Summary in Graph

Exam Summary (GO Classes Test Series 2024 | Compiler Design | Test 1)

Qs. Attempted:	**14** (5 + 9)	**Correct Marks:**	**15** (3 + 12)
Correct Attempts:	**9** (3 + 6)	**Penalty Marks:**	**1** (0.33 + 0.67)
Incorrect Attempts:	**5** (2 + 3)	**Resultant Marks:**	**14** (2.66 + 11.33)

Total Questions:	**15** (5 + 10)
Total Marks:	**25** (5 + 20)
Exam Duration:	**45 Minutes**
Time Taken:	**36 Minutes**

EXAM RESPONSE | EXAM STATS | FEEDBACK

Technical

Q #1 | Multiple Choice Type | Award: 1 | Penalty: 0.33 | Compiler Design

Which of the following is true about the relationship between $\mathrm{LALR}(1)$ and $\mathrm{SLR}(1)$ parsers?

A. There are grammars that can be parsed by $\mathrm{SLR}(1)$ parsers that cannot be parsed by $\mathrm{LALR}(1)$ parsers.
B. There are grammars that can be parsed by $\mathrm{LALR}(1)$ parsers that cannot be parsed by $\mathrm{SLR}(1)$ parsers.
C. $\mathrm{LALR}(1)$ parsers are as powerful as $\mathrm{SLR}(1)$ parsers, so any grammar that can be parsed by an $\mathrm{LALR}(1)$ parser can be parsed by a $\mathrm{SLR}(1)$ parser.
D. For any grammar G which can be parsed by a $\mathrm{SLR}(1)$ parser, number of states in the $\mathrm{SLR}(1)$ parser can be less than number of states in the $\mathrm{LALR}(1)$ parser for G.

Your Answer: B | **Correct Answer: B** | Correct | Discuss

Q #2 | Multiple Choice Type | Award: 1 | Penalty: 0.33 | Compiler Design

Which of the following is the name of the data structure in a compiler that is responsible for managing information about variables and their attributes?

A. Abstract Syntax Tree (AST)

B. Attribute Grammar
C. Symbol Table
D. Parse Table

Your Answer: A | Correct Answer: C | Incorrect | Discuss

Q #3 | Multiple Select Type | Award: 1 | Penalty: 0 | Compiler Design

Which of the following is true about grammar ambiguity?

A. A grammar is said to be ambiguous if it can produce more than one parse tree for some sentence/string.
B. A grammar is said to be ambiguous when, for some sentence, two different sequences of leftmost derivations can produce the same sentence from the same start symbol.
C. A grammar is ambiguous if there is some string w such that w's right-most derivation differs from its left-most derivation.
D. A grammar is said to be ambiguous if and only if, for some sentence, two different sequences of rightmost derivations can produce the same sentence from the same start symbol.

Your Answer: A | Correct Answer: A;B;D | Incorrect | Discuss

Q #4 | Multiple Choice Type | Award: 1 | Penalty: 0.33 | Compiler Design

In SLR parsing to get a shift-reduce conflict for state I on terminal symbol ' a ',

A. $\mathrm{A} \rightarrow \alpha . \beta$ with $\mathrm{First}\ (\beta)$ containing ' a ' should be in I
B. $\mathrm{A} \rightarrow \delta .$ be in I with $\mathrm{Follow}(\mathrm{A})$ having ' a '
C. $\mathrm{A} \rightarrow \alpha . a \beta$ should be in I and $\mathrm{X} \rightarrow \delta .$ be in I with $\mathrm{Follow}\ (\mathrm{X})$ having ' a '
D. None of the other options

Your Answer: C | Correct Answer: C | Correct | Discuss

Q #5 | Multiple Select Type | Award: 1 | Penalty: 0 | Compiler Design

Consider the following grammar:
$\mathrm{A} \rightarrow \mathrm{A}$ and $\mathrm{A} \mid \mathrm{A}$ or $\mathrm{A} \mid (\mathrm{A}) \mid$ true $\mid$ false
Which of the following is/are true about First sets for each production and nonterminal?

A. $\mathrm{FIRST}(\mathrm{true}) = \{$ “true” $\}$
B. $\mathrm{FIRST}(\ \mathrm{false}\ ) = \{$ “false” $\}$
C. $\mathrm{FIRST}((\mathrm{A}))=\{$“(”$\}$
D. $\mathrm{FIRST}(\ \mathrm{A\ and\ A}\ ) = \mathrm{FIRST}(\mathrm{A\ or\ A}\ ) = \mathrm{FIRST}(\mathrm{A}\ ) = \{$“true”, “false” $\}$

Your Answer: A;B;C | Correct Answer: A;B;C | Correct | Discuss

Q #6 | Multiple Choice Type | Award: 2 | Penalty: 0.67 | Compiler Design

In LR parsing, If the parser is in a state containing an $\mathrm{LR}(k)$ item $[\mathrm{A} \rightarrow \alpha \bullet \gamma, \delta]$, what must be at the top of the stack?

A. α

B. γ
C. δ
D. A

Your Answer: A **Correct Answer: A** Correct Discuss

Q #7 Multiple Choice Type Award: 2 Penalty: 0.67 Compiler Design

In LR parsing, Given an $\mathrm{LR}(k)$ item $[\mathrm{A} \rightarrow \alpha \bullet \gamma, \delta]$, what lookahead allows a shift to be performed (ignoring any other $\mathrm{LR}(k)$ items added to the state by closure)?

A. The terminal at the beginning of γ.
B. The terminal at the ending of α.
C. The terminal at the beginning of α.
D. The terminal at the beginning of δ.

Your Answer: **Correct Answer: A** Not Attempted Discuss

Q #8 Multiple Choice Type Award: 2 Penalty: 0.67 Compiler Design

In $\mathrm{LR}(1)$ parsing, what constitutes a shift/reduce conflict?

A. A DFA state contains both LR items allowing shift and reduce for the different lookahead.
B. A DFA state contains both LR items allowing shift and reduce for the same lookahead.
C. A DFA state contains both LR items allowing shifts for the same lookahead.
D. A DFA state contains both LR items allowing shifts for the different lookahead.

Your Answer: B **Correct Answer: B** Correct Discuss

Q #9 Multiple Choice Type Award: 2 Penalty: 0.67 Compiler Design

Consider the following sets of $\mathrm{LR}(1)$ items in the states of a $\mathrm{LR}(1)$ parser.

$$\begin{array}{ll} \text{State } 0: & \text{State } 2: \\ [\mathrm{A} \rightarrow \bullet \mathrm{a}, \mathrm{b}] & [\mathrm{A} \rightarrow \bullet \mathrm{a}, \mathrm{c}] \\ [\mathrm{A} \rightarrow \mathrm{a} \bullet, \mathrm{c}] & [\mathrm{A} \rightarrow \mathrm{a} \bullet, \mathrm{b}] \\ [\mathrm{B} \rightarrow \mathrm{a} \bullet, \mathrm{b}] & [\mathrm{B} \rightarrow \mathrm{a} \bullet, \mathrm{a}] \end{array}$$

$$\begin{array}{ll} \text{State } 1: & \text{State } 3: \\ [\mathrm{A} \rightarrow \bullet \mathrm{a}, \mathrm{a}] & [\mathrm{A} \rightarrow \bullet \mathrm{a}, \mathrm{b}] \\ [\mathrm{A} \rightarrow \bullet \mathrm{a}, \mathrm{b}] & [\mathrm{B} \rightarrow \bullet \mathrm{a}, \mathrm{b}] \\ [\mathrm{B} \rightarrow \mathrm{a} \bullet, \mathrm{b}] & \end{array}$$

What states would be merged in a $\mathrm{LALR}(1)$ parser?

A. States 1 and 2.
B. States 0 and 1.
C. States 0 and 2.
D. States 2 and 3.

Your Answer: C | Correct Answer: C | Correct | Discuss

Q #10 | Multiple Choice Type | Award: 2 | Penalty: 0.67 | Compiler Design

Consider the following sets of $\mathrm{LR}(1)$ items in the states of a $\mathrm{LR}(1)$ parser.

$$\begin{array}{ll} \text{State 0:} & \text{State 2:} \\ [\mathrm{A} \rightarrow \bullet\mathrm{a}, \mathrm{b}] & [\mathrm{A} \rightarrow \bullet\mathrm{a}, \mathrm{c}] \\ [\mathrm{A} \rightarrow \mathrm{a}\bullet, \mathrm{c}] & [\mathrm{A} \rightarrow \mathrm{a}\bullet, \mathrm{b}] \\ [\mathrm{B} \rightarrow \mathrm{a}\bullet, \mathrm{b}] & [\mathrm{B} \rightarrow \mathrm{a}\bullet, \mathrm{a}] \\ \text{State 1 :} & \text{State 3 :} \\ [\mathrm{A} \rightarrow \bullet\mathrm{a}, \mathrm{a}] & [\mathrm{A} \rightarrow \bullet\mathrm{a}, \mathrm{b}] \\ [\mathrm{A} \rightarrow \bullet\mathrm{a}, \mathrm{b}] & [\mathrm{B} \rightarrow \bullet\mathrm{a}, \mathrm{b}] \\ [\mathrm{B} \rightarrow \mathrm{a}\bullet, \mathrm{b}] & \end{array}$$

What happens when we merge states(which can be merged according to $\mathrm{LALR}(1)$ parser) in a $\mathrm{LALR}(1)$ parser?

A. A new reduce/reduce conflicts introduced in the $\mathrm{LALR}(1)$ parser
B. A new shift/reduce conflicts introduced in the $\mathrm{LALR}(1)$ parser
C. New reduce/reduce and shift/reduce conflicts introduced in the $\mathrm{LALR}(1)$ parser
D. No new conflicts introduced in the $\mathrm{LALR}(1)$ parser

Your Answer: C | Correct Answer: A | Incorrect | Discuss

Q #11 | Multiple Choice Type | Award: 2 | Penalty: 0.67 | Compiler Design

Consider the following sets of $\mathrm{LR}(1)$ items in the states of a $\mathrm{LR}(1)$ parser for some grammar G.

$$\begin{array}{ll} \text{State 0:} & \text{State 2:} \\ [\mathrm{A} \rightarrow \bullet\mathrm{a}, \mathrm{b}] & [\mathrm{A} \rightarrow \bullet\mathrm{a}, \mathrm{c}] \\ [\mathrm{A} \rightarrow \mathrm{a}\bullet, \mathrm{c}] & [\mathrm{A} \rightarrow \mathrm{a}\bullet, \mathrm{b}] \\ [\mathrm{B} \rightarrow \mathrm{a}\bullet, \mathrm{b}] & [\mathrm{B} \rightarrow \mathrm{a}\bullet, \mathrm{a}] \\ \text{State 1:} & \text{State 3 :} \\ [\mathrm{A} \rightarrow \bullet\mathrm{a}, \mathrm{a}] & [\mathrm{A} \rightarrow \bullet\mathrm{a}, \mathrm{b}] \\ [\mathrm{A} \rightarrow \bullet\mathrm{a}, \mathrm{b}] & [\mathrm{B} \rightarrow \bullet\mathrm{a}, \mathrm{b}] \\ [\mathrm{B} \rightarrow \mathrm{a}\bullet, \mathrm{b}] & \end{array}$$

Which of the following is true for the given set of $\mathrm{LR1}$ items and the given $\mathrm{LR1}$ parser (Assume the parser has only the given set of $\mathrm{LR1}$ items for G)?

A. G is $\mathrm{LR1}$.
B. G is not $\mathrm{LR1}$ because reduce/reduce conflict is caused by $[\mathrm{A} \rightarrow \mathrm{a}\bullet, \mathrm{c}]$ and $[\mathrm{B} \rightarrow \mathrm{a}\bullet, \mathrm{b}]$ in state 1.
C. G is not $\mathrm{LR1}$ because Shift/reduce conflict is caused by $[\mathrm{A} \rightarrow \bullet\mathrm{a}, \mathrm{c}]$ and $[\mathrm{B} \rightarrow \mathrm{a}\bullet, \mathrm{a}]$ on lookahead a in state 2.
D. G is not $\mathrm{LR1}$ because Shift/shift conflict is caused by $[\mathrm{A} \rightarrow \bullet\mathrm{a}, \mathrm{b}]$ and $[\mathrm{B} \rightarrow \bullet\mathrm{a}, \mathrm{b}]$ on lookahead a in state 3.

Your Answer: C | Correct Answer: C | Correct | Discuss

Q #12 Numerical Type | Award: 2 | Penalty: 0 | Compiler Design

Consider the following grammar:

$$\mathrm{S} \rightarrow \mathrm{S} \; ; \; \mathrm{a} \mid \mathrm{a}$$

Number of states in the canonical set of $\mathrm{SLR}(1)$ items for the grammar? (Do not include final accept state $(\mathrm{S'} \rightarrow \mathrm{S} \; \$. \;)$ in counting.

Your Answer: 5 | **Correct Answer: 5** | Correct | Discuss

Q #13 Numerical Type | Award: 2 | Penalty: 0 | Compiler Design

Consider the following grammar:

$$\mathrm{S} \rightarrow \mathrm{abS} \mid \mathrm{acS} \mid \mathrm{c}$$

Number of states in the canonical set of $\mathrm{LR}(0)$ items for the grammar? (Do not include final accept state $(\mathrm{S'} \rightarrow \mathrm{S} \; \$. \;)$ in counting.

Your Answer: 6 | **Correct Answer: 8** | Incorrect | Discuss

Q #14 Multiple Select Type | Award: 2 | Penalty: 0 | Compiler Design

We have a scanner (lexical analyzer) "flex". Consider the following tokens and their associated regular expressions, given as a flex scanner specification :

Regular expression	Token
$(01 \mid 10)$	printf(“ X ”)
$0(01)^*1$	printf(“ Y ”)
$(1010^*1 \mid 0101^*0)$	printf(“ Z ”)

When using regular expressions to scan an input, we resolve conflicts by taking the largest possible match at any point. If two longest matches are of same length then we give preference in the below order : $\mathrm{X} > \mathrm{Y} > \mathrm{Z}$

For Which of the following inputs to this scanner, we will get an Lexical error?

A. 0101
B. 010101
C. 01010101
D. 0101010101

Your Answer: B;C | **Correct Answer: B;C** | Correct | Discuss

Q #15 Multiple Select Type | Award: 2 | Penalty: 0 | Compiler Design

In lexical analyser, When using regular expressions to scan an input, for resolving conflicts that occur when using regular expressions to scan an input, Our approach was to use maximal-munch to always choose the longest possible match at any point, then to break ties based on the priorities of the regular expressions. However, this is not the only way that we could have resolved conflicts.
In some cases we may have a set of regular expressions for which it is possible to tokenize a particular input

string, but for which the maximal-munch algorithm will not be able to break the input into tokens.
Consider the following scanner (lexical analyzer) “flex”.
Consider the following tokens and their associated regular expressions, given as a flex scanner specification :

Regular expression	Token
aa	$\text{print}(\text{“1”})$
a	$\text{print}(\text{ “2”})$
ab	$\text{print}(\text{“3”})$

If two longest matches are of same length then we give preference in the below order : $1 > 2 > 3$
For which of the following input strings : The string can be broken apart into substrings, where each substring matches one of the regular expressions, but ● The maximal-munch algorithm will fail to break the string apart in a way where each piece matches one of the regular expressions.

A. $cabab$
B. aab
C. abb
D. aba

Your Answer: A;B;C **Correct Answer: B** Incorrect Discuss

You're doing good, you can target above 70 percentage!

Copyright & Stuff