ECE 553: Compilers
Midterm

Name: NetID:

There are 8 questions, with the point values as shown below. You have 75 minutes with a total of 100 points. Pace yourself accordingly.

This exam must be individual work. You may not collaborate with your fellow students. However, this exam is open notes, so you may use your class notes.

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>10</td>
</tr>
<tr>
<td>2</td>
<td>Regular Languages</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>NFA to DFA</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>NFA to Regexp</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>Context Free Grammars</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>LR Parsing</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>Types</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>Type Inference</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td></td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Percent</td>
<td></td>
<td>100</td>
</tr>
</tbody>
</table>
Question 1 SML [10 pts]

1. Write \texttt{map} (curried or taking a tuple—either is fine) using \texttt{foldr}.

2. If you had written your \texttt{map} function exactly the same, but used \texttt{foldl} instead, what would have been wrong with the result?
3. Suppose I had the following:

```haskell
functor X(K: ORD_KEY) =
  struct
    structure S = SplaySetFn(K)
    structure M = SplayMapFn(K)

    fun invert m =

  end
```

Write the code for the `invert` function, which takes has type `K.ord_key M.map -> S.set M.map`. This function should create the inverse map of the map passed in. If the input map contains `1 \mapsto 2, 3 \mapsto 4, 5 \mapsto 2`, the output map should contain `2 \mapsto 1, 5, 4 \mapsto 3`—that is, if you look up a key in the output map, you should get the set of keys in the input map which map (in the input map) back to that value. You may find it useful to recall that `ORD_MAP` has:

- `val empty : 'a map`
- `val insert : ('a map * Key.ord_key * 'a) -> 'a map`
- `val find : ('a map * Key.ord_key) -> 'a option`
- `val foldli : ((Key.ord_key * 'a * 'b) -> 'b) -> 'b -> 'a map -> 'b`

and that `ORD_SET` has:

- `val empty : set`
- `val add : (set * item) -> set`
Question 2 Regular Languages [10 pts]

1. Which of the following strings are in the language of the regexp \((a^*(b|c)d)^*e?\) (Circle all that are in the language)

<table>
<thead>
<tr>
<th></th>
<th>e</th>
<th>ae</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>ade</td>
<td>abdaacde</td>
</tr>
<tr>
<td>b</td>
<td>abdbd</td>
<td>dce</td>
</tr>
</tbody>
</table>

2. Write a regular expression for all strings of 0s and 1s where any occurrence of the sequence 111 is followed by at least two 0s.

3. Write a regular expression for all strings of xs and ys where an odd number of xs must be followed by an even number of ys (an even number of xs may be followed by any number of ys).
Question 3 NFA to DFA [10 pts]
Convert the following NFA to a DFA.
Question 4 NFA to Regexp [10 pts]
Convert the following NFA to a Regular Expression. Showing your work is encouraged, as it makes it easier to give partial credit if you go wrong. You have the next page for this problem too:
(This page is for the NFA to Regexp Problem)
Question 5 Context Free Grammars [15 pts]

1. Which of the following strings are in the language of the following CFG? (Circle all that are in the language)

\[
S \rightarrow S \ b \ A \\
| \ e \\
A \rightarrow a \ A \ x \\
| \ B \\
B \rightarrow b \ B \\
| \ /* \ \epsilon \ */
\]

<table>
<thead>
<tr>
<th></th>
<th>e</th>
<th>ebaxabbbx</th>
</tr>
</thead>
<tbody>
<tr>
<td>eb</td>
<td>ebbbbb</td>
<td>ebxeebexe</td>
</tr>
</tbody>
</table>

2. Suppose you have terminals of id num [ ], = { and } and a language for describing values (similar to, but not the same as JSON). In this language, a value is either a number, an object, or an array. An object is represented by curly braces around a comma separated list of id = values. An array is a comma separated list of values. Examples (one per line) of valid strings in this language include:

- num
- {id=num, id={id=num}, id=[num, {id=num, id=num}]}
- [num, num, {id=num}]
- {}
- [{}, num, {id={id=num}, id=num}, id=[num,num]]

Write the CFG for this language:
Question 6 LR Parsing [15 pts]

Draw the FSM for an LR parser for the following grammar:

\[
\begin{align*}
S & \rightarrow X \ $ \\
X & \rightarrow a \ X \ b \\
& \quad | \ c \ Y \\
Y & \rightarrow y \ Y \\
& \quad | \ z
\end{align*}
\]
1. Suppose \( \Gamma = \{ f \mapsto \text{string} \rightarrow \text{int}, x \mapsto \text{int}, y \mapsto \text{string}, z \mapsto \text{Record(a:int,b:int)} \} \). Draw the typing derivation for

\[
\text{if } f(y) < 4 \text{ then } x + 2 \text{ else } z.a
\]

2. Suppose \( \text{Lion} \sqsubseteq \text{Cat} \sqsubseteq \text{Animal} \). Now, suppose I have a \text{CatInputStream} (that is, an input stream that reads Cats). Fill in the blanks below with either \( \sqsubseteq \) or \( \sqsupseteq \) to correctly describe the subtype relationship between \text{CatInputStream}, \text{AnimalInputStream}, and \text{LionInputStream}:

\[
\text{AnimalInputStream} \quad \quad \text{CatInputStream} \quad \quad \text{LionInputStream}
\]

3. What would the relationship be for \text{OutputStreams} (which write Animals/Cats/Lions)?

\[
\text{AnimalOutputStream} \quad \quad \text{CatOutputStream} \quad \quad \text{LionOutputStream}
\]
Question 8 Type Inference [15 pts]

1. Perform type inference on the following function. You MUST show your work for full credit.

```haskell
fun f x = case x of
    (g, [a]) => g a
    | (g, a::l) => g(f(g,l))
```

(More on next page)
Compiling the previous code gives “Warning: match nonexhaustive”. The compiler implicitly inserts `raise Match` for the missing cases, resulting in the following being the code that was actually type checked:

```sml
fun f x = case x of
    (g, [a]) => g a
  | (g, a::l) => g(f(g,l))
  | (g, []) => raise Match
```

2. What type does the SML compiler (which does NOT have subtyping) give the expression `raise Match`?

3. If your Tiger compiler (which does have subtyping, but does NOT have parametric polymorphism) supported exceptions, what type would `raise Match` have?