POSTAL Book Package 2021

Computer Science & IT

Objective Practice Sets

Compiler Design

Contents

MADE EASY Publications

Note: This book contains copyright subject matter to MADE EASY Publications, New Delhi. No part of this book may be reproduced, stored in a retrieval system or transmitted in any form or by any means. Violators are liable to be legally prosecuted.

2 CHAPTER Lexical Analysis

Q.1 The task of the lexical analysis phase is
(a) to parse the source program into the basic elements or tokens of the language
(b) to build a literal table and an identifier table
(c) to build a uniform symbol table
(d) all of the above

Q.2 The output of lexical analyser is
(a) a set of regular expressions
(b) syntax tree
(c) set of tokens
(d) strings of characters

Q.3 How many tokens are contained in the following FORRTAN statement:

```
IF(NUMB   EQ  MAX) GOTO 500
```

(a) 8 (b) 10
(c) 22 (d) 24

Q.4 The number of tokens in the following *C* statement

```
printf ("i = %d, & i = % x", i, & i);
```

(a) 3 (b) 26
(c) 10 (d) 21

Q.5 Consider the following statements:
1. Lexical analysis removes white spaces and not the comments.
2. for (i = 0) generates lexical error in *C*
3. Lexeme is the string and not the pattern.
4. *fi* ($a > b$) generates no lexical error in *C*.

Identify the false statements:
(a) Only 1 and 4 (b) Only 1 and 2
(c) Only 2 (d) Only 4

Q.6 Consider the following C-program:

```
int strange (int x)
{
  if (x < = 0) return 0;
  if (x%2! = 0) return x – 1;
  return 1 + strange (x – 1);
}
```

Find out the number of tokens?
(a) 44 (b) 42
(c) 43 (d) 75

Q.7 Consider line-3 of the following c-program.

```
int main ( ) {          /* line1*/
int i, n ;              /* line2*/
For (i = 0; i < n; i ++); /* line 3*/
```

Identify the compiler response about the line 3 while creating the object module.
(a) No compilation error
(b) Only a lexical error
(c) Only Syntactic error
(d) Both lexical error and syntactic error

Q.8 Find the line number in which lexical error present in the following program.

```
1. main( )
2. {
3.     int x; y; z;
4.     /* comment1*/
5.     /*com/*ment2*/end*/
6.     x = "A";
7. }
```

(a) 3 (b) 5
(c) 6 (d) No lexical error

Q.9 Consider the following program.

```
main ( )
{
    char ch = 'A';
    int x, y;
    x = y = 20;
    x ++;
    printf("%d% d", x, y);
}
```

The number of tokens in the above program are ________.

Q.10 Which of the following equivalent one is used in lexer?
(a) Regular expression
(b) Context free language
(c) Both (a) and (b)
(d) Neither (a) nor (b)

Q.11 In some programming languages, an identifier is permitted to be a letter followed by any number of letters or digits. If *L* and *D* denotes the sets of letters and digits respectively, which expression defines an identifier?
(a) $(LUD)^+$ (b) $L(LUD)^*$
(c) $(L.D)^*$ (d) $L.(L.D)^*$

Q.12 Consider the following C-prog. Find the number of tokens in the output of the following program if that is passed as a string to a lexical analyzer.

```
main ()
{
    char ∗ s[ ] = {“ice”, “green”, “water”, “hi”};
    char ∗∗ ptr[ ] = {s + 3, S + 2, S + 1, s}
    char ∗∗∗ p = ptr;
print f(“in, % s”, ∗∗ ++ P);
}
```

(a) 5 (b) 1
(c) 4 (d) 3

Q.13 Find the number of tokens in the following *C* code using lexical analyzer of compiler.

```
main( )
{
   int ∗a, b;
   b = 10;
   a = &b;
   printf(“%d%d”, b, *a);
   b = ∗/∗pointer∗/b;
}
```

(a) 35 (b) 45
(c) 55 (d) 65

Q.14 Count Number of tokens in the following given code:

```
main ()
{
    int a;
    int* a;
    For (i = 0; i < n; i++)
    {
      Printf (“True”);
      ++* a;
      a = a + a;
    }
}
```

Q.15 Find the regular expression that correctly identifies the variable name in C program.
(a) [a – z][a – zA – Z_O – 9]*
(b) [a – zA – Z_][a – zA – Z_O – 9]*
(c) [a – zA – Z_][a – zA – ZO – 9]*
(d) [a – zA – Z][a – zA – Z_O – 9]*

Q.16 Consider the following code

```
int a = 10;
Float b = 20.5;
Printf(“c = %d, d = %”, ++a, b++);
```

The number of tokens in the given code fragment is ________.

Q.17 Consider the following C Program

```
Void main ()
{
    int i = 20;
    while(i – –)
    Printf(“GATE 2020\n”);
}
```

The output of the given program is
(a) Prints Gate 2020 20 Times
(b) Prints Gate 2020 19 times
(c) Lexical error
(d) Syntax error

Q.18 Which of the following errors that are detected in the Lexical Analysis Phase.
1. Numeric literals that are too long.
2. Identifies that are too long
3. Ill-formed numeric literals
4. Input characters that are not in language.

(a) 1 and 2 only (b) 3 and 4 only
(c) 1, 3 and 4 only (d) 1, 2, 3 and 4

Answers Lexical Analysis

1.	(a)	2.	(c)	3.	(a)	4.	(c)	5.	(b)	6.	(b)	7.	(c)	8.	(d)	9.	(33)
10.	(a)	11.	(b)	12.	(b)	13.	(a)	14.	(42)	15.	(b)	16.	(21)	17.	(d)	18.	(d)
19.	(b)	20.	(18)	21.	(46)	22.	(c)	23.	(41)	24.	(24)						

Explanations Lexical Analysis

3. (a)

The given statement can be broken into

$\underline{\text{IF}}\ \underline{(}\ \underline{\text{NUMB}}\,\underline{\text{EQ}}\ \underline{\text{MAX}}\ \ \underline{)}\ \ \underline{\text{GOTO}}\ \underline{\text{50C}}$

∴ Number of tokens = 8

4. (c)

Number of tokens in C statement is 10

$\underline{\text{Print } f}\ \underline{(}\ \underline{\text{“}I = \%d, \& I=\%x\text{”}}\ \underline{,}\ \underline{i}\ \underline{,}\ \underline{\&}\ \underline{I}\ \underline{)}\ \underline{;}$

5. (b)

Lexical analysis removes comments as well. ($i = 0$) will not generate lexical error. It will tokenise as for, (, i, =, 0,). No error is generated.

6. (b)

$\overset{1}{\text{int}}\ \overset{2}{\text{strange}}\ \overset{3}{(}\ \overset{4}{\text{int}}\ \overset{5}{x}\ \overset{6}{)}$

$\overset{7}{\{}$

$\overset{8}{\text{if}}\ \overset{9}{(}\ \overset{10}{x}\ \overset{11}{<=}\ \overset{12}{0}\ \overset{13}{)}\ \overset{14}{\text{return}}\ \overset{15}{0}\ \overset{16}{;}$

$\overset{17}{\text{if}}\ \overset{18}{(}\ \overset{19}{(}\ \overset{20}{x}\ \overset{21}{\%}\ \overset{22}{2}\ \overset{23}{)}\ \overset{24}{!=}\ \overset{25}{0}\ \overset{26}{)}\ \overset{27}{\text{return}}\ \overset{28}{x}\ \overset{29}{-}\ \overset{30}{1}\ \overset{31}{;}$

$\overset{32}{\text{return}}\ \overset{33}{1}\ \overset{34}{+}\ \overset{35}{\text{strange}}\ \overset{36}{(}\ \overset{37}{x}\ \overset{38}{-}\ \overset{39}{1}\ \overset{40}{)}\ \overset{41}{;}$

$\overset{42}{\}}$

7. (c)

The C-code contains syntax error, where ‘fro’ is written in place of ‘for’. The lexical analyzer will not declare it as an error, whether ‘fro’ is a misspelling of the keyword ‘for’ or an undeclared function identifier. Since ‘fro’ is a valid lexeme for the token id, the lexeme analyzer must return the token id to the parser and let some other phase of the compiler (parser in this case) handle the error due to transposition of the letters.

8. (d)

The given program contain no lexical error even though it contains syntax errors. In line number “5”, comment started and searches for the first close comment pattern when it finds, it consider a comment. There is no start comment pattern (/*) but there is end comment at last in line 5, hence it is not lexical error but it is syntax error.

$\underbrace{\text{/ * com / * ment2 * /}}_{\text{Comment}}\ \underbrace{\text{end}}_{\text{Identifier}}\ \underbrace{*\ /}_{\text{Operator}}\ \Rightarrow \text{No lexical error}$

9. (33)

$\underset{1}{\text{main}}\ \underset{2}{(}\ \underset{3}{)}$

$\underset{4}{\{}$

$\underset{5}{\text{char}}\ \underset{6}{\text{ch}}\ \underset{7}{=}\ \underset{8}{\text{‘A’}}\ \underset{9}{;}$

$\underset{10}{\text{int}}\ \underset{11}{x}\ \underset{12}{,}\ \underset{13}{y}\ \underset{14}{;}$

$\underset{15}{x}\ \underset{16}{=}\ \underset{17}{y}\ \underset{18}{=}\ \underset{19}{20}\ \underset{20}{;}$

$\underset{21}{x}\ \underset{22}{++}\ \underset{23}{;}$

$\underset{24}{\text{printf}}\ \underset{25}{(}\ \underset{26}{\text{“\%d\%d”}}\ \underset{27}{,}\ \underset{28}{x}\ \underset{29}{,}\ \underset{30}{y}\ \underset{31}{)}\ \underset{32}{;}$

$\underset{33}{\}}$

10. (a)

Regular expression is used in lexical analysis to identify the tokens.

12. (b)

We need not to find out the actual output of the given C-program. Just by seeing the C-code we can understand that the output is a string and that is passed to lexical analyzer and that would be treated as a single valid lexeme for lexical analyzer. For instance we can see the output of the program will be 'water'.

|Water| is considered as a single token.
(1)

13. (a)

```
main ( )
{
    int * a , b ;
    b = 10 ;
    a = & b ;
    printf ( "%d%d" , b , * a ) ;
    b = * /*pointer*/ b ;
}
```

14. (42)

Code	Number of Tokens
Main ()	3
{	1
int a;	3
int*a;	4
For (i = 0; i < n; i ++)	13
{	1
Printf ("True")	5
++* a;	4
a = a + a;	6
}	1
}	1
	Total = 42

S2: Target code is backend of compiler. Hence true.

S3 : It is also true.

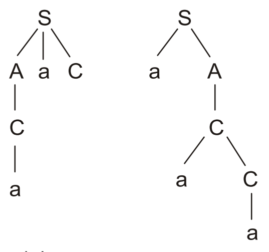
15. (b)
In C program, Variable Name consists of letters, digits and underscore symbol. Variable name must begin with a letter or underscore.
[a – zA – Z_][a – zA – Z_O – 9]*

16. (21)
int a = 10; Number of tokens are 5.
Float b = 20.5; Number of tokens are 5.
Printf("c = %d, d = %F", ++a, b++);
Number of tokens are 11
Total number of tokens are = 21

17. (d)
Compiler produces an error as the spelling of the keyword while is written as while. Lexical analyser treats it is a valid identifier. Syntax analyser will found and display an error message.

18. (d)
Lexical Analysis will detect all the mentioned errors. Numeric literals that are too long.
Identifiers that are too long.
ICC-formed numeric literals.
Input characters that are not in the language.

19. (b)
$S \to AaC / aA$
$A \to bC / C$
$C \to aC / a$
Grammer is non left recursive
$A \to bC / C$
$C \to aCa$
aaa can be generated by two different Parse tree.



So, it is ambiguous.

20. (18)
$S \to aA\ BbCD$
$A \to A\ Sd / \varepsilon$
$B \to SAc / hC / \varepsilon$
$C \to Sf / Cg$
$D \to BD / \varepsilon$
Delete $D \to \varepsilon$
$S \to aA\ BbCD\ /\ aA\ Bb\ C$
$A \to A\ Sd / \varepsilon$
$B \to SAc / hC / \varepsilon$
$C \to Sf / Cg$
$D \to BD / B$
Delete $B \to \varepsilon$
$S \to aA\ BbCD / aABbC/aAbCD/aAbC$
$A \to A\ Sd / \varepsilon$
$B \to SAc / hC$
$C \to Sf / Cg$