COMP - Grammars and Parsers (MIEIC - Compilers - 2021)

119	1.8	Active
Responses	Average Score	Status

1. Consider the following token definitions and grammar (note that the '{' and '}' used in the grammar rules means 0 or more occurrences of the sequence of terminal and non-terminal symbols between brackets): TOKENS: INT= int IDENT= [a-z][0-9a-z]* VIRG=, PVIRG=; CONST=[0-9]+ IGUAL== MULT=* CFG A: Start -> {Decl} {AttribConst} {AttribExpr} Decl -> INT IDENT {VIRG IDENT} PVIRG AttribConst -> IDENT IGUAL CONST PVIRG AttribExpr -> IDENT IGUAL Expr PVIRG Expr -> IDENT MULT IDENT

2. Considering the input: int a, b, c; b=3; a=b*c; (a) What do you think would be the chain of tokens from the lexical analyzer (scanner)? (1 point)

   92% of respondents (107 of 116) answered this question correctly.

INT IDENT("a") VIRG IDENT("b...	107	✓
INT IDENT("a") IDENT("b") IDE...	5	
INT IDENT("a") IDENT("b") IDE...	0	
Other	4	

3. (b) Draw the CST (concrete syntax tree) resultant from the chain of tokens and using the CFG A and a possible AST (abstract syntax tree); (1 point)

102	Latest Responses
Responses	*"CST: Start Decl INT int IDENT a VIRG , IDENT b "*
	" CST Start Decl "

4. Let's start analyzing how to implement the CFG A as a recursive, predictive, top-down parser. What is the value needed for the lookahead? (1 point)

87% of respondents (104 of 119) answered this question correctly.

1	3
2	7
3	104 ✓
4	2
not fixed	3

5. Show the LL(1) parser table for CFG A and justify why the CFG A is not an LL(1) grammar. (1 point)

107
Responses

Latest Responses

"TOKENS: INT= int IDENT= [a-z][0-9a-z] VIRG=, PVIRG=; CONST=[0...*

"Table screenshot at group files. "

6. Present the modifications needed in order to have an LL(1) grammar. Let's name it as CFG B. (1 point)

97
Responses

Latest Responses

"CFG B: Start -> {Decl} (AttribConst)? Decl -> INT IDENT {VIRG IDENT}...

7. Show the LL(1) parser table for CFG B. Is CFG B LL(1)? Justify your answer. (1 point)

85
Responses

Latest Responses

"| | INT | IDENT | ----------...

8. Implement a syntactic analyzer for CFG B without considering the construction of the tree (use pseudocode inspired in Java code); (1 point)

83
Responses

Latest Responses

9. Implement a syntactic analyzer for the CFG A without considering the construction of the tree (use pseudocode inspired in Java code); (1 point)

89

Responses

Latest Responses

"Main() { Chain of tokens = Scanner(); // it can stop here! Token // first...

10. Show the JavaCC implementation of CFG A: (1 point)

74

Responses

Latest Responses

11. Show the JavaCC implementation of CFG B: (1 point)

70

Responses

Latest Responses