EC3204: Programming Languages and Compilers (Fall 2023)
Mid-term Exam

100 points in total, 25% of the total score

Date and Time: 10/25, 10:30 – 11:45
Place: Oryong Hall 203 (오룡관 203호)

Student ID: ____________________________

Name: _________________________________

* Leave the score table blank

	Min Scores	Max Scores	Your Scores
Problem 1	0	10	
Problem 2	0	10	
Problem 3	0	10	
Problem 4	0	10	
Problem 5	0	10	
Problem 6	-50	50	
Total	0	100	

Problem 1. (10pt) NFA to DFA

Suppose we have an NFA, represented by the lower-left transition table, for the regular expression $((a \cdot b) \mid c)^*$. In the table, 0 is an initial state and 9 is an accepting state.

State	ϵ	a	b	c
0	$\{1,9\}$	$\emptyset$	$\emptyset$	$\emptyset$
1	$\{2,6\}$	$\emptyset$	$\emptyset$	$\emptyset$
2	$\emptyset$	$\{3\}$	$\emptyset$	$\emptyset$
3	$\{4\}$	$\emptyset$	$\emptyset$	$\emptyset$
4	$\emptyset$	$\emptyset$	$\{5\}$	$\emptyset$
5	$\{8\}$	$\emptyset$	$\emptyset$	$\emptyset$
6	$\emptyset$	$\emptyset$	$\emptyset$	$\{7\}$
7	$\{8\}$	$\emptyset$	$\emptyset$	$\emptyset$
8	$\{1,9\}$	$\emptyset$	$\emptyset$	$\emptyset$
9	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

$T = \emptyset$
repeat
  $T' = T$
  $T = T' \cup F(T')$
until $T = T'$
return T

Given a set of states I, ϵ-closure$(\{I\})$ can be obtained by the upper-right fixed point iteration algorithm, where $F(X) = I \cup \bigcup_{s \in X} \delta(s, \epsilon)$ and δ is a transition function.

1. (5pt) Describe the computation process of ϵ-closure$(\{7\})$ according to the fixed point algorithm above.

Iteration	T'	T
1		
2		
3		
4		
5		

2. (5pt) Complete the DFA as a transition table following the subset construction algorithm. Explicitly state the initial state and the final states. Omit the transitions from the dead state.

State	a	b	c

Problem 2. (10pt) Ambiguity

For each sub-problem here, no partial points will be given for incorrect answers or justification.

1. (3pt) Consider the grammar below. Is the grammar ambiguous? Justify your answer.

$$E \to E + T \mid T, \qquad T \to T * F \mid F, \qquad F \to id \mid (E)$$

2. (3pt) Consider the grammar below. Is the grammar ambiguous? Justify your answer.

$$S \to \epsilon \mid (S) \mid SS$$

3. (4pt) Consider the grammar below. Can the grammar be parsed by an LL(1) parser? Justify your answer.

$$S \to aSbS \mid bS \mid \epsilon$$

Problem 3. (10pt) Top-Down Parsing

Consider the following grammar where the start variable is S and the terminal symbols are $\{x, y, z\}$.

$$S \to AS \mid BC \mid CA, \qquad A \to x, \qquad B \to y \mid \epsilon, \qquad C \to z$$

1. (2pt) List the *First* and *Follow* sets for the above grammar.

$$\begin{aligned} First(S) &= \{ \qquad\qquad\qquad \}, & Follow(S) &= \{ \qquad\qquad\qquad \} \\ First(A) &= \{ \qquad\qquad\qquad \}, & Follow(A) &= \{ \qquad\qquad\qquad \} \\ First(B) &= \{ \qquad\qquad\qquad \}, & Follow(B) &= \{ \qquad\qquad\qquad \} \\ First(C) &= \{ \qquad\qquad\qquad \}, & Follow(C) &= \{ \qquad\qquad\qquad \} \end{aligned}$$

2. (2pt) Complete the LL(1) parsing table for the grammar.

	x	y	z	$
S				
A				
B				
C				

3. (3pt) Is the grammar in LL(1)? Justify your answer.

4. (3pt) Complete the LL(1) parsing sequence for the input string xyz. Extend the template below if necessary.

Stack	**Input**	**Action**
$S\$$	$xyz\$$	

Problem 4. (10pt) Bottom-Up Parsing

This problem aims to evaluate your understanding of the SLR parsing process. Consider the following expression grammar.

$$
\begin{array}{llllllllllll}
(1) & E & \rightarrow & E+T & (2) & E & \rightarrow & T & (3) & T & \rightarrow & T*F \\
(4) & T & \rightarrow & F & (5) & F & \rightarrow & (E) & (6) & F & \rightarrow & \textbf{id}
\end{array}
$$

The SLR parsing table for the grammar is the following.

State	**id**	+	*	(	)	$	E	T	F
0	$s5$			$s4$			$g1$	$g2$	$g3$
1		$s6$				acc			
2		$r2$	$s7$		$r2$	$r2$			
3		$r4$	$r4$		$r4$	$r4$			
4	$s5$			$s4$			$g8$	$g2$	$g3$
5		$r6$	$r6$		$r6$	$r6$			
6	$s5$			$s4$				$g9$	$g3$
7	$s5$			$s4$					$g10$
8		$s6$			$s11$				
9		$r1$	$s7$		$r1$	$r1$			
10		$r3$	$r3$		$r3$	$r3$			
11		$r5$	$r5$		$r5$	$r5$			

Complete the SLR parsing action sequence for the input string **id**+**id**∗**id**+**id**. Extend the template below if necessary.

Stack	**Symbols**	**Input**	**Action**
0		$\textbf{id} + \textbf{id} * \textbf{id} + \textbf{id}\$$	shift to 5
0 5	**id**	$+\textbf{id} * \textbf{id} + \textbf{id}\$$	reduce by 6 ($F \rightarrow \textbf{id}$)

Problem 5. (10pt) Necessity of Semantic Analysis

Why do we need semantic analyses in compilers? Explain their necessity with a concrete example program that is syntactically valid but semantically ill-formed. Your example program should contain semantic errors other than type errors; that is, your example should not be similar to the one in the slides from Lecture 12. Examples of acceptable semantic errors include, but are not limited to: division-by-zero, integer overflow, buffer overflow, null-dereference, memory-leak, etc.

Problem 6. (50pt) O/X Questions

This problem aims to evaluate your overall understanding of important concepts in this course. You will get 2 points for each correct answer. You will lose 2 points for each wrong answer.

(1) An OCaml compiler can be implemented using the OCaml programming language. (O, X)

(2) The following OCaml program is lexically valid. (O, X)

```
let rec f a b = a + b
```

(3) $\emptyset^* = \emptyset$. (O, X)

(4) $R^* = (R^*)^*$. (O, X)

(5) The string recognition of DFA may not terminate for some finite input strings. (O, X)

(6) Given a regular expression R, there exists only a single DFA that can recognize the language defined by R. (O, X)

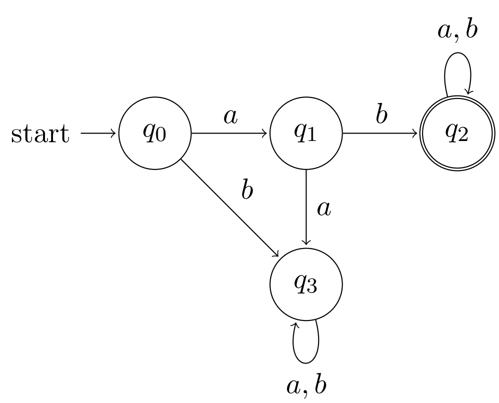
(7) Regarding NFA (non-deterministic finite automaton), the term "non-deterministic" indicates that the results of the string recognition are non-deterministic. (O, X)

(8) A transition function of an NFA is a partial function. (O, X)

(9) Following Thompson's construction without any modifications, the resulting NFAs will always have a single final state (O, X).

(10) Every DFA can be converted into NFA. (O, X)

(11) In the DFA below, the state q_3 is responsible for recognizing the strings that start with b (O, X).



(12) The language of a context-free grammar is the set of all sentential forms. (O, X)

(13) The following OCaml program is lexically valid but syntactically invalid. (O, X)

```
let f a b = (match a with [] -> b | h::t -> t + b)
```

(14) There is a language that is regular but not context-free. (O, X)

(15) Some context-free languages can be recognized by DFA. (O, X)

(16) Every lexical pattern of C programs can be expressed by context-free grammars. (O, X)

(17) Every syntactically valid Python program can be expressed by regular expressions. (O, X)

(18) There is a parse tree that has multiple left-most derivations. (O, X)

(19) The grammar below is in LL(1). (O, X)

$$S \rightarrow iEtS \mid iEtSeS \mid a$$

(20) Top-down parsers cannot parse all strings defined by a left-recursive grammar. (O, X)

(21) In Problem 3, $First(BC) = \{y, z\}$. (O, X)

(22) Bottom-up parsers can handle left-recursive grammars, because bottom-up parsers scan input strings from left to right. (O, X)

(23) In bottom-up parsing, the reduce action always occurs at the rightmost substring. (O, X)

(24) Some LR(1) grammars can be parsed by LL(1) parsers. (O, X)

(25) An SLR parsing process may not terminate for some input strings. (O, X)