

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, RAMAPURAM CAMPUS
FACULTY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

ANSWER KEY SUBMISSION

Date of Exam & Session	17.02.2023 & AN	Category of Exam	CLA1
Course Name	COMPILER DESIGN	Course Code	18CSC304J
Name of the Faculty submitting	Dr. C. SIVASANKAR	Date of submission of Answer Key	20.02.2023
Department to which the Faculty belongs to	CSE	Total Marks	25

PART A (5 X 1= 5)
ANSWER ALL THE QUESTIONS

Q.No.	MCQ Questions	Marks	CO	BL	PI
1.	What is the output of lexical analyzer? a) A parse tree b) A list of tokens c) Intermediate code d) Machine code	1	1	1	1.3.1
2.	An NFA's transition function returns a) A Boolean value b) A state c) A set of states d) An edge	1	1	1	2.1.3
3.	When is the type checking usually done? a) Syntax directed translation b) lexical analysis c) code optimization d) syntax analysis	1	1	1	2.1.3
4.	Transition function maps. a) $\Sigma^* Q \rightarrow \Sigma$ b) $Q^* Q \rightarrow \Sigma$ c) $\Sigma^* \Sigma \rightarrow Q$ d) $Q^* \Sigma \rightarrow Q$	1	1	1	2.1.3
5.	Which of the following is used for grouping of characters into tokens (in a computer) a) A parser b) Code optimizer c) Code generator d) Scanner	1	1	1	1.3.1

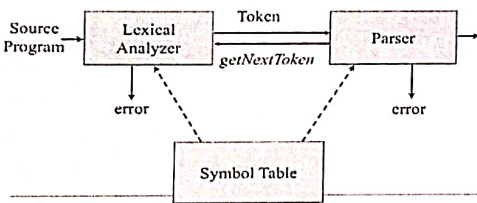
PART B (2 X 4 =8)
ANSWER ALL THE QUESTIONS

Q.No.	Questions	Marks	CO	BL	PI
6.	Discuss Cousins of Compiler in detail COUSINS OF COMPILER 1. Preprocessor 2. Assembler 3. Loader and 4. Linker Preprocessor :A preprocessor is a program that processes its input data to produce output that is used as input to another program. The output is said to be a preprocessed form of the input data, which is often used by some subsequent programs like compilers. They may perform the following functions : 1. Macro processing 3. Rational Preprocessors 2. File Inclusion 4. Language extension	4	1	1	2.2.3

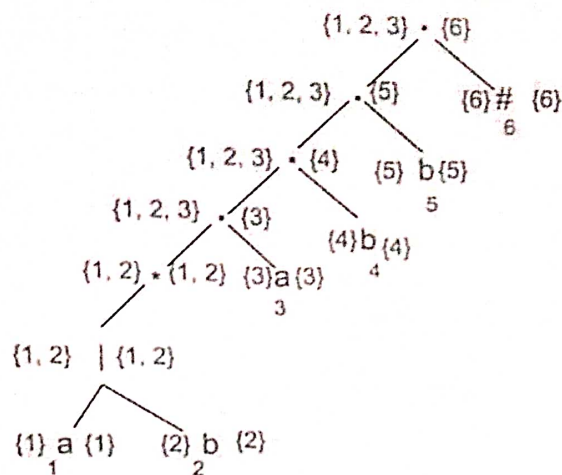
	<p>Assembler: Assembler creates object code by translating assembly instruction mnemonics into machine code.</p> <p>There are two types of assemblers: ·</p> <ul style="list-style-type: none"> • One-pass assemblers • Two-pass assemblers <p>Linker and Loader : A linker or link editor is a program that takes one or more objects generated by a compiler and combines them into a single executable program. Three tasks of the linker are 1. Searches the program to find library routines used by program, e.g. printf(), math routines.</p> <p>2. Determines the memory locations that code from each module will occupy and relocates its instructions by adjusting absolute references</p> <p>3. Resolves references among files</p> <p>A loader is the part of an operating system that is responsible for loading programs in memory, one of the essential stages in the process of starting a program</p>				
7.	<p>What data used to evaluate NFA with DFA:</p> <ul style="list-style-type: none"> • Input Symbol • Transition Function • Transition Table • Transition Diagram • No of States • Time Complexity • Space Complexity <p>Input Symbol:</p> <p>i) For each symbolic representation of the alphabet, there is only one state transition in DFA.</p> <p>Each pair of state and input symbol can have many possible next states in NFA</p> <p>ii) Epsilon Transition:</p> <p>DFA cannot use Empty String transition where as NFA can use Empty String transition</p> <p>Transition Function :</p> <p>i) δ: It is a transition function that takes two arguments, a state, and an input symbol, it returns a single state. – In DFA (corresponding to an input symbol, there is a single resultant state i.e. there is only one transition)</p> <p>ii) δ: It is a transition function that takes a state from Q and an input symbol and returns a subset of Q more than one possible transition from one state on the same input symbol. (In NFA)</p> <p>Transition Diagram: Diagram for any Regex: for example a/b (Using NFA and DFA)</p> <p>Transition Table : Table for same Regex: a/b (Transition table for NFA and DFA)</p>	4	1	3	2.2.4

PART C (1 X 12 = 12)
ANSWER THE QUESTION (A OR B)

Q. No.	Questions	Marks	CO	BL	PI
8.	<p>a) i) Describe the Role of Lexical Analyzer in the process of Compilation</p> <p>The lexical Analyzer being the first phase in compiler</p>	6	1	1	3.1.1

	<p>designing, plays an important role.</p> <p>It reads the source program, scans it, groups them into lexemes, and gives a sequence of tokens.</p> <p>It eliminates the comments and white spaces present in the source code.</p> <p>It helps in removing errors.</p> <p>It reads the source program character by character and returns tokens of the source program.</p> <p>It feeds the information about identifiers into the symbol table.</p> <p>It expands macros in the source code if any is found.</p> <p><u>The Role of the Lexical Analyzer</u></p>  <pre> graph LR SP[Source Program] --> LA[Lexical Analyzer] LA -- Token --> P[Parser] P -- getNextToken() --> LA LA -. error .-> ST[Symbol Table] P -. error .-> ST </pre>				
	<p>ii) Show the operation performed on Regular Languages with example</p> <p>Operations performed on regular expressions:</p> <p>1. Union: The union of two regular languages, L1 and L2, which are represented using $L1 \cup L2$, is also regular and which represents the set of strings that are either in L1 or L2 or both.</p> <p>Example: $L1 = (1+0).(1+0) = \{00, 10, 11, 01\}$ and $L2 = \{\epsilon, 100\}$ then $L1 \cup L2 = \{\epsilon, 00, 10, 11, 01, 100\}$.</p> <p>2. Concatenation: The concatenation of two regular languages, L1 and L2, which are represented using $L1.L2$ is also regular and which represents the set of strings that are formed by taking any string in L1 concatenating it with any string in L2.</p> <p>Example: $L1 = \{0, 1\}$ and $L2 = \{00, 11\}$ then $L1.L2 = \{000, 011, 100, 111\}$.</p> <p>3. Kleene closure If L1 is a regular language, then the Kleene closure i.e. $L1^*$ of L1 is also regular and represents the set of those strings which are formed by taking a number of strings from L1 and the same string can be repeated any number of times and concatenating those strings.</p> <p>Example: $L1 = \{0, 1\} = \{\epsilon, 0, 1, 00, 01, 10, 11, \dots\}$, then $L1^*$ is all strings possible with symbols 0 and 1 including a null string.</p> <p>Positive Closure R is a regular expression whose language is $L.R^+$ is a regular</p>	6	1	3	3.1.6

<p>expression whose language is L^+</p> <p>Example Suppose $R = (a)$ then its language will be $L = \{a\}$. Now apply positive Closure on given regular expression and language. if $R^+ = (a)^+$ then its language will be $L^+ = \{a, aa, aaa, aaaa, \dots\}$</p>				
OR				
<p>b) Solve the given Regular Expression $(a b)^*abb$ into DFA by Direct Method</p> <p>Firstly, we construct the augmented regular expression for the given expression. By concatenating a unique right-end marker '#' to a regular expression r, we give the accepting state for r a transition on '#' making it an important state of the NFA for $r\#$.</p> <p>So, $r' = (a b)^*abb\#$</p> <p>2. Then we construct the syntax tree for $r\#$.</p> <div data-bbox="399 672 845 1164"> </div> <p>Next we need to evaluate four functions nullable, firstpos, lastpos, and followpos.</p> <ol style="list-style-type: none"> 1. nullable(n) is true for a syntax tree node n if and only if the regular expression represented by n has ϵ in its language. 2. firstpos(n) gives the set of positions that can match the first symbol of a string generated by the subexpression rooted at n. 3. lastpos(n) gives the set of positions that can match the last symbol of a string generated by the subexpression rooted at n. <p>We refer to an interior node as a cat-node, or-node, or star-node if it is labeled by a concatenation, or * operator, respectively</p> <p>Compute Firstpos and last Pos:</p>	12	1	2	3.2.1



Let us now compute the followpos bottom up for each node in the syntax tree

NODE	followpos
1	{1, 2, 3}
2	{1, 2, 3}
3	{4}
4	{5}
5	{6}
6	\emptyset

4. Now we construct $Dstates$, the set of states of DFA D and $Dtran$, the transition table for D . The start state of DFA D is $firstpos(root)$ and the accepting states are all those containing the position associated with the endmarker symbol $\#$.
According to our example, the $firstpos$ of the root is $\{1, 2, 3\}$. Let this state be A and consider the input symbol a . Positions 1 and 3 are for a , so let $B = followpos(1) \cup followpos(3) = \{1, 2, 3, 4\}$. Since this set has not yet been seen, we set $Dtran[A, a] := B$.

When we consider input b , we find that out of the positions in A , only 2 is associated with b , thus we consider the set $followpos(2) = \{1, 2, 3\}$. Since this set has already been seen before, we do not add it to $Dstates$ but we add the transition $Dtran[A, b] := A$.

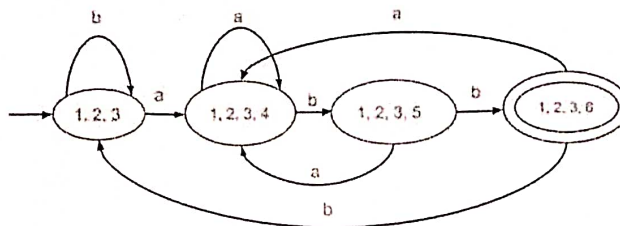
Continuing like this with the rest of the states, we arrive at the below transition table

Input

State	a	b
→ A	B	A
B	B	C
C	B	D
D	B	A

Here, A is the start state and D is the accepting state.

5. Finally we draw the DFA for the above transition table.
The final DFA will be :



HOD (CSE)

2012/25