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)

Problem 2: Context Free Grammars - Acceptance

[Total 6 pts]

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

$$
\begin{aligned}
S &\rightarrow \; greater?\ M\ \ N \mid N\ ?\ M\ \ :\ \ M \mid M \\
M &\rightarrow \; add\ \ M\ \ M \mid sub\ \ M\ \ M \mid N \\
N &\rightarrow \; n \mid b \mid (S)
\end{aligned}
$$

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)`

Problem 3: Context Free Grammars - Derivations

[Total 8 pts]

$$
\begin{aligned}
S &\rightarrow \; greater?\ M\ \ N \mid N\ ?\ M\ \ :\ \ M \mid M \\
M &\rightarrow \; add\ \ M\ \ M \mid sub\ \ M\ \ M \mid N \\
N &\rightarrow \; n \mid b \mid (S)
\end{aligned}
$$

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

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

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

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)

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  =
```

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.

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

(b) Free Variables: [6 pts]

Circle the free variables in the expression below:

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

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

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

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

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

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

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

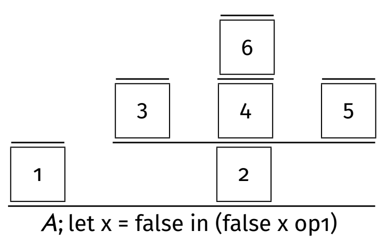
$$\frac{A; e_2 \Rightarrow v_1 \qquad A, x : v_1; e_1 \Rightarrow v_2}{A; (fun\ x \longrightarrow e_1)\ e_2 \Rightarrow v_2}$$

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

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

(b) Prove the given expression by writing an OpSem proof using language 1.

[12 pts]



Scratch space:

IMPORTANT: you must fill in the blanks in the next page to receive credit.

Blank 1:

Blank 2:

Blank 3:

Blank 4:

Blank 5:

Blank 6: