CMSC330 - Organization of Programming Languages
Spring 2023 - Exam 2 Solution

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><strong>Total</strong></td>
<td><strong>80</strong></td>
</tr>
</tbody>
</table>
**Problem 1: Language Concepts**

[Total 10 pts]

- An improper garbage collector can cause security vulnerabilities
  - True  False

- Having things stay in memory for too long is a security vulnerability
  - True  False

- Modern Languages use a combination of Reference Counting, Mark and Sweep and Stop and Copy
  - Stated in Lecture, more efficient to do this
  - True  False

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

- Lambda Calculus cannot be expressed as a FSM
  - True  False

- The relation of FSM to Regex is bijective (1 to 1)
  - Some NFAs represent the same regex
  - True  False

- Eager and Lazy Evaluation will always give the same result
  - Consider cases of infinite reduction: \((\lambda x.a)((\lambda x.xx)(\lambda y.yy))\)
  - 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.  

[12 pts]

NFA: 

```
<table>
<thead>
<tr>
<th>State</th>
<th>Transitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a, e</td>
</tr>
<tr>
<td>1</td>
<td>e, a</td>
</tr>
<tr>
<td>2</td>
<td>e, b</td>
</tr>
<tr>
<td>3</td>
<td>c, e</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
</tr>
<tr>
<td>5</td>
<td>e</td>
</tr>
</tbody>
</table>
```

DFA: 

```
<table>
<thead>
<tr>
<th>State</th>
<th>Transitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>E1</td>
</tr>
<tr>
<td>S2</td>
<td>E2, E8</td>
</tr>
<tr>
<td>S3</td>
<td>E3, E4, E5, E6</td>
</tr>
<tr>
<td>S4</td>
<td>E7, E8</td>
</tr>
</tbody>
</table>
```

So since state S4 and S2 can be swapped, any transitions to them can also be interchanged.
(b) Write a regex to describe the language of the above NFA

\[(a+)|(b|c)+\]

States 0 and 1 represent a+, States 2, 4, 5 represent b+, States 2, 3, 5 represent c+. So together, 2, 3, 4, 5 represents (c|b)+.

(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.

\text{N,D,N is $0.20 (S_0 \rightarrow S_4 \rightarrow S_5 \rightarrow S_1)}
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 \epsilon )</td>
<td>( S \rightarrow ASB )</td>
</tr>
<tr>
<td>( \mid a\alpha Sb )</td>
<td>( A \rightarrow a \mid \epsilon )</td>
<td>( A \rightarrow aA \mid \epsilon )</td>
</tr>
<tr>
<td>( \mid a\alpha \alpha Sb )</td>
<td>( B \rightarrow b )</td>
<td>( B \rightarrow bbbB \mid \epsilon )</td>
</tr>
<tr>
<td>( \mid \epsilon )</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

Not possible because if \( y = 0 \), then \( x \) has to be negative. Grammar 1 and Grammar 2 describe the same thing: \( x \leq 3y \). The third does not terminate.

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

\[
S \rightarrow AAASB \rightarrow AASB \rightarrow ASB \rightarrow aSB \rightarrow aB \rightarrow ab \\
S \rightarrow AAASB \rightarrow aAASB \rightarrow aASB \rightarrow aSB \rightarrow aB \rightarrow ab
\]

(c) Draw the abstract syntax tree that would be generated by parsing the following string with the given CFG using a leftmost derivation. [5 pts]

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

```
    *        1
   / \       / \\
  +   3     4
```

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:  \[ \text{A}; e_1 \Rightarrow v_1 \quad \text{A}; e_2 \Rightarrow v_2 \quad v_1 \neq v_2 \quad \text{A}; \text{DIFFRINT} e_1 \text{ AN } e_2 \Rightarrow \text{WIN} \]

Rule 4:  \[ \text{A}; e_1 \Rightarrow v_1 \quad \text{A}; e_2 \Rightarrow v_2 \quad v_1 = v_2 \quad \text{A}; \text{DIFFRINT} e_1 \text{ AN } e_2 \Rightarrow \text{FAIL} \]

Rule 5:  \[ \text{A}; e_1 \Rightarrow v_1 \quad \text{A}; e_2 \Rightarrow v_2 \quad \text{A}; \text{HAS A x ITZ} e_1 \text{ \& } e_2 \Rightarrow v_2 \]

Rule 6:  \[ \text{A}; e_1 \Rightarrow v_1 \quad \text{A}; x : v_1; e_2 \Rightarrow v_2 \quad \text{A}; \text{HAS A x ITZ} e_1 \text{ \& } e_2 \Rightarrow v_2 \]

Rule 7:  \[ \text{A}; e_1 \Rightarrow v_1 \quad \text{A}; e_2 \Rightarrow v_2 \quad v_3 = \text{if } v_1 > v_2 \text{ then } v_1 \text{ else } v_2 \quad \text{A}; \text{BIGGR OF} e_1 \text{AN } e_2 \Rightarrow v_3 \]

Rule 8:  \[ \text{A}; n \Rightarrow n \]

(a) What are the axioms in this language? Select all that 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:

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

\[ \begin{array}{c}
\text{A}; x : 7; 2 \Rightarrow 2 \\
\text{A}; x : 7; x \Rightarrow 7 \\
\text{A}; x : 7; 4 \Rightarrow 5 \\
\text{A}; x : 7; \text{ DIFFRINT 2 AN } 2 \Rightarrow \text{WIN}
\end{array} \]

Blank 1: \[ \text{A}; 7 \Rightarrow 7 \]

Blank 2: \[ \text{BIGGR OF 2 AN x} \]

Blank 3: \[ \text{A}; x : 7; 2 \Rightarrow 2 \]

Blank 4: \[ \text{BIGGR OF 2 AN x} \]

Blank 5: \[ 7 \]

Blank 6: \[ 7 \Rightarrow 7 \]

Blank 7: \[ \text{if } 2 > 7 \text{ then } 2 \text{ else } 7 \]

Blank 8: \[ \text{A}; x : 7(x) \Rightarrow 7 \]
**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) $(\lambda x.x\lambda y. xy)(y(\lambda x.yx))$

$$y(\lambda x.yx)\lambda y.(y(\lambda x.yx))y$$
- We use $(y(\lambda x.yx))$ as input to $(\lambda x.x\lambda y. xy)$

(b) $(\lambda x.\lambda x.xx)((\lambda x.yx)((\lambda a.aa)b))$

$$\lambda x.xx$$
- We use $(\lambda x.yx)((\lambda a.aa)b)$ as input to $(\lambda x.\lambda x.xx)$

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) $(\lambda x.x\lambda y. xy)(y(\lambda x.yx))$

$$y(\lambda x.yx)\lambda y.(y(\lambda x.yx))y$$
- We still need to beta reduce
- but since the argument cannot be reduced further, we just reduce the outermost expression. Same as part (a)

(d) $(\lambda x.\lambda x.xx)((\lambda x.yx)((\lambda a.aa)b))$

$$(\lambda x.\lambda x.xx)((\lambda x.yx)(bb))$$
- We have to evaluate the argument, but to do so, we need to evaluate $((\lambda a.aa)b)$ first

(e) Convert the following to Beta Normal Form: $(\lambda x.(\lambda y.xa)b)(\lambda x.ax)$

$$\lambda x.ax$$
- $c d$ b a
- $a a$
- can’t reduce
- infinite recursion
- None

Consider the following lambda calculus bindings:

true = $\lambda x.\lambda y. x$
false = $\lambda x.\lambda y. y$
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

$$\lambda x.\lambda y.x)(\lambda x.\lambda y.y)(\lambda x.\lambda y.x)$$
Problem 6: Lexing, Parsing, Evaluation

Consider the following modified Math-ew from lecture:

\[ E \Rightarrow + E \mid * E \mid sq E \mid exp E \mid and E \mid or E \mid N \]
\[ \quad N \Rightarrow \minuses 0 \mid \minuses 1 \mid \minuses 2 \mid \minuses 3 \mid \minuses 4 \mid \minuses 5 \mid \minuses 6 \mid \minuses 7 \mid \minuses 8 \mid \minuses 9 \mid \minuses \text{true} \mid \minuses \text{false} \]

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 2 3
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, C, D: Basically, which phrases have symbols not in the grammar?

(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 2 3
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

A, E, H, I: Basically, which phrases are grammatically incorrect?

(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 2 3
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

F: Basically, which phrases don't make sense? The only 2 left are F and G. Since we said behaviour is same as Ocaml, we can definitely or and and boolens of true and false. We cannot however, multiply 2 and the result of and-ing true and false.
You can use this page for scratch work: