ECE 553: Compilers
Midterm

Name: NetID:

There are 9 questions, with the point values as shown below. You have 75 minutes with a total of 75 points. Pace yourself accordingly.

This exam must be individual work. You may not collaborate with your fellow students. However, this exam is open book and open notes. You may use any printed materials, but no electronic nor interactice resources.

I certify that the work shown on this exam is my own work, and that I have neither given nor received improper assistance of any form in the completion of this work.

Signature:

<table>
<thead>
<tr>
<th>#</th>
<th>Question</th>
<th>Points Earned</th>
<th>Points Possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SML</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Regular Languages</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>NFA to Regexp</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>NFA to DFA</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>Regexp to NFA</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>Context Free Languages</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>LL Parsing</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>LR Parsing</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>Types</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td></td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>Percent</td>
<td></td>
<td>100</td>
</tr>
</tbody>
</table>
Part A: Evaluate the following SML expressions (1 point each):

1. let val a = 42
   val b = 20
   in
      (if (a > b) then a else b) * 2
   end

2. let val a = 7
   val q = 6
   val b = let val a = 9
           val q = 15
           in
                 a + q
           end
   in
       a - b + q
   end

3. let fun f (a, 0) = a
       | f (a, b) = f (a+2*b, b-1)
       in
           f (4, 2)
       end
Part B: Write down the type for each of these SML functions (1 point each):

1. `fun f x y = (x y) ^ (x 3)`

2. `fun f(x,[]) = []
   | f(x,a::l) = case x of
     | SOME y => y::f(x,l)
     | NONE => f(x,l)`
Question 2 Regular Languages [5 pts]

1. Write a regular expression for all strings of 0s and 1s which does not contain the substring 0011.

2. Write a regular expression for all strings of xs and ys where the number of ys between any two xs is a multiple of 3.
Question 3 NFA to Regexp [10 pts]

Convert the following NFA to a regular expression:

```
1 -> 2 -> 3 -> 4 -> 5

1: 
2: a, c
3: d, f
4: b, e
5: g
```
Workspace for question 3
Workspace for question 3
Question 4 NFA to DFA [10 pts]

Convert the following NFA to a DFA:
Question 5 Regexp to NFA [10 pts]
Draw an NFA for the following regular expressions:

1. (a |c) (b | d)

2. a*(bc)*
3. $((a(b|e)^*d) \mid e^*)f$
Question 6 Context Free Languages [5 pts]
Write context free grammars for the following languages (your grammar does not have to be LR(1), LL(1) etc):

1. The language described by the regular expression ((a|c)d | e*)f

2. The language of sequences of 0s and 1s that are palindromes (the same forwards as backwards):
Question 7 LL Parsing [10 pts]

Consider the following grammar:

\[
\begin{align*}
S &\rightarrow A \times B \\
    &\mid C \\
A &\rightarrow A \times z A \\
    &\mid y \\
B &\rightarrow b B b \\
    &\mid \\
C &\rightarrow C y \\
    &\mid B
\end{align*}
\]

• Which non-terminals (if any) can derive empty? (1 point)

• What are the FIRST sets of Q and S? (1 point)

• What are the FOLLOW sets of Q and S? (1 point)

• This grammar can not be parsed by an LL(0) or LL(1) parser. Explain why not (2 points).

• Rewrite the grammar so that it accepts the same language, but can be parsed by an LL(1) parser (5 points).
Question 8 LR Parsing [10 pts]

Consider the following grammar:

0: S -> X $
1: X -> X a X b
2: X -> c

Below is the state table (.grm.desc file) from ml-yacc, with the non-error actions removed (replaced by blanks). Your job is to fill in the missing actions (shift N, goto N, reduce by rule R where N is the appropriate state number, and R is the appropriate rule number). The first (State 0) and last (State 6) are done for you:

state 0:

S : . X

c shift 2
S goto 6
X goto 1
.
error

state 1:

S : X .
X : X . a X b

a -----------------------------
.
-----------------------------

state 2:

X : c .

.
-----------------------------

state 3:

X : X a . X b

c -----------------------------
X -----------------------------
state 4:

\[ X : X . a X b \]
\[ X : X a X . b \]

state 5:

\[ X : X a X b . \]

state 6:

EOF accept
.
error
Question 9 Types [10 pts]

1. Show the typing derivation for the Tiger statement \( y := \text{if } f(r.a) \text{ then } 3 \text{ else } y + 2 \). You may assume that your initial environment \( (\Gamma_0) \) has the following mappings (in addition to the base Tiger environment):
\[
\Gamma_0(y) = \text{int} \\
\Gamma_0(r) = \text{Record}(a:\text{int}, b:\text{string}) \\
\Gamma_0(f) = \text{int} \rightarrow \text{int}
\]

2. If \( \text{bat} \sqsubseteq \text{animal} \) and \( \text{fruitbat} \sqsubseteq \text{bat} \), then which of the following types are a subtype of \( \text{bat} \rightarrow \text{bat} \) (circle all types that are \( \sqsubseteq \text{bat} \rightarrow \text{bat} \))?

| animal  \(
\rightarrow
\) animal | animal  \(
\rightarrow
\) bat | animal  \(
\rightarrow
\) fruitbat |
|----------------|----------------|----------------|
| bat  \(
\rightarrow
\) animal | bat  \(
\rightarrow
\) bat | bat  \(
\rightarrow
\) fruitbat |
| fruitbat  \(
\rightarrow
\) animal | fruitbat  \(
\rightarrow
\) bat | fruitbat  \(
\rightarrow
\) fruitbat |
3. Given the subtyping relationship in the previous part, which of the following types are subtypes of 

$$(\text{bat} \rightarrow \text{bat}) \rightarrow \text{bat}$$

Again, circle *all* that apply:

<table>
<thead>
<tr>
<th>(fruitbat→fruitbat)→ fruitbat</th>
<th>(fruitbat→fruitbat)→ bat</th>
<th>(fruitbat→fruitbat)→ animal</th>
</tr>
</thead>
<tbody>
<tr>
<td>(fruitbat→ bat) → fruitbat</td>
<td>(fruitbat→ bat) → bat</td>
<td>(fruitbat→ bat) → animal</td>
</tr>
<tr>
<td>(fruitbat→ animal) → fruitbat</td>
<td>(fruitbat→ animal) → bat</td>
<td>(fruitbat→ animal) → animal</td>
</tr>
<tr>
<td>(bat→fruitbat)→ fruitbat</td>
<td>(bat→fruitbat)→ bat</td>
<td>(bat→fruitbat)→ animal</td>
</tr>
<tr>
<td>(bat→ bat) → fruitbat</td>
<td>(bat→ bat) → bat</td>
<td>(bat→ bat) → animal</td>
</tr>
<tr>
<td>(bat→ animal) → fruitbat</td>
<td>(bat→ animal) → bat</td>
<td>(bat→ animal) → animal</td>
</tr>
<tr>
<td>(animal→fruitbat)→ fruitbat</td>
<td>(animal→fruitbat)→ bat</td>
<td>(animal→fruitbat)→ animal</td>
</tr>
<tr>
<td>(animal→ bat) → fruitbat</td>
<td>(animal→ bat) → bat</td>
<td>(animal→ bat) → animal</td>
</tr>
<tr>
<td>(animal→ animal) → fruitbat</td>
<td>(animal→ animal) → bat</td>
<td>(animal→ animal) → animal</td>
</tr>
</tbody>
</table>