

	DR. BABASAHEB AMBEDKAR TECHNOLOGICAL UNIVERSITY, LONERE Supplementary Semester Examination – Winter 2023 Course: B. Tech. Branch : Computer Engineering and Allied Semester : VI Subject Code & Name: BTCOC601_Y23- Compiler Design Max Marks: 60 Date: 16/01/2024 Duration: 3 Hr.		
	Instructions to the Students: 1. All the questions are compulsory. 2. The level of question/expected answer as per OBE or the Course Outcome (CO) on which the question is based is mentioned in () in front of the question. 3. Use of non-programmable scientific calculators is allowed. 4. Assume suitable data wherever necessary and mention it clearly.		
		(Level/CO)	Marks
Q. 1	Solve any Two of the following.		
A)	What is a compiler? Draw a neat diagram and explain different phases of compiler.	(Remembering)	6
B)	Define patterns, lexeme, and tokens. Explain lexical errors with examples.	(Remembering)	6
C)	i) In a string of length <i>n</i> , how many prefixes, suffixes, and proper prefixes are there? ii) Write regular definitions for the following languages: 1. All strings of lowercase letters that contain the five vowels in order. 2. All strings of lowercase letters in which the letters are in ascending lexicographic order.	(Analyzing)	6
Q.2	Solve any Two of the following.		
A)	Give the significance of input buffering and sentinel in lexical analysis.	(Remembering)	6
B)	List the tokens and their types in the following C++ code fragment. float limited_square(x) {float x; /* returns 100 or x-squared */ return(x <= -10.0 x >= 10.0)?100 : x*x; }	(Applying)	6
C)	Explain with flow diagram how lexical analyzer is created with Lex. Write a Lex program for counting words, lines, and characters in a paragraph / document file.	(Remembering) (Analyzing)	6
Q. 3	Solve any Two of the following.		
A)	With neat diagram describe the role of parser.	(Remembering)	6
B)	What is left recursion? Eliminate the left recursion from the following grammar: E → E+T T T → T*F F F → (E) id	(Remembering) (Applying)	6

C)	Compute FIRST and FOLLOW and also construct the predictive parsing table for the following grammar: $S \rightarrow +SS \mid *SS \mid a$	(Applying)	6																
Q.4	Solve any Two of the following.		12																
A)	For the SDD given below, give annotated parse tree for the expression $3*5+4$ <table border="1"> <thead> <tr> <th>PRODUCTION</th> <th>SEMANTIC RULES</th> </tr> </thead> <tbody> <tr> <td>1) $L \rightarrow E \mathbf{n}$</td> <td>$L.val = E.val$</td> </tr> <tr> <td>2) $E \rightarrow E_1 + T$</td> <td>$E.val = E_1.val + T.val$</td> </tr> <tr> <td>3) $E \rightarrow T$</td> <td>$E.val = T.val$</td> </tr> <tr> <td>4) $T \rightarrow T_1 * F$</td> <td>$T.val = T_1.val \times F.val$</td> </tr> <tr> <td>5) $T \rightarrow F$</td> <td>$T.val = F.val$</td> </tr> <tr> <td>6) $F \rightarrow (E)$</td> <td>$F.val = E.val$</td> </tr> <tr> <td>7) $F \rightarrow \mathbf{digit}$</td> <td>$F.val = \mathbf{digit.lexval}$</td> </tr> </tbody> </table>	PRODUCTION	SEMANTIC RULES	1) $L \rightarrow E \mathbf{n}$	$L.val = E.val$	2) $E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$	3) $E \rightarrow T$	$E.val = T.val$	4) $T \rightarrow T_1 * F$	$T.val = T_1.val \times F.val$	5) $T \rightarrow F$	$T.val = F.val$	6) $F \rightarrow (E)$	$F.val = E.val$	7) $F \rightarrow \mathbf{digit}$	$F.val = \mathbf{digit.lexval}$	(Applying)	6
PRODUCTION	SEMANTIC RULES																		
1) $L \rightarrow E \mathbf{n}$	$L.val = E.val$																		
2) $E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$																		
3) $E \rightarrow T$	$E.val = T.val$																		
4) $T \rightarrow T_1 * F$	$T.val = T_1.val \times F.val$																		
5) $T \rightarrow F$	$T.val = F.val$																		
6) $F \rightarrow (E)$	$F.val = E.val$																		
7) $F \rightarrow \mathbf{digit}$	$F.val = \mathbf{digit.lexval}$																		
B)	Consider following SDT to generate Three Address Code. Give Three Address Code for the expression $a+b*c$: <table border="1"> <thead> <tr> <th>Productions</th> <th>Semantic Actions</th> </tr> </thead> <tbody> <tr> <td>$S \rightarrow id=E$</td> <td>{gen(id.name=E.place);}</td> </tr> <tr> <td>$E \rightarrow E_1+T$</td> <td>{E.place=newTemp(); gen(E.place= E₁.place + T.place);}</td> </tr> <tr> <td> T</td> <td>{E₁.place = T.place;}</td> </tr> <tr> <td>$T \rightarrow T_1* F$</td> <td>{T.place=newTemp(); gen(T.place= T₁.place * F.place);}</td> </tr> <tr> <td> F</td> <td>{T₁.place + F.place;}</td> </tr> <tr> <td>$F \rightarrow id$</td> <td>{F.place = id.name;}</td> </tr> </tbody> </table>	Productions	Semantic Actions	$S \rightarrow id=E$	{gen(id.name=E.place);}	$E \rightarrow E_1+T$	{E.place=newTemp(); gen(E.place= E ₁ .place + T.place);}	T	{E ₁ .place = T.place;}	$T \rightarrow T_1* F$	{T.place=newTemp(); gen(T.place= T ₁ .place * F.place);}	F	{T ₁ .place + F.place;}	$F \rightarrow id$	{F.place = id.name;}	(Analyzing)	6		
Productions	Semantic Actions																		
$S \rightarrow id=E$	{gen(id.name=E.place);}																		
$E \rightarrow E_1+T$	{E.place=newTemp(); gen(E.place= E ₁ .place + T.place);}																		
T	{E ₁ .place = T.place;}																		
$T \rightarrow T_1* F$	{T.place=newTemp(); gen(T.place= T ₁ .place * F.place);}																		
F	{T ₁ .place + F.place;}																		
$F \rightarrow id$	{F.place = id.name;}																		
C)	Explain Syntax Tree and DAG as intermediate code representations. Construct the DAG and identify the Value Numbers for the subexpression $a+b+(a+b)$ assuming + associates from the left.	(Remembering) (Applying)	6																
Q. 5	Solve any Two of the following.																		
A)	List the issues in designing code generator. Generate code for the following three-address statements assuming all variables are stored in memory locations. $x = b * c$ $y = a + x$	(Remembering) (Applying)	6																
B)	Define basic block and flow graph. Give basic blocks and flow graph for the following sequence of three address code statements:	(Remembering) (Applying)	6																

	<pre> (1) prod := 0 (2) i := 1 (3) t₁ := 4* i (4) t₂ := a[t₁] /*compute a[i] */ (5) t₃ := 4* i (6) t₄ := b[t₃] /*compute b[i] */ (7) t₅ := t₂*t₄ (8) t₆ := prod+t₅ (9) prod := t₆ (10) t₇ := i+1 (11) i := t₇ (12) if i<=20 goto (3) </pre>		
C)	Explain with examples the following concepts with respect to code optimization: i) Common subexpression elimination. ii) Dead-code elimination. iii) Algebraic transformations.	(Remembering)	6
	*** End ***		

The grid and the borders of the table will be hidden before final printing.