Internal Assessment Question Paper – 1

Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)
Department of Computer Science & Engineering

Programme: B.E **Term: March to July 2022**
Course: Compiler Design **Course Code: CS61** **Date: 13-05-2022**
CIE: I **Sem: VI** **Section: A(APK), B(SA),C(HS)**
Max Marks: 30 **Time: 1 Hr** **Portions for Test: L1-L16**

Instructions to Candidates:

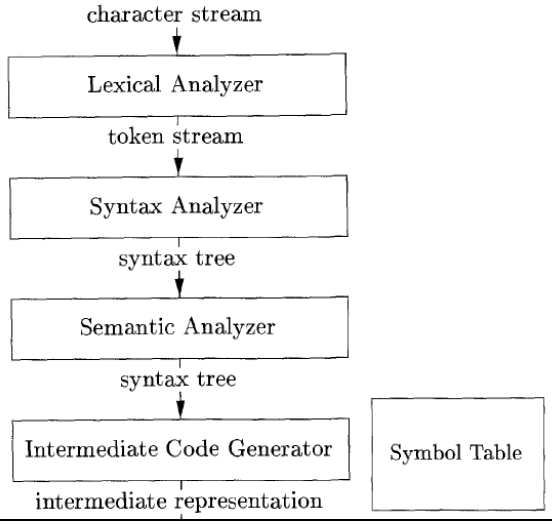
1. Question 1 is **compulsory.** Answer two full questions.
2. Each Question carries 15M.
3. Mobiles, smart watches or any electronic gadgets are strictly banned.

Sl #	Question	Marks	Bloom's Level	CO Mapping
1	a. Illustrate the analysis phase of the compiler by translating the assignment statement. Specify the values there in symbol table. **Sum=Sum*B+C*Sum;**	4	Understand	CO1
	b. Construct SLR parse table for the Grammar Given G: **S→ L** **L→ (L)L | a | ε** Do parsing for the input: **(a)$**	6	Apply	CO2
	c. Compute FIRST and FOLLOW of all Non-terminals for the grammar G i. **S→AB | Bb |c** **A→Bd | ε** **B→d | e**	2	Apply	CO2
	d. Explain the input buffering technique "sentinels" used for tokenization in lexical analysis phase with example.	3	Understand	CO1
2	a. Explain how the handles are identified for the given grammar by constructing a shift reduce parser. G: **S→AB** **A→Aa | a** **B→Bb | b**	2	Understand	CO2
	b. Define token, pattern and lexeme. "Assume that all our Transition diagrams are deterministic". Justify your answer for this statement. Construct the transition diagram for the given tokens **i. logical operators** **ii. while, which, what, wait**	3+1+3=7	Understand, Analyze	CO1
	c. Write an algorithm to construct LR (1) set of items. Applying this algorithm construct LR (1) set of items for the grammar. Construct the CLR (1) parse table for the grammar. G: **S→CC** **C→aC | b**	6	Apply	CO2
	(OR)			
3	a. Answer the following i. Write regular expression for identifying USN number of CS students. ii. Describe the languages denoted by the following regular expressions A. (a|b)*a?(a|b) B. a*b+a*b+a+	2+2=4	Understand, Apply	CO1

	b. Consider the grammar given below. If not suitable for Predictive Parsing make necessary changes and construct predictive parsing table for the grammar given below. Check whether the grammar is LL(1). G: A→ Bd | a B→ Ae | b | ε C→ B | c	7	Apply	CO2
	c. List and Explain the tasks of a lexical analyzer. Show how the lexical analyzer interacts with parser.	4	Remember, Understand	CO1

Course Outcomes meant to be assessed by the IA Test 1:
CO1: Construct lexical analyzer to recognize inputs using patterns.
CO2: Devise different types of syntax analyzers using grammars.

SCHEME & SOLUTIONS

<table>
<tr><th>Sl.No</th><th>Scheme</th><th>Marks</th></tr>
<tr><td>1 a</td><td> 
</td><td>Each phase 1M
1*4=4M</td></tr>
<tr><td>b</td><td>LR(0) set of items
S'→S
S→ L
L→ (L)L | a | ε
Items ----------2M
<table>
<tr><td>I0:
S'→.S
S→.L
L→.(L)L
L→.a
L→.</td><td>I0,S=I1
S'→S.</td><td></td><td></td><td></td></tr>
<tr><td></td><td>I0,L=I2
S→L.</td><td></td><td></td><td></td></tr>
<tr><td></td><td>I0,(=I3
L→(.L)L
L→.(L)L
L→.a
L→.</td><td>I3.L=I5
L→(L.)L

I3,(=I3
I3,a=I4</td><td>I5,)=I6
L→(L).L
L→.(L)L
L→.a
L→.</td><td>I6,L=I7
L→(L)L.
I6,(=I3
I6,a=I4</td></tr>
</table></td><td>2+2+2=6M</td></tr>
</table>

	I0,a=I4 L→a.			

Table ----------**2M**

	Action				GOTO	
States	(	)	a	$	S	L
0	S3	R4	S4	R4	1	2
1				acc		
2				R1		
3	S3	R4	S4	R4		5
4		R3		R3		
5		S6				
6	S3	R4	S4	R4		7
7		R2		R2		

Follow(S)={$}
Follow(L)={$,)}

Parsing on input: **(a)$ ----------2M**

Stack	Symbols	Input	Action
$0		(a)$	Shift
$03	(	a)$	Shift
$034	(a	)$	Reduce L→a
$035	(L	)$	Shift
$0356	(L)	$	Reduce L→ ε
$03567	(L)L	$	Reduce L→(L)L
$02	L	$	Reduce S→L
$01	S	$	accept

c

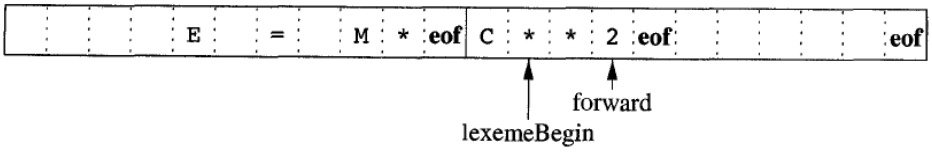
S→AB | Bb |c
A→Bd | ε
B→d | e

Non Terminal	FIRST	FOLLOW
S	{d,e,c}	{$}
A	{d,e, ε}	{d,e}
B	{d,e}	{b,d,$}

First- 1M
Follow-1M

d

Input Buffering Technique: sentinels



Sentinel is a special character eof.

Two pointers to the input are maintained:

1. Pointer `lexemeBegin`, marks the beginning of the current lexeme, whose extent we are attempting to determine.
2. Pointer `forward` scans ahead until a pattern match is found; the exact strategy whereby this determination is made will be covered in the balance of this chapter.

Lexeme Begin and Forward pointer-1M
Diagram -1M
Explanation-1M

```
switch ( *forward++ ) {
        case eof:
                if (forward is at end of first buffer ) {
                        reload second buffer;
                        forward = beginning of second buffer;
                }
                else if (forward is at end of second buffer ) {
                        reload first buffer;
                        forward = beginning of first buffer;
                }
                else /* eof within a buffer marks the end of input */
                        terminate lexical analysis;
                break;
        Cases for the other characters
}
```

2a

S→AB
A→Aa | a
B→Bb | b
Handle are the substrings that matches the body of the production, whose reduction represents one step along the reverse of the rightmost derivation.
Any valid input: ab$
Shift reduce parser: ---------- 2M

Stack	Handle	input	Action
$		ab$	shift
$a	a	b$	Reduce
$A		b$	
$Ab	b	$	Shift
$AB	AB	$	Reduce
$S			Accept

b

Token: pair consisting of token name and optional attribute value
------- 1M
Pattern : description of the form that the lexeme of a token may take
-----1M
Lexeme: sequence of characters there in the source program that matches the pattern for a token. **---------- 1M**
All transition diagrams are deterministic, there is never more than one edge out of a given state with a given symbol among its labels.
---------- 1M

i. **logical operators ---------- 1.5M**

ii. **while, which, what, wait ----------1.5M**

c

```
SetOfItems CLOSURE(I) {
        repeat
                for ( each item [A → α·Bβ, a] in I )
                        for ( each production B → γ in G' )
                                for ( each terminal b in FIRST(βa) )
                                        add [B → ·γ, b] to set I;
        until no more items are added to I;
        return I;
}
```

------2M

S→CC
C→Ac | b
ITEMS---------- 2M

I0: S'→.S, $ S→CC,$ C→.Ac, a/b C→.b ,a/b	I0,S=I1 S'→S., $			
	I0,C=I2 S→C.C,$ C→.Ac, $ C→.b , $	I2, C=I5 S→CC.,$	I2,a=I6 C→a.C,$ C→.Ac, $ C→.b , $	I2,b=I7 C→b., $
	I0,a=I3 C→a.C, a/b C→.aC, a/b C→.b ,a/b	I3,C=I8 C→aC. , a/b I3,a=I3 I3,b=I4	I6,C=I9 C→aC. ,$ I6,a=I6 I6,b=I7	
	I0,b=I4 C→b., a/b			

PT---------- **2M**

	Action			GOTO	
States	a	b	$	S	C
0	S3	S4		1	2
1			acc		
2	S6	S7			5
3	S3	S4			8
4	R3	R3			
5			R1		
6	S6	S7			9
7			R3		
8	R2	R2			
9			R2		

3a

i. Write regular expression for identifying USN number of CS students.
["1MS"|"1ms"][12][0-9]["CS"|"cs"][0-9][0-9][0-9] ---------**2M**

ii. Describe the languages denoted by the following regular expressions
(a|b)*a?(a|b) – All strings containing a's and b's with alteast one a or b ----**1M**
a*b+a*b+a+ - All strings containing a's and b's atleast length of the string is 3. ----**1M**

b

A→ Bd | a
B→ Ae | b | ε
C→ B | c

Indirect Left recursion:
A→Bd|a
B→Bde | ae |b | ε
C→ B | c
After eliminating left recursion ---------- **2M**
A→Bd|a
B→aeB'|bB'|B'
B'→deB'| ε
C→ aeB'|bB'|B'|c OR **C→B |c //please consider**

FIRST AND FOLLOW ----------**2M**

First (B)={a,b,d, ε}
First(B')={d, ε}
Follow(B)={d}
PT----------**2M**

Input Non Terminals	a	b	c	d	e	$
A	A→a A→Bd	A→Bd		A→Bd		
B	B→aeB'	B→bB'		B→B'		
B'				B'→deB'		
C	C→ aeB'	C→ bB'	C→c	C→B'		

Not LL(1) ---------- **1M**

c

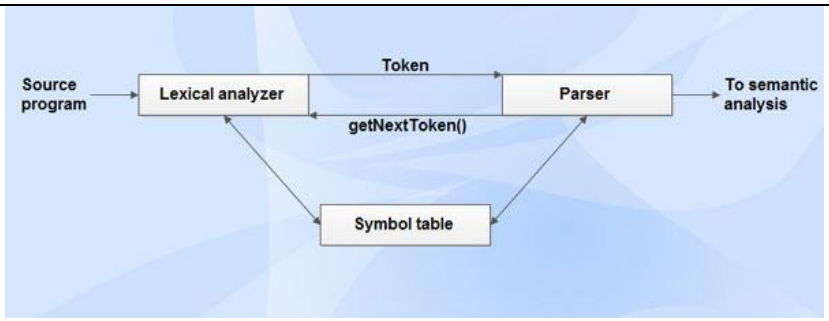


Diagram-2M

Lexical analyzer performs the following tasks: --------- 2M

• Reads the source program, scans the input characters, group them into lexemes and produce the token as output.

• Enters the identified token into the symbol table.

• Strips out white spaces and comments from source program.

• Correlates error messages with the source program i.e., displays error message with its occurrence by specifying the line number.

• Expands the macros if it is found in the source program.

The sequence of tokens produced by lexical analyzer helps the parser in analyzing the syntax of programming languages.