SRM Institute of Science and Technology
College of Engineering and Technology
SCHOOL OF COMPUTING
SRM Nagar, Kattankulathur – 603203, Chengalpattu District, Tamilnadu

Mode of Exam
OFFLINE

Academic Year: 2022-23 (EVEN) **SET-A**

Test: CLAT-1 **Date: 17.2.2023**
Course Code & Title: 18CSC304J -COMPILER DESIGN **Duration: 1 HOUR**
Year & Sem: III & VI **Max. Marks: 25**

Course Articulation Matrix:

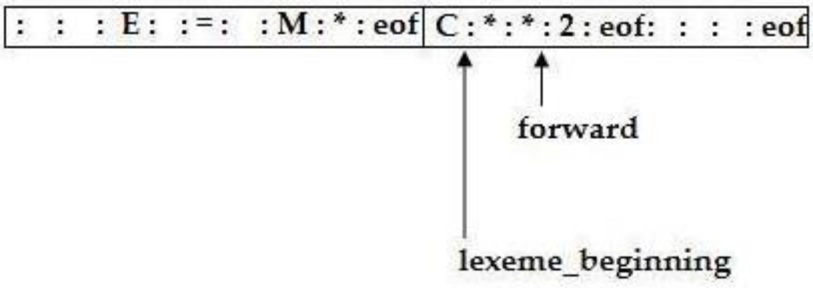
S.No.	Course Outcome	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
1	CO1	3	3	3									

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) Nonempty binary strings** 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 has exactly 2 a's and at least 2 b's} a) **10** b) 11 c)12 d)13 Answr\]er: 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 Φ* is equivalent to **a) ϵ** b) Φ 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. Automatic code generators** Answer : D	1	1	1	1	1.3.1

Part – B (2 x 4 = 8 Marks)
Instructions: Answer any TWO

Q. No	Question	Marks	BL	CO	PO	PI Code
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

<table>
<tr>
<td></td>
<td>
The two tests can be reduced to one, if each buffer half holds a sentinel character at the end.
The sentinel is a special character of eof.


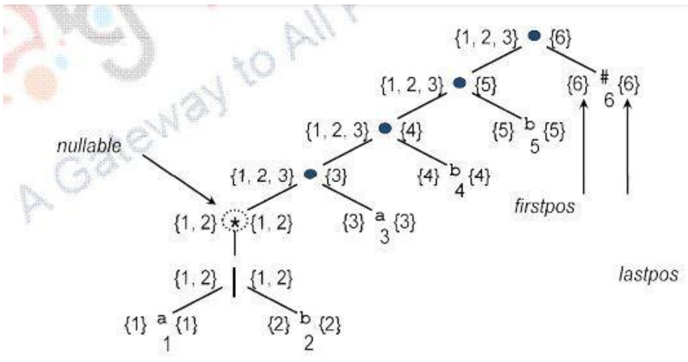
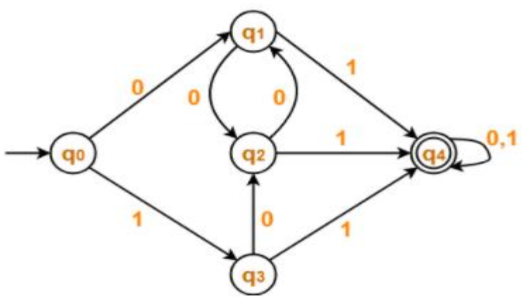
<pre>
forward := forward + 1’
if forward = eof then begin
        if forward at end of first half then begin
                reload second half;
                forward := forward + 1
        end
        else if forward at end of second half then begin
                reload first half;
                move forward to beginning of first half
        end
        else
                terminate lexical analysis
end
</pre>
</td>
<td></td><td></td><td></td><td></td><td></td>
</tr>
<tr>
<td>7</td>
<td>
Construct a syntax tree with firstpos and lastpos for all nodes of (a|b)*abb.


Figure 6.3 Syntax tree with firstpos() and lastpos() marked.
</td>
<td>4</td><td>2</td><td>1</td><td>2</td><td>2.3.1</td>
</tr>
<tr>
<td>8</td>
<td>
Construct the minimal DFA for the below diagram.


</td>
<td>4</td><td>3</td><td>1</td><td>2</td><td>2.3.1</td>
</tr>
</table>

Answer

First Construct Transition table for the given diagram

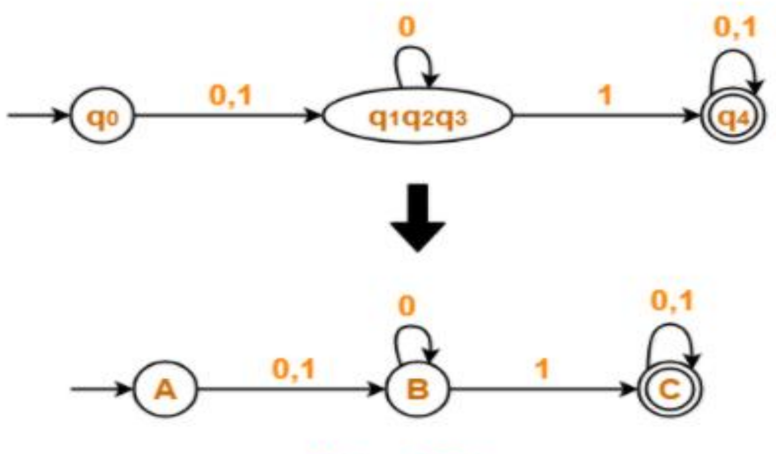
	0	1
→q0	q1	q3
q1	q2	*q4
q2	q1	*q4
q3	q2	*q4
***q4**	*q4	*q4

Now using Equivalence Theorem, we have-

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

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

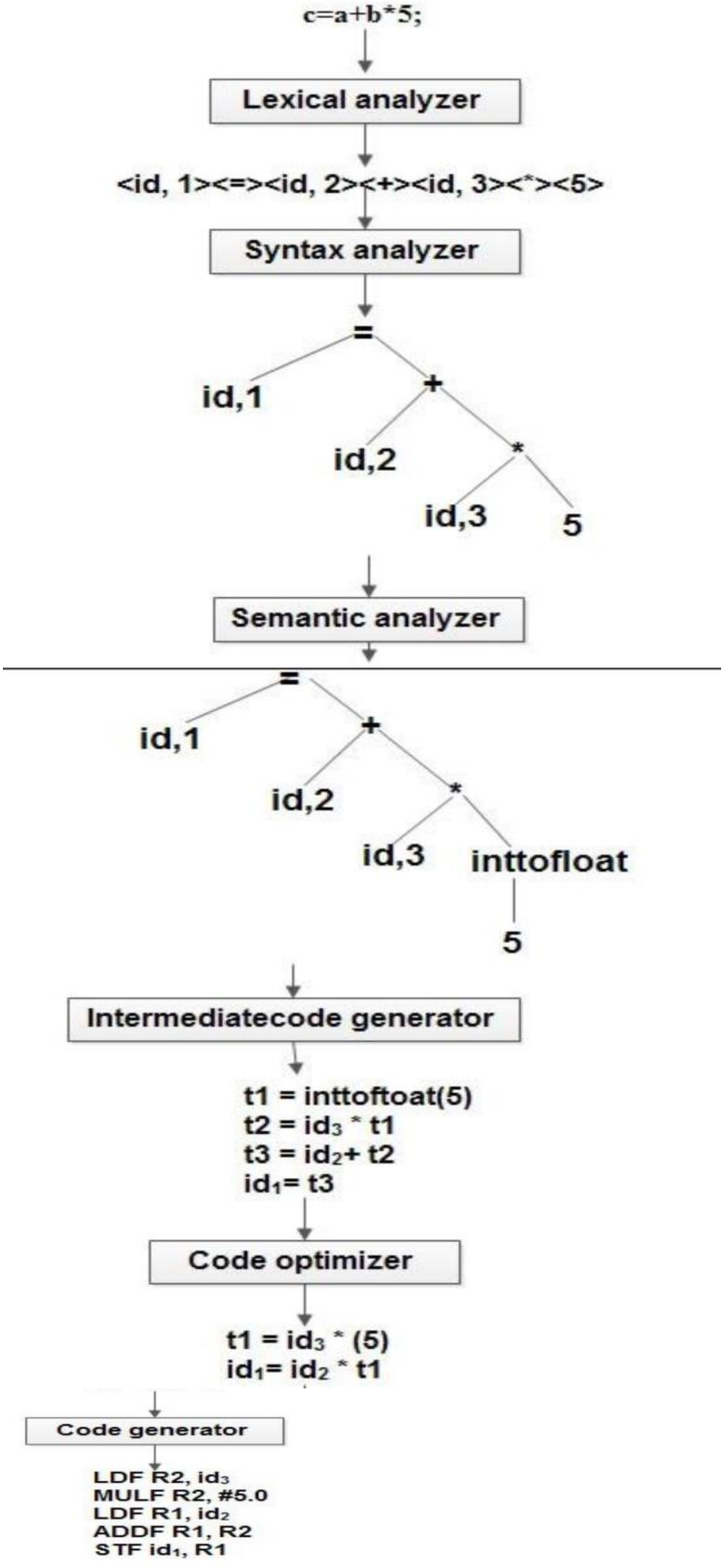
$P_2 = \{ q_0 \} \{ q_1 , q_2 , q_3 \} \{ q_4 \}$



Minimal DFA

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 4	2	1	2	2.2.1



(ii). Illustrate LEX code with an example.

Answer:

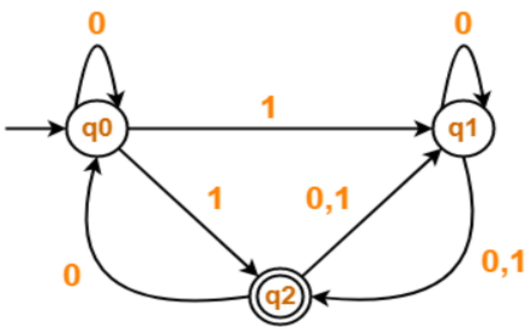
```
%{

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

```
            return(LETTER);
          }
[0-9]     {
            c = yytext[0];
            yylval = c - '0';
            return(DIGIT);
          }
[^a-z0-9\b]    {
                 c = yytext[0];
                 return(c);
               }
```

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



4

Answer : Accept any method of conversion for this question

Transition Table

State \ Input	0	1
→q_0	q_0	q_1, q_2
q_1	q_1, q_2	q_2
q_2	q_0, q_1	q_1

Let δ' be the transition function of the DFA.

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

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

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

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

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

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

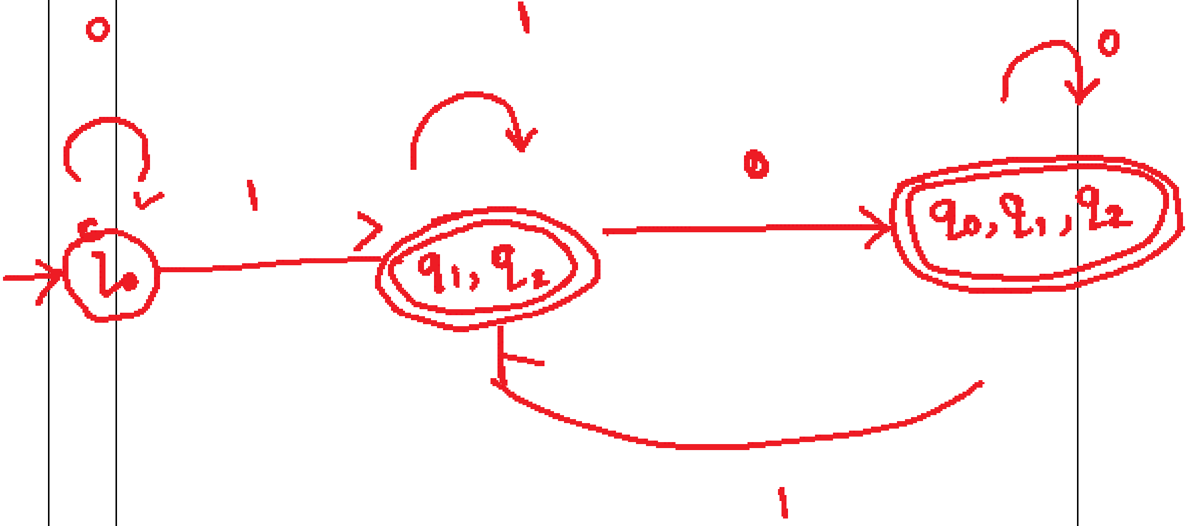
$$\delta'([q_0, q_1, q_2], 0) = \delta([q_0], 0) \cup \delta([q_1], 0) \cup \delta([q_2], 0)$$

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

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

$$= [q_1, q_2]$$

	0	1
→[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 :

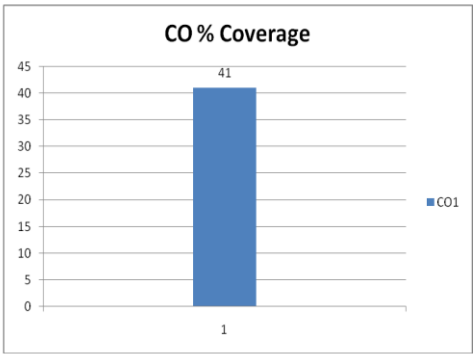
Some commonly used compiler-construction tools. include

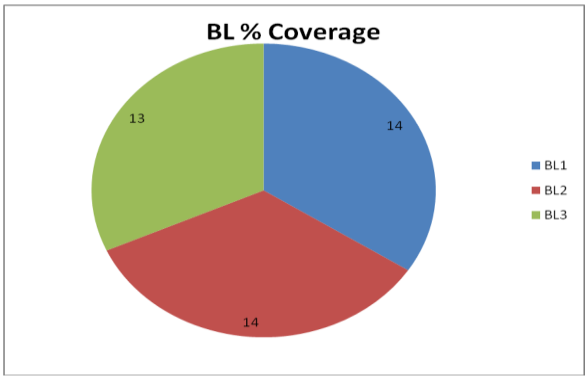
1. Parser generators.
2. Scanner generators.
3. Syntax-directed translation engines.
4. Automatic code generators.
5. Data-flow analysis engines.
6. Compiler-construction toolkits.

	Parser Generators **Input:** Grammatical description of a programming language **Output:** Syntax analyzers. Parser generator takes the grammatical description of a programming language and produces a syntax analyzer. ***Scanner Generators*** **Input:** Regular expression description of the tokens of a language **Output:** Lexical analyzers. Scanner generator generates lexical analyzers from a regular expression description of the tokens of a language. ***Syntax-directed Translation Engines*** **Input:** Parse tree. **Output:** Intermediate code. Syntax-directed translation engines produce collections of routines that walk a parse tree and generates intermediate code. ***Automatic Code Generators*** **Input:** Intermediate language. **Output:** Machine language. 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. ***Data-flow Analysis Engines*** Data-flow analysis engine gathers the information, 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. ***Compiler Construction Toolkits*** 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.					

***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