CMSC330 - Organization of Programming Languages  
Spring 2023 - 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

• You may use anything on the accompanying reference sheet anywhere on this exam
• Please write legibly. **If we cannot read your answer you will not receive credit**
• You may not leave the room or hand in your exam within the last 10 minutes of the exam
• The last page is blank and scratch work can be done there.
• If anything is unclear, ask a proctor. If you are still confused, write down your assumptions in the margin

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>10</td>
</tr>
<tr>
<td>Q2</td>
<td>18</td>
</tr>
<tr>
<td>Q3</td>
<td>10</td>
</tr>
<tr>
<td>Q4</td>
<td>12</td>
</tr>
<tr>
<td>Q5</td>
<td>15</td>
</tr>
<tr>
<td>Q6</td>
<td>15</td>
</tr>
<tr>
<td>Total</td>
<td>80</td>
</tr>
</tbody>
</table>
Problem 1: Language Concepts

[Total 10 pts]

An improper garbage collector can cause security vulnerabilities True False

Modern Languages use a combination of Reference Counting, Mark and Sweep and Stop and Copy True False

Lambda Calculus Expressions can be converted to Finite State Machines True False

The relation of FSM to Regex is bijective (1 to 1) True False

Eager and Lazy Evaluation will always give the same result True False

Problem 2: Finite State Machines

[Total 18 pts]

(a) Using the subset algorithm, convert the following NFA to a DFA, and fill in the blanks appropriately matching the DFA provided with the right nodes and transitions. Only the blanks will be graded.  

NFA: 

DFA:

Scratch Space (if needed)

S1:  S2:  S3:  S4:  S5:
E1:  E2:  E3:  E4:  E5:
E6:  E7:  E8:
(b) Write a regex to describe the language of the above NFA

(c) Vending Machine Fun

Suppose there is a vending machine which takes in quarters (Q), dimes (D) and nickles (N). Consider the following actions you can perform when interacting with the vending machine:

Action N: Insert a Nickle    Action D: Insert a Dime    Action Q: Insert a Quarter

The price of each item is $0.25. However, the FSM for the machine was leaked and turns out you can pay less than $0.25 per item. List out the operations you want to perform to pay less than $0.25. For example, if you wanted to put in 2 quarters, followed by 1 dime, followed by 3 nickles, your answer should be Q, Q, D, N, N, N.
Problem 3: CFGs

Consider the following Grammars:

<table>
<thead>
<tr>
<th>Grammar 1</th>
<th>Grammar 2</th>
<th>Grammar 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow aSb$</td>
<td>$S \rightarrow AAASB \mid e$</td>
<td>$S \rightarrow ASB$</td>
</tr>
<tr>
<td>$\mid aaSb$</td>
<td>$A \rightarrow a \mid e$</td>
<td>$A \rightarrow aA \mid e$</td>
</tr>
<tr>
<td>$\mid aaaSb$</td>
<td>$B \rightarrow b$</td>
<td>$B \rightarrow bbbbB \mid e$</td>
</tr>
<tr>
<td>$\mid e$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) Which of the following grammars describe strings of $a^x b^y$, $x < 3y$? Select all that apply. [2 pts]

Grammar 1   Grammar 2   Grammar 3   None

(b) Prove that Grammar 2 is ambiguous [3 pts]

(c) Draw the abstract syntax tree that would be generated by parsing the following string with the given CFG using a leftmost derivation.

String: "1 * 2 + 3"

CFG:
$S \rightarrow M * S \mid M$
$M \rightarrow M + N \mid N$
$N \rightarrow 1 \mid 2 \mid 3 \mid (N)$, where n is any number
Problem 4: Operational Semantics

Consider the following rules for LOLCODE, using OCaml as the Metalanguage:

Rule 1: \( \text{WIN} \rightarrow \text{WIN} \)

Rule 2: \( \text{FAIL} \rightarrow \text{FAIL} \)

Rule 3: \( A; e_1 \Rightarrow v_1 \quad A; e_2 \Rightarrow v_2 \quad v_1 \leftrightarrow v_2 \)

\( A; \text{DIFFRINT } e_1 \text{ AN } e_2 \Rightarrow \text{WIN} \)

Rule 4: \( A; e_1 \Rightarrow v_1 \quad A; e_2 \Rightarrow v_2 \quad v_1 = v_2 \)

\( A; \text{DIFFRINT } e_1 \text{ AN } e_2 \Rightarrow \text{FAIL} \)

Rule 5: \( A; e_1 \Rightarrow v_1 \quad A; e_2 \Rightarrow v_2 \quad v_3 = \text{if } v_1 > v_2 \text{ then } v_1 \text{ else } v_2 \)

\( A; \text{BIGGR OF } e_1 \text{ AN } e_2 \Rightarrow v_3 \)

Rule 6: \( A; e_1 \Rightarrow v_1 \quad A. x : v_1; e_2 \Rightarrow v_2 \)

\( A; \text{HAS A x ITZ } e_1 \text{ \n e_2 } \Rightarrow v_2 \)

Rule 7: \( A; e_1 \Rightarrow v_1 \quad A; e_2 \Rightarrow v_2 \quad v_3 = \text{if } v_1 > v_2 \text{ then } v_1 \text{ else } v_2 \)

\( A; \text{BIGGR OF } e_1 \text{ AN } e_2 \Rightarrow v_3 \)

Rule 8: \( A; n \rightarrow n \)

(a) What are the axioms in this language? Select all the apply.

Rule 1   Rule 2   Rule 3   Rule 4   Rule 5   Rule 6   Rule 7   Rule 8   none

(b) Complete the opsem proof for the following program:

\( \text{HAS A x ITZ 7 \n DIFFRINT 2 AN (BIGGR OF 2 AN x) } \Rightarrow \text{WIN} \)

\( A. x : 7; e_1 \Rightarrow v_1 \quad A. x : 7; e_2 \Rightarrow v_2 \quad v_3 = \text{if } v_1 > v_2 \text{ then } v_1 \text{ else } v_2 \)

\( A; \text{BIGGR OF } e_1 \text{ AN } e_2 \Rightarrow v_3 \)

\( A; \text{HAS A x ITZ } e_1 \text{ \n e_2 } \Rightarrow v_2 \)

\( A; e_1 \Rightarrow v_1 \quad A. x : v_1; e_2 \Rightarrow v_2 \quad v_1 = v_2 \)

\( A; \text{DIFFRINT } e_1 \text{ AN } e_2 \Rightarrow \text{FAIL} \)

\( A; e_1 \Rightarrow v_1 \quad A; e_2 \Rightarrow v_2 \quad v_1 <> v_2 \)

\( A; \text{DIFFRINT } e_1 \text{ AN } e_2 \Rightarrow \text{WIN} \)

Blank 1:   Blank 2:   Blank 3:   Blank 4:   Blank 5:   Blank 6:   Blank 7:   Blank 8:   
Problem 5: Lambda Calculus

For the following questions perform a single $\beta$-reduction using lazy (call by name) evaluation on the outermost expression. If you cannot reduce it, write Beta Normal Form. You may not $\alpha$-convert your final answer.

(a) $(\lambda x. x y) (y (\lambda x. y x))$ [2 pts]

(b) $(\lambda x. x x) ((\lambda x. y x) ((\lambda a. a a) b))$ [2 pts]

For the following questions perform a single $\beta$-reduction using Eager (call by value) evaluation on the outermost expression. If you cannot reduce it, write Beta Normal Form. You may not $\alpha$-convert your final answer.

(c) $(\lambda x. x y. y) (y (\lambda x. y x))$ [2 pts]

(d) $(\lambda x. x x) ((\lambda x. y x) ((\lambda a. a a) b))$ [2 pts]

(e) Convert the following to Beta Normal Form: $(\lambda x. (\lambda y. a a) b) (\lambda x. a x)$ [3 pts]

\[ \lambda x. a x \quad c \quad d \quad b \quad a \quad a \quad \text{can't reduce} \quad \text{infinite recursion} \quad \text{None} \]

Consider the following lambda calculus bindings:
true $= \lambda x. y. x$
false $= \lambda x. y. y$
if $e_1$ then $e_2$ else $e_3 = e_1 \ e_2 \ e_3$

(f) Encode the following expression: if false then false else true [4 pts]
Problem 6: Lexing, Parsing, Evaluation

Consider the following modified Math-ew from lecture:

\[
E \Rightarrow + E | * E E | sq E | exp E E | and E E | or E E | N
\]
\[
N \Rightarrow 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | true | false
\]

You may assume that the behaviour is the same as Ocaml.

(a) Lexing [5 pts]

Which of the following phrases would fail the lexing stage for the Math-ew Language? Please bubble in the circle

A. \(2 * 3 sq 2 3\)
B. \(4 ^ 5\)
C. \(- + 1 23\)
D. \(exp -2 5\)
E. \(5 exp 2 + 6\)
F. \(* 2 and true false\)
G. \(and true or false false\)
H. \(false true\)
I. \(true and false or true\)

(b) Parsing [5 pts]

Which of the following phrases would fail the parsing stage for the Math-ew Language? If it failed the lexing phase, do not mark it.

A. \(2 * 3 sq 2 3\)
B. \(4 ^ 5\)
C. \(- + 1 23\)
D. \(exp -2 5\)
E. \(5 exp 2 + 6\)
F. \(* 2 and true false\)
G. \(and true or false false\)
H. \(false true\)
I. \(true and false or true\)

(c) Evaluation [5 pts]

Which of the following phrases would fail the evaluator stage for the Math-ew Language? If it failed the lexing or parsing phase, do not mark it.

A. \(2 * 3 sq 2 3\)
B. \(4 ^ 5\)
C. \(- + 1 23\)
D. \(exp -2 5\)
E. \(5 exp 2 + 6\)
F. \(* 2 and true false\)
G. \(and true or false false\)
H. \(false true\)
I. \(true and false or true\)
You can use this page for scratch work.