \text{a})$$

(c) Alpha Equivalence: [4 pts]

Which of the following are alpha equivalent to the expression above, $(\lambda a.(\lambda y.a\ x)\ y\ y)\ a\ ((\lambda z.x\ (\lambda z.z))\ z)$? Select all that apply.

(A) $(\lambda b.(\lambda d.b\ x)\ dd)\ a\ ((\lambda z.x\ (\lambda c.c))\ z)$
(B) $(\lambda a.(\lambda y.a\ x)\ y\ y)\ a\ ((\lambda b.x\ (\lambda b.b))\ b)$
(C) $(\lambda c.(\lambda b.c\ x)\ bb)\ c\ ((\lambda z.x\ (\lambda d.d))\ z)$
(D) $(\lambda b.(\lambda y.b\ x)\ y\ y)\ a\ ((\lambda z.x\ (\lambda c.c))\ z)$

Which of the following are alpha equivalent to the expression above, $(\lambda z.(\lambda y.z\ y)\ y)\ z\ (\lambda x.x\ (\lambda y.cy)\ z\ y))\ a$? Select all that apply.

(A) $(\lambda a.(\lambda y.a\ y)\ y)\ a\ (\lambda x.x\ (\lambda t.ct)\ z\ t))\ a$
(B) $(\lambda b.(\lambda a.b\ a)\ y)\ z\ (\lambda d.d\ (\lambda m.cm)\ z\ y))\ a$
(C) $(\lambda a.(\lambda b.b\ a)\ y)\ z\ (\lambda x.x\ (\lambda y.cy)\ a\ y))\ d$
(D) $(\lambda y.(\lambda z.y\ y)\ z)\ z\ (\lambda x.x\ (\lambda y.cy)\ z\ y))\ a$

Which of the following are alpha equivalent to the expression above, $(x\ \lambda a.(\lambda y.a\ x)\ y)\ a\ (\lambda z.x\ (\lambda z.z)\ a)$? Select all that apply.

(A) $(d\ \lambda a.(\lambda y.a\ d)\ y)\ a\ (\lambda z.x\ (\lambda z.z)\ a)$
(B) $(x\ \lambda b.(\lambda y.b\ x)\ y)\ a\ (\lambda d.x\ (\lambda z.z)\ a)$
(C) $(x\ \lambda b.(\lambda y.b\ x)\ y)\ b\ (\lambda z.x\ (\lambda z.z)\ b)$
(D) $(x\ \lambda t.(\lambda y.t\ x)\ y)\ a\ (\lambda z.x\ (\lambda d.z)\ a)$

Problem 8: Operational Semantics

[Total 18 pts]

Consider the following rules for 2 Languages. Take note of the order of e1 and e2 that is bolded in Language B.

Language 1

$$\frac{}{\text{true} \longrightarrow \text{true}}$$

$$\frac{}{\text{false} \longrightarrow \text{false}}$$

$$\frac{A(x) = v}{A; x \Rightarrow v}$$

$$\frac{A; e_1 \Rightarrow v_1 \qquad A; e_2 \Rightarrow v_2 \qquad v_3 = \text{if } v_1 \text{ then not } v_2 \text{ else } v_2}{A; e_1\ e_2\ op1\ \Rightarrow v_3}$$

$$\frac{A; e_1 \Rightarrow v_1 \qquad A, x : v_1; e_2 \Rightarrow v_2}{A; let\ x = e_1 \text{ in } e_2 \Rightarrow v_2}$$

Language 2

$$\frac{}{\text{true} \longrightarrow \text{true}}$$

$$\frac{}{\text{false} \longrightarrow \text{false}}$$

$$\frac{A(x) = v}{A; x \Rightarrow v}$$

$$\frac{A; e_1 \Rightarrow v_1 \qquad A; e_2 \Rightarrow v_2 \qquad v_3 = \text{if } v_1 \text{ then not } v_2 \text{ else } v_2}{A; op2\ \mathbf{e_2}\ \mathbf{e_1}\ \Rightarrow v_3}$$

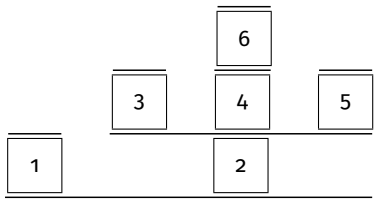
$$\frac{A; e_2 \Rightarrow v_1 \qquad A, x : v_1; e_1 \Rightarrow v_2}{A; (fun\ x \longrightarrow e_1)\ e_2 \Rightarrow v_2}$$

(a) Convert the following Language 1 sentence to its language 2 counterpart [6 pts]

A; let x = false in (false x) op1

$(fun\ x \longrightarrow op2\ x\ false)\ false$

(b) Prove the original expression by writing an OpSem proof using language 1. [12 pts]



A; let x = false in (false x) op1

Blank 1: A; false $\longrightarrow false$

Blank 2: A, x : false; false x op1 $\longrightarrow false$

Blank 3: A, x : false; false $\longrightarrow false$

Blank 4: A, x: false; x $\longrightarrow false$

Blank 5: false = if false then not false else false

Blank 6: A, x: false(x) = false

Language 1

$$\frac{}{\text{true} \longrightarrow \text{true}}$$

$$\frac{}{\text{false} \longrightarrow \text{false}}$$

$$\frac{A(x) = v}{A; x \Rightarrow v}$$

$$\frac{A; e_1 \Rightarrow v_1 \qquad A; e_2 \Rightarrow v_2 \qquad v_3 = \text{if } v_1 \text{ then not } v_2 \text{ else } v_2}{A; op1\ (e_1\ e_2) \Rightarrow v_3}$$

$$\frac{A; e_1 \Rightarrow v_1 \qquad A, x : v_1; e_2 \Rightarrow v_2}{A; let\ x = e_1 \text{ in } e_2 \Rightarrow v_2}$$

Language 2

$$\frac{}{\text{true} \longrightarrow \text{true}}$$

$$\frac{}{\text{false} \longrightarrow \text{false}}$$

$$\frac{A(x) = v}{A; x \Rightarrow v}$$

$$\frac{A; e_1 \Rightarrow v_1 \qquad A; e_2 \Rightarrow v_2 \qquad v_3 = \text{if } v_1 \text{ then not } v_2 \text{ else } v_2}{A; \mathbf{e}_2\ op2\ \mathbf{e}_1 \Rightarrow v_3}$$

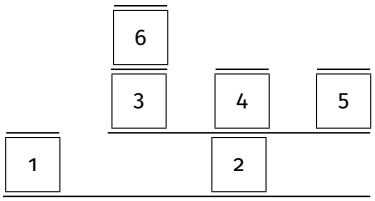
$$\frac{A; e_2 \Rightarrow v_1 \qquad A, x : v_1; e_1 \Rightarrow v_2}{A; (fun\ x \longrightarrow e_1)\ e_2 \Rightarrow v_2}$$

(c) Convert the following Language 1 sentence to its language 2 counterpart [6 pts]

A; let x = true in op1 (x false)

$(fun\ x \longrightarrow false\ op2\ x)\ true$

(d) Prove the original expression by writing an OpSem proof using language 1. [12 pts]



A; let x = true in op1 (x false)

Blank 1: A; true $\longrightarrow true$

Blank 2: A, x : true; op1 x false $\longrightarrow true$

Blank 3: A, x : true; x $\longrightarrow true$

Blank 4: A, x: true; false $\longrightarrow false$

Blank 5: true = if true then not false else false

Blank 6: A, x: true(x) = true

Language 1

$$\frac{}{\text{true} \longrightarrow \text{true}}$$

$$\frac{}{\text{false} \longrightarrow \text{false}}$$

$$\frac{A(x) = v}{A; x \Rightarrow v}$$

$$\frac{A; e_1 \Rightarrow v_1 \qquad A; e_2 \Rightarrow v_2 \qquad v_3 = \text{if } v_1 \text{ then not } v_2 \text{ else } v_2}{A; (e_1\ e_2)\ op1 \Rightarrow v_3}$$

$$\frac{A; e_1 \Rightarrow v_1 \qquad A, x : v_1; e_2 \Rightarrow v_2}{A; let\ x = e_1 \text{ in } e_2 \Rightarrow v_2}$$

Language 2

$$\frac{}{\text{true} \longrightarrow \text{true}}$$

$$\frac{}{\text{false} \longrightarrow \text{false}}$$

$$\frac{A(x) = v}{A; x \Rightarrow v}$$

$$\frac{A; e_1 \Rightarrow v_1 \qquad A; e_2 \Rightarrow v_2 \qquad v_3 = \text{if } v_1 \text{ then not } v_2 \text{ else } v_2}{A; op2\ (\mathbf{e}_2\ \mathbf{e}_1) \Rightarrow v_3}$$

$$\frac{A; e_2 \Rightarrow v_1 \qquad A, x : v_1; e_1 \Rightarrow v_2}{A; (fun\ x \longrightarrow e_1)\ e_2 \Rightarrow v_2}$$

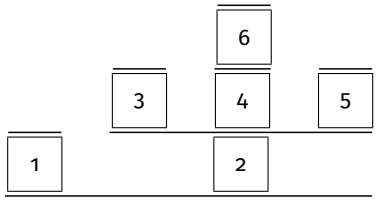
(e) Convert the following Language 1 sentence to its language 2 counterpart [6 pts]

A; let x = true in (true x) op1

$(fun\ x \longrightarrow op2\ x\ true)\ true$

(f) Prove the expression you wrote above by writing an OpSem proof using language 1.

[12 pts]



A; let x = true in (true x) op1

Blank 1: A; true $\longrightarrow true$

Blank 2: A, x : true; true x op1 $\longrightarrow false$

Blank 3: A, x : true; true $\longrightarrow true$

Blank 4: A, x: true; x $\longrightarrow true$

Blank 5: false = if true then not true else true

Blank 6: A, x: true(x) = true

Language 1

$$\frac{}{\text{true} \longrightarrow \text{true}}$$

$$\frac{}{\text{false} \longrightarrow \text{false}}$$

$$\frac{A(x) = v}{A; x \longrightarrow v}$$

$$\frac{A; e_1 \longrightarrow v_1 \qquad A; e_2 \longrightarrow v_2 \qquad v_3 = \text{if } v_1 \text{ then } v_1 \text{ else not } v_2}{A;\ e_1\ e_2\ op1 \longrightarrow v_3}$$

$$\frac{A; e_1 \longrightarrow v_1 \qquad A, x : v_1; e_2 \longrightarrow v_2}{A; let\ x = e_1 \text{ in } e_2 \longrightarrow v_2}$$

Language 2

$$\frac{}{\text{true} \longrightarrow \text{true}}$$

$$\frac{}{\text{false} \longrightarrow \text{false}}$$

$$\frac{A(x) = v}{A; x \longrightarrow v}$$

$$\frac{A; e_1 \longrightarrow v_1 \qquad A; e_2 \longrightarrow v_2 \qquad v_3 = \text{if } v_1 \text{ then } v_1 \text{ else not } v_2}{A; \mathbf{e}_2\ op2\ \mathbf{e}_1 \longrightarrow v_3}$$

$$\frac{A; e_2 \longrightarrow v_1 \qquad A, x : v_1; e_1 \longrightarrow v_2}{A; (fun\ x \longrightarrow e_1)\ e_2 \longrightarrow v_2}$$

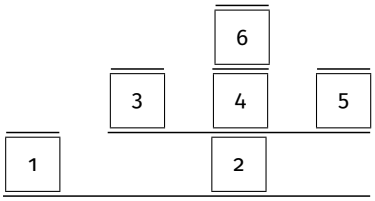
(g) Convert the following Language 1 sentence to its language 2 counterpart [6 pts]

let y = false in false y op1

$(fun\ y \longrightarrow\ y\ op2\ false)\ false$

(h) Prove the expression you wrote above by writing an OpSem proof using language 1.

[12 pts]



A; let y = false in false y op1 $\longrightarrow$ true

Blank 1: A; false $\longrightarrow$ $false$

Blank 2: A, y : false; false y op1 $\longrightarrow$ $true$

Blank 3: A, y : false; false $\longrightarrow$ $false$

Blank 4: A, y: false; y $\longrightarrow$ $false$

Blank 5: true = if false then false else not false

Blank 6: A, y: false(y) = false