Proctoring TA: ________________________   Name: ________________________

Problem 1: Basics

[3 pts]

Please circle **True** or **False** for the following statements:

- All Regular Expressions can be expressed as CFGs  
  - **True**  
  - **False**

- Regular expressions have a starting character and terminals and nonterminals, all things that can be expressed as a CFG.
  - **True**
  - **False**

- One could theoretically implement NFA to DFA in Lambda Calculus  
  - **True**  
  - **False**

- Since lambda calculus expresses functions, one could write a function that converts an NFA to a DFA using lambda calc.
  - **True**  
  - **False**

- Ambiguous Grammars have at maximum, one right-most and one left-most derivation for any given string  
  - **True**  
  - **False**

- Ambiguous grammars have more than one left-most derivation or more than one right most derivation of the same string.  
  - **True**  
  - **False**

Problem 2: Operational Semantics

[6 pts]

Consider the following rules for a subset of OCaml

\[
\begin{align*}
\text{true} & \rightarrow \text{true} \\
\text{false} & \rightarrow \text{false} \\
A; e_1 \Rightarrow v_1 & \quad A; e_2 \Rightarrow v_2 \quad \vdash v_1 \text{ is equal to } v_2 \\
A; e_1 \not\Rightarrow e_2 & \Rightarrow \text{false} \\
A; e_1 \Rightarrow v_1 & \quad A; e_2 \Rightarrow v_2 \quad \vdash v_1 \text{ is not } v_2 \\
A; e_1 \not\Rightarrow e_2 & \Rightarrow \text{true} \\
A; x; v; (x) = v & \Rightarrow A; x; v; x \Rightarrow v \\
A; e_1 \Rightarrow v_1 & \quad A; x; v_1; e_2 \Rightarrow v_2 \\
A; \text{let } x = e_1 \text{ in } e_2 & \Rightarrow v_2 \\
A; e_1 \Rightarrow v_1 & \quad A; e_2 \Rightarrow v_2 \\
A; \text{max}(e_1, e_2) \Rightarrow v_3 & \Rightarrow v_3 \text{ is the max of } v_1 \text{ and } v_2 \\
A; n \Rightarrow n &
\end{align*}
\]

Prove the following statement is valid and returns **true**

\[
\text{let } x = 8 \text{ in } 3 \not\Rightarrow \text{max}(3, x)
\]

\[
\begin{align*}
A; x : 8; 3 \Rightarrow 3 & \\
A; x : 8; x \Rightarrow 8 & \\
8 \text{ is the max of } 3 \text{ and } 8 & \\
A; x : 8; 3 \Rightarrow 3 & \\
A; x : 8; 3 \not\Rightarrow 2 & \Rightarrow \text{true}
\end{align*}
\]
Problem 3: Context Free Grammars

Consider the following Grammars:

<table>
<thead>
<tr>
<th>Grammar 1</th>
<th>Grammar 2</th>
<th>Grammar 3</th>
<th>Grammar 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>S -&gt; AB</td>
<td>S -&gt; ASB</td>
<td>c</td>
<td>S -&gt; ASB</td>
</tr>
<tr>
<td>A -&gt; aA</td>
<td>a</td>
<td>A -&gt; aA</td>
<td>a</td>
</tr>
<tr>
<td>B -&gt; bbB</td>
<td>e</td>
<td>B -&gt; bbB</td>
<td>e</td>
</tr>
</tbody>
</table>

(a) Which grammar accepts both "aaabb" and "aaabbcc"

Grammar 1 Grammar 2 Grammar 3

(b) Which Grammar is ambiguous?

Grammar 1 Grammar 2 Grammar 3

(c) Which strings are accepted by Grammar 4?

aaacbbbc  aaacbbbb  ccaabbbbcc  cacacbbbc

(a) Grammar 3 accepts both "aaabb" and "aaabbcc". Let us look at the production for both:

1. "aaabb": S -> AB -> aAB -> aaAB -> aab -> aaabbB -> aaabb (as B goes to e)
2. "aaabbcc": $S \rightarrow Sc \rightarrow Scc \rightarrow aABcc \rightarrow aaABcc \rightarrow aaabbBcc \rightarrow aaabbcc$ (as B goes to $\epsilon$)

(b) Grammar 2 is ambiguous. Let us look at two leftmost derivations of "aac":

1. $S \rightarrow ASB \rightarrow aSB \rightarrow aASB \rightarrow aaSB \rightarrow aacB \rightarrow aac$
2. $S \rightarrow ASB \rightarrow aASB \rightarrow aaSB \rightarrow aacB \rightarrow aac$

(c) Only "aaacbbbb" is accepted by Grammar 4:

1. "aaacbbb" is not accepted as $S \rightarrow ASB \rightarrow ...$ and B only terminates in an even number of "b"s. Since the string has 3 "b"s it will not be accepted.
2. "aaacbbbb": $S \rightarrow cSc \rightarrow ccScc \rightarrow ccASBcc \rightarrow ...$ Since we have an S in between A and B, and we know S will always terminate in a c, it would only accept strings which has 1 or more c's in between a's and b's. Our string here, only consists of c's at the end and not in between aaa and bbbb, therefore, it will not be accepted.

Problem 4: Lambda Calculus

(a) Circle the free variables and underline the bound variables in the following lambda calculus expression

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

Consider the following $\lambda$ expressions

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

(b) Which of the following is the result of reducing the outer-most expression once using lazy (call by name) evaluation?

\[
(\lambda x . (\lambda y . x y))a \quad \lambda y . ((\lambda y . a)(\lambda x . x))y \quad \lambda y . ay
\]

(c) Which of the following is the result of reducing the outer-most expression once using eager (call by value) evaluation?

\[
(\lambda x . (\lambda y . x y))a \quad \lambda y . ((\lambda y . a)(\lambda x . x))y \quad \lambda y . ay
\]

(a) Since $x$ is within the body of the outermost lambda expression, it is bound. Same for $y$ - since it is in the body of the inner lambda expression, it is bound. $z$ has no binding in this lambda expression and is therefore free. We can see the same thing with $w$.

(b) Consider $(\lambda x . (\lambda y . x y))$ as $e_1$ and $((\lambda y . a)(\lambda x . x))$ as $e_2$. Since we are performing lazy evaluation, we do beta reduction without evaluating $e_2$. Applying the argument $e_2$, we get $\lambda y . ((\lambda y . a)(\lambda x . x))y$

(c) Consider $(\lambda x . (\lambda y . x y))$ as $e_1$ and $((\lambda y . a)(\lambda x . x))$ as $e_2$. Now, since we are performing eager evaluation, we do beta reduction by first evaluating $e_2$. Within $e_2$ we have 2 lambda expressions. Applying the argument $(\lambda x . x)$ to the first expression, we get $(\lambda x . (\lambda y . x y))a$