

**Test: CLAT-1**

**Course Code & Title: 18CSC304J -COMPILER DESIGN**

**Year & Sem: III & VI**

**Date: 17.2.2023**

**Duration: 1 HOUR**

**Max. Marks: 25**

**Course Articulation Matrix:**

S.No.	Course Outcome	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
1	CO1	3	3	3									

**Part – A ( 5 x 1 = 5 Marks)**

**Instructions: Answer ALL**

Q. No	Question	Marks	BL	CO	PO	PI Code
1	The regular expression $(0 1)^*(0 1)$ represents a language with a) <b>Nonempty binary strings</b> b) Empty and nonempty binary strings c) Odd nonempty strings d) Even nonempty strings  Answer: a	1	2	1	1	1.4.1
2	The total number of states to build the given language using DFA: $L = \{w   w \text{ has exactly 2 a's and at least 2 b's}\}$ a) <b>10</b> b) 11    c) 12    d) 13 Answer: a	1	3	1	2	2.1.3
3	Which of the following is not a regular expression? a) $[(a+b)^*(aa+bb)]^*$ b) $[(0+1)-(0b+a1)^*(a+b)]^*$ c) $(01+11+10)^*$ d) $(1+2+0)^*(1+2)^*$  Answer: b	1	2	1	2	2.1.2
4	Regular expression $\Phi^*$ is equivalent to a) $\epsilon$ b) $\Phi$ c) 0    d) 1 Answer :a	1	1	1	1	1.2.1
5	_____ takes collection of rules that define the translation of each operation of the intermediate language into the machine language for the target machine. a. Parser generators b. Scanner generators c. Syntax-directed translation engines d. <b>Automatic code generators</b> Answer : D	1	1	1	1	1.3.1

**Part – B ( 2 x 4 = 8 Marks)**

**Instructions: Answer any TWO**

6	The two tests schemes can be reduced to one in input buffering technique? justify your answer with an algorithm.	4	1	1	1	1.3.1
---	--	---	---	---	---	-------



Answer

First Construct Transition table for the given diagram

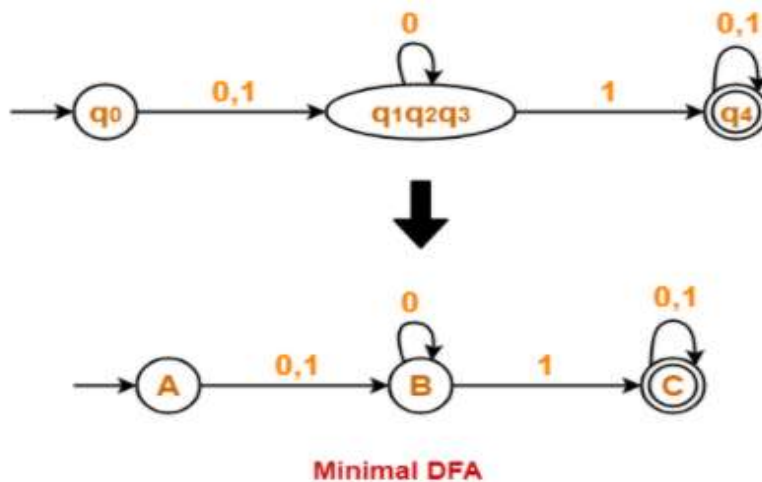
	0	1
→q0	q1	q3
q1	q2	*q4
q2	q1	*q4
q3	q2	*q4
*q4	*q4	*q4

Now using Equivalence Theorem, we have-

$$P_0 = \{ q_0, q_1, q_2, q_3 \} \{ q_4 \}$$

$$P_1 = \{ q_0 \} \{ q_1, q_2, q_3 \} \{ q_4 \}$$

$$P_2 = \{ q_0 \} \{ q_1, q_2, q_3 \} \{ q_4 \}$$

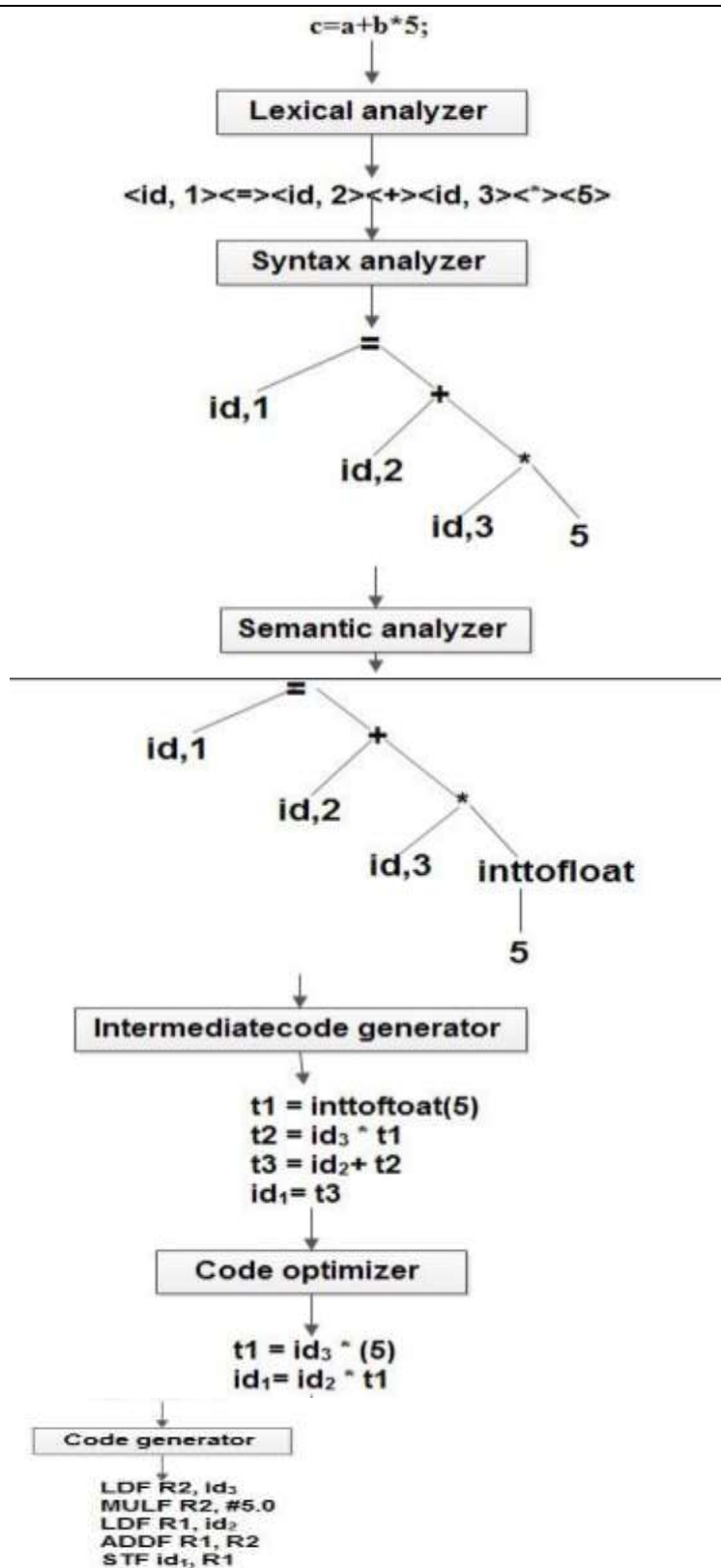


**Part – C ( 1 x 12 = 12 Marks)**

**Instructions: Answer any ONE**

9	(i). Consider the input $c = a + b * 5$ . With a neat sketch, illustrate how the input is transformed into assembly code, using all the phases of compiler. (Problem solving-4 marks, explanation-4)	8	2	1	2	2.2.1
---	---	---	---	---	---	-------

4



(ii). Illustrate LEX code with an example.

Answer:

```

%{

#include <stdio.h>
#include "y.tab.h"
int c;
extern int yylval;
%}
%%
" "      ;
[a-z]    {
    c = yytext[0];
    yylval = c - 'a';
}
  
```

	<pre> return(LETTER); } [0-9] { c = yytext[0]; yyval = c - '0'; return(DIGIT); } [^a-z0-9\b] { c = yytext[0]; return(c); } </pre>					
--	---	--	--	--	--	--

OR																		
10	(i). Convert the following Non-Deterministic Finite Automata (NFA) to Deterministic Finite Automata (DFA) using subset construction method.	8	3	1	3	3.3.2												
<pre>graph LR     start(( )) --&gt; q0((q0))     q0 -- 0 --&gt; q0     q0 -- 1 --&gt; q1((q1))     q1 -- 0 --&gt; q1     q1 -- "0,1" --&gt; q2(((q2)))     q2 -- 0 --&gt; q0     q2 -- "0,1" --&gt; q1     style start fill:none,stroke:none</pre>		4																
Answer : Accept any method of conversion for this question																		
<p><u>Transition Table</u></p> <table><tr><th>Input state</th><th>0</th><th>1</th></tr><tr><td>→ q<sub>0</sub></td><td>q<sub>0</sub></td><td>q<sub>1</sub>, q<sub>2</sub></td></tr><tr><td>q<sub>1</sub></td><td>q<sub>1</sub>, q<sub>2</sub></td><td>q<sub>2</sub></td></tr><tr><td>Ⓠ<sub>2</sub></td><td>q<sub>0</sub>, q<sub>1</sub></td><td>q<sub>1</sub></td></tr></table>							Input state	0	1	→ q <sub>0</sub>	q <sub>0</sub>	q <sub>1</sub> , q <sub>2</sub>	q <sub>1</sub>	q <sub>1</sub> , q <sub>2</sub>	q <sub>2</sub>	Ⓠ <sub>2</sub>	q <sub>0</sub> , q <sub>1</sub>	q <sub>1</sub>
Input state	0	1																
→ q <sub>0</sub>	q <sub>0</sub>	q <sub>1</sub> , q <sub>2</sub>																
q <sub>1</sub>	q <sub>1</sub> , q <sub>2</sub>	q <sub>2</sub>																
Ⓠ <sub>2</sub>	q <sub>0</sub> , q <sub>1</sub>	q <sub>1</sub>																

Let  $\delta'$  be the transition function of the DFA.

Let  $[q_0]$  be the initial state of the DFA.

$$\delta'([q_0], 0) = \delta([q_0], 0) = [q_0]$$

$$\delta'([q_0], 1) = \delta([q_0], 1) = [q_1, q_2]$$

$$\delta'([q_1, q_2], 1) = \delta([q_1], 1) \cup \delta([q_2], 1)$$

$$\delta'([q_1, q_2], 0) = \delta([q_1], 0) \cup \delta([q_2], 0)$$

$$= [q_0, q_1, q_2] \cup \delta([q_1], 0) \cup \delta([q_2], 0)$$

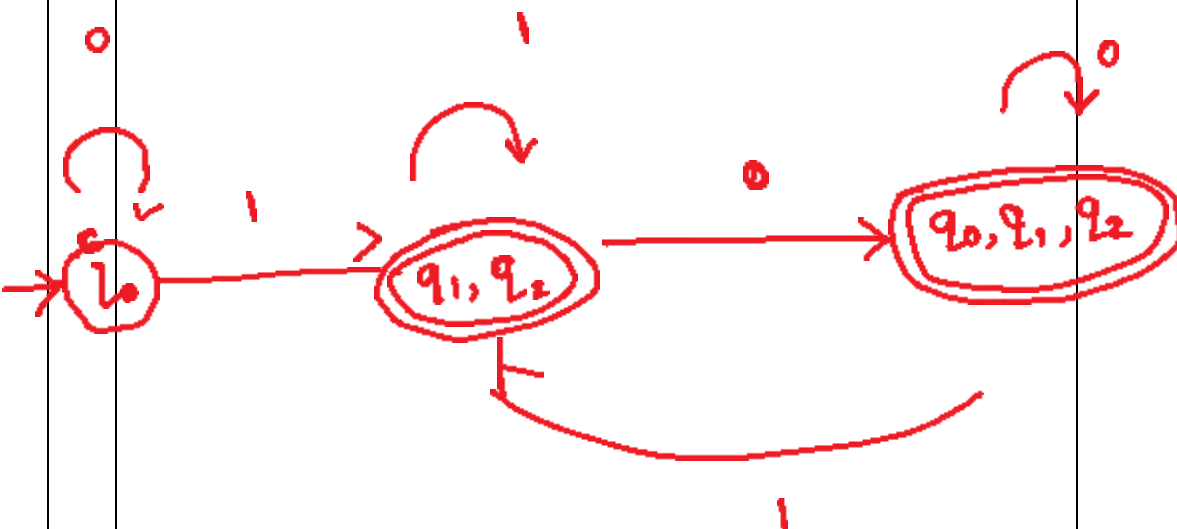
$$\delta'([q_0, q_1, q_2], 0) = \delta([q_0], 0)$$

$$= [q_0, q_1, q_2]$$

$$\delta'([q_0, q_1, q_2], 1) = \delta([q_0], 1) \cup \delta([q_1], 1) \cup \delta([q_2], 1)$$

$$= [q_1, q_2]$$

0		1
$\rightarrow [q_0]$	$[q_0]$	$[q_1, q_2]$
$[q_1, q_2]$	$[q_0, q_1, q_2]$	$[q_1, q_2]$
$[q_0, q_1, q_2]$	$[q_0, q_1, q_2]$	$[q_1, q_2]$



(ii). Inference the importance of the compiler construction tools

Answer :

Some commonly used compiler-construction tools. include

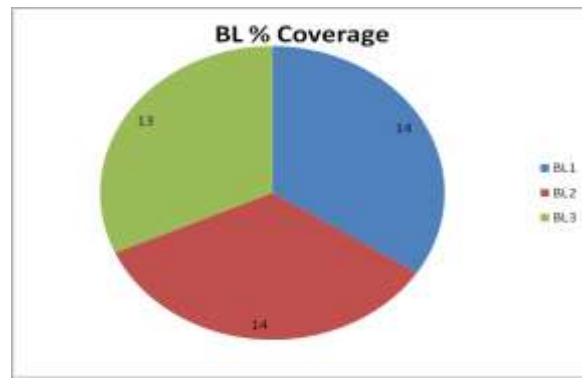
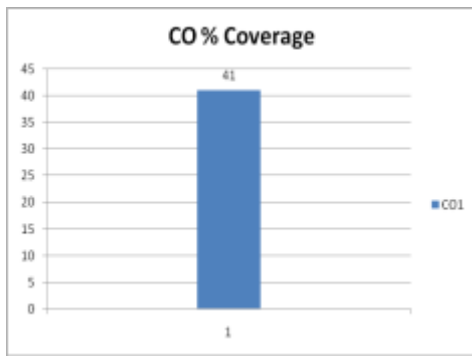
1. Parser generators.
2. Scanner generators.
3. Syntax-directed translation engines.
4. Automatic code generators.
5. Data-flow analysis engines.
6. Compiler-construction toolkits.

	<p><b>Parser Generators</b></p> <p><b>Input:</b> Grammatical description of a programming language  <b>Output:</b> Syntax analyzers.</p> <p>Parser generator takes the grammatical description of a programming language and produces a syntax analyzer.</p> <p><b>Scanner Generators</b></p> <p><b>Input:</b> Regular expression description of the tokens of a language  <b>Output:</b> Lexical analyzers.</p> <p>Scanner generator generates lexical analyzers from a regular expression description of the tokens of a language.</p> <p><b>Syntax-directed Translation Engines</b></p> <p><b>Input:</b> Parse tree.  <b>Output:</b> Intermediate code.</p> <p>Syntax-directed translation engines produce collections of routines that walk a parse tree and generates intermediate code.</p> <p><b>Automatic Code Generators</b></p> <p><b>Input:</b> Intermediate language.  <b>Output:</b> Machine language.</p> <p>Code-generator takes a collection of rules that define the translation of each operation of the intermediate language into the machine language for a target machine.</p> <p><b>Data-flow Analysis Engines</b></p> <p>Data-flow analysis engine gathers the <a href="#">information</a>, that is, the values transmitted from one part of a program to each of the other parts. Data-flow analysis is a key part of code optimization.</p> <p><b>Compiler Construction Toolkits</b></p> <p>The toolkits provide integrated set of routines for various phases of compiler. Compiler construction toolkits provide an integrated set of routines for construction of phases of compiler.</p>					
--	--	--	--	--	--	--

\*Performance Indicators are available separately for Computer Science and Engineering in AICTE examination reforms policy.

Course Outcome (CO) and Bloom's level (BL) Coverage in Questions





**Approved by the Audit Professor/Course Coordinator**