UNIVERSITY OF TORONTO
Department of Electrical and Computer Engineering

CSC467– Compilers and Interpreters
Midterm Examination Solution, Fall 2018

Course Instructor: Xu Zhao

2018/10/17

Name: _____

UTOR ID: _____

Student Number: _____

This exam contains **??** pages (including this cover page) and **??** questions. Total of points is **??**. Good luck!

Distribution of Marks

Run LaTeX again to produce the table

1. **Multiple choices. There are one or more correct choices for each question. Mark all of them. You will lose marks for wrong choice.**

   (a) (2 points) Which of the followings are valid MiniGLSL identifier tokens?

   A. 0abc
   B. if
   C. vec2
   D. temp

   Solution: D.

   (b) (2 points) Suppose you are going to remove the commented part of the following MiniGLSL snippet:

   ```
   int nest = /*/*/1*/**/0;
   ```

   Which of the followings are the correct results of removing the commented part of the code snippet above? Suppose we do NOT support nested comments.

   A. `int nest = 1*0;`
   B. `int nest = 1;`
   C. `int nest = *0;`
   D. None of the above

   Solution: A.

   (c) (2 points) Which of the followings are valid integer tokens in the MiniGLSL language? Suppose octal numbers are not supported.

   A. `0.0`
   B. `042`
   C. `42`
   D. `-042`

   Solution: C.

   (d) (2 points) Which of the following code snippets' error needs to be reported in the lexical analysis? Suppose octal numbers are not supported.

   A. `int i = (2));`
   B. `floa j = 0.0;`
   C. `int k = 012;`
   D. `bool p = 42;`

   Solution: C.

   (e) (2 points) Choose the steps of compilation that are in GCC but not in MiniGLSL.

A. Lexical analysis
B. Parsing
C. Semantic analysis
D. Optimization
E. Code generation

Solution: D.

2. (10 points) **Lexical Analysis**
   Consider the following flex-style lexical specification:

```
%%
a*b         { printf("1"); }
caa         { printf("2"); }
a*ca*       { printf("3"); }
```

Given the following input string:

```
abcaacacaaab
```

what is the output of the lexer?

Solution: 12331

3. **Regular Expression and CFG**

   (a) (10 points) Write a regular expression for the set of all strings over the alphabet of `{a, b}` that matches all strings that do not contain two (or more) `a` or `b` consecutively, including the empty string. Examples:

   - In the set of strings: `ab`, `ba`, `aba`
   - Not in the set of strings: `aab`, `abb`

   Solution:

   ```
   (ab)*|(ba)*|a(ba)*|b(ab)*
   ```

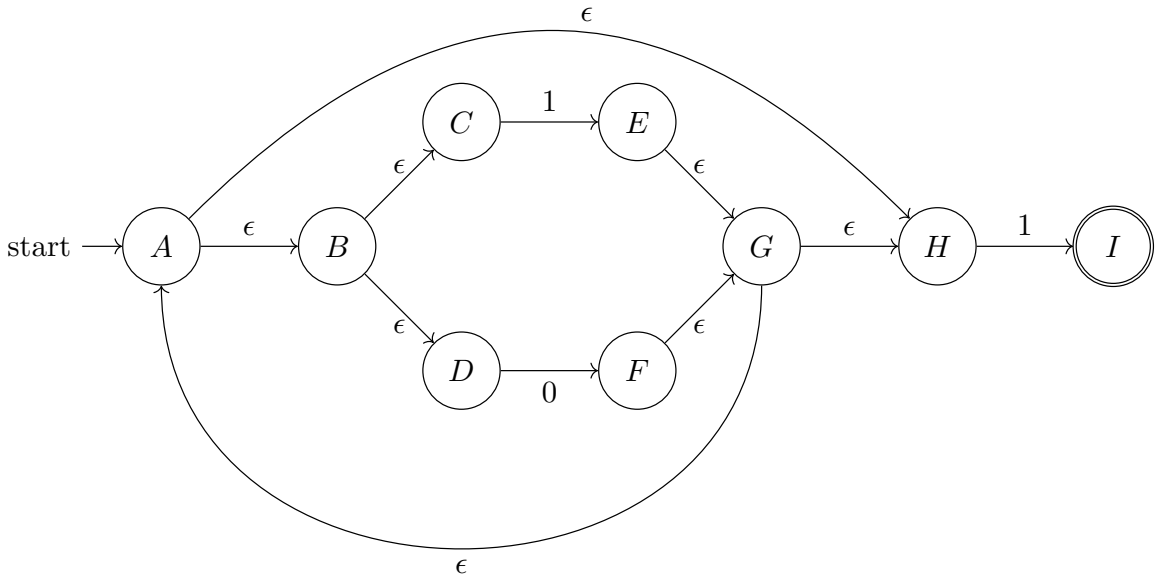
   (b) (10 points) Write a context-free grammar for the set of all strings over the alphabet of `{(, )}` that are either empty string, or matched parenthesizes.

   For example, both "(())" and "()()" belong to the language of the grammar, but ")(()" does not. Please use `S` as the starting symbol of your grammar.

   Solution:

   $$\begin{aligned} \mathrm{S} &\to \epsilon \\ \mathrm{S} &\to SS \\ \mathrm{S} &\to (S) \end{aligned}$$

4. **DFA and NFA**
Consider the following diagram of an NFA.



(a) (5 points) Write down the equivalent regular expression of this NFA.

Solution:

```
(0|1)*1
```

(b) (10 points) Fill in the following table about ϵ-closures of the NFA.
Solution:

ϵ-CLOSURE of A	A, B, C, D, H
ϵ-CLOSURE of B	B, C, D
ϵ-CLOSURE of G	A, B, C, D, G H
ϵ-CLOSURE of E	A, B, C, D, E, G, H

The (c) part of this question is on the next page.

(c) (15 points) Draw the diagram of the equivalent DFA. Please name the DFA states with the corresponding states in the NFA.

Solution:

```
Start state: ABCDH
Accepting state: ABCDEGHI
ABCDH to ABCDEGHI on 1
ABCDH to FGHABCD on 0
ABCDEGHI self-loop on 1
FGHABCD self-loop on 0
FGHABCD to ABCDEGHI on 1
ABCDEGHI to FGHABCD on 0
```

5. **Predictive Parsing**
Suppose you are going to write a LL(1) parser to parse the following context-free grammar. Note that the starting symbol is `S`.
Non-terminals are `{S, A, B, C, D}`, terminals are {id, +, *, (,)}.
$ (dollar sign) represents the end of the input and ϵ represents the empty string.

$$
\begin{aligned}
\texttt{S} &\rightarrow \texttt{A B} \\
\texttt{A} &\rightarrow \texttt{C D} \mid \epsilon \\
\texttt{B} &\rightarrow \texttt{+ A B} \mid \epsilon \\
\texttt{C} &\rightarrow \texttt{(S)} \mid \text{id} \\
\texttt{D} &\rightarrow \texttt{* C D} \mid \epsilon
\end{aligned}
$$

(a) (10 points) Fill in the following `FIRST` and `FOLLOW` set table.

FIRST(S)	(, id, +, ϵ
FOLLOW(S)	$,)
FIRST(A)	(, id, ϵ
FOLLOW(A)	$,), +
FIRST(B)	+, ϵ
FOLLOW(B)	$,)
FIRST(C)	(, id
FOLLOW(C)	$,), +, *
FIRST(D)	*, ϵ
FOLLOW(D)	$,), +

(b) (10 points) Fill in the blanks of the LL(1) parsing table. If there are any conflicting entries, use circle to mark them. You only need to fill in the body part of the rule in each cell. For example, for the rule `S`→ `A B`, you only need to fill in "`A B`".

	id	*	+	(	)	$
S	AB		AB	AB	AB	AB
A	CD		ϵ	CD	ϵ	ϵ
B			+AB		ϵ	ϵ
C	id			(S)		
D		*CD	ϵ		ϵ	ϵ

(c) (10 points) Does the following string

```
id + id * id
```

belongs to the language of this grammar? If so, write down a left-most derivation.

Solution:

$$
\begin{aligned}
S &\rightarrow AB \\
&\rightarrow CDB \\
&\rightarrow idDB \\
&\rightarrow idB \\
&\rightarrow id + AB \\
&\rightarrow id + CDB \\
&\rightarrow id + idDB \\
&\rightarrow id + id * CDB \\
&\rightarrow id + id * idDB \\
&\rightarrow id + id * idB \\
&\rightarrow id + id * id
\end{aligned}
$$

This page is intentionally left blank to accommodate work that wouldn't fit elsewhere and/or scratch work.

This page is intentionally left blank to accommodate work that wouldn't fit elsewhere and/or scratch work.