REG.NO: 0245

VIT

Vellore Institute of Technology

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CONTINUOUS ASSESSMENT TEST - II

FALL SEMESTER 2024-2025

SLOT: C2+TC2

Programme Name & Branch : B.Tech

Course Code and Course Name : BCSE307L - Compiler Design

Faculty Name(s): PROF. SAHAAYA ARUL MARY S A, PROF. KANNADASAN R, PROF. VISHNUPRIYA, PROF. VETRISELVI T, PROF. BHUVANESWARI M, PROF. KANAGARAJ R, PROF. KALAIVANI K, PROF. SATHYA K, PROF. BASKARAN P, PROF. SABYASACHI KAMILA, PROF. UMA PRIYA D, PROF. MUKKU NISANTH KARTHEEK, PROF. ARUMUGA ARUN R, PROF. ISLABUDEEN M, PROF. SUGANTHINI C, PROF. NAGA PRIYADARSINI R, PROF. BHAWANA TYAGI, PROF. BAIJU B V, PROF. UMAMAHESWARI M

Class Number(s): VL2024250101546, VL2024250101550, VL2024250101556, VL2024250101591, VL2024250101607, VL2024250101617, VL2024250101627, VL2024250101635, VL2024250101644, VL2024250101655, VL2024250101665, VL2024250101671, VL2024250101674, VL2024250101679, VL2024250101688, VL2024250101729, VL2024250101738, VL2024250102054, VL2024250108000

Date of Examination : 15-Oct-2024

Exam Duration : 90 minutes Maximum Marks: 50

General instruction(s): Answer All Questions:

- Course Outcomes

CO2: Develop language specifications using context free grammars (CFG)

CO3: Apply the ideas, the techniques, and the knowledge acquired for the purpose of developing software systems.

CO4: Constructing symbol tables and generating intermediate code.

CO5: Obtain insights on compiler optimization and code generation

Q. No	Question	M	CO	BL
1.	The grammar is given for parsing simple method calls and variable assignment in java. Check whether the given grammar is LALR(1) grammar or not. **Program→Statement;** **Statement→MethodCall | assignment** **MethodCall→identifier (Argument)** **Argument→identifier | ε** **assignment→identifier=identifier** By using the above grammar parse the following method call **SetBestCollege(VIT);** and show the stack implementation.	10	CO2	BT3
2.	a) Consider the syntax directed translation given by the following grammar and semantic rules. Here N, I, F and B are non-terminals. N is the starting non-terminal, and #,0 and 1 are lexical tokens corresponding to input letters "#", "0" and "1", respectively. X.val denotes the synthesized attribute (a numeric value) associated with a non-terminal X. I_1 and F_1 denote occurrences of I and F on the right hand side of a production, respectively. For the tokens 0 and 1, 0.val=0 and 1.val=1. **N→I#F {N.val=I.val+F.val}** **I→I_1B {I.val=(2 I_1.val)+ B.val }** **I→B {I.val= B.val }** **F→BF_1 {F.val=1/2 (B.val+F_1.val)}** **F→B {F.val =1/2 B.val}** **B→0 {B.val=0.val}** **B→1 {B.val=1.val}** Find the value computed by the translation scheme and Draw an annotated parse tree showing the detailed computation of attribute values for the given string **10#011.**	5	CO3	BT3
	b) Explain how SDT is applied in the construction of the syntax tree for the string:	5		

VIT
Vellore Institute of Technology

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - II
FALL SEMESTER 2024-2025

SLOT: C2+TC2

3*4-7/2. Describe how attributes are passed and computed at each node, focusing on the flow of values.			

Given the input code:
 	
`if ((a>b)&&(a>c)&&(a>d)) { printf(" largest is %d",a);}`
`else if ((b>a)&&(b>c)&&(b>d)) { printf(" largest is %d",b);}`
`else if ((c>a) && (c>b)&&(c>d)) { printf(" largest is %d",c);}`
`else { printf("largest is %d",d);}`
 	
Illustrate the steps of generating the intermediate code with backpatching. Include the backpatching lists and the final generated code.
	10	CO4	BT3

Given the following code snippet, identify the basic blocks, construct a control flow graph (CFG).
 	
`int globalvar;`
`void process(int value) {`
`int localvar;`
`if (value < 0) {`
`throw "Negative value error";`
`} else {`
`for (int i = 0; i < value; i++) {`
`if (i % 2 == 0) {`
`globalvar += i;`
`} else {`
`localvar += i;`
`}}}`
`print (globalvar);`
`print(localvar);`
`}`
 	
`int main() {`
`try {`
`process (5);`
`} catch (const char* e) {`
`print(e);`
`} finally {`
`print("Execution completed.");`
`}`
`}`
	10	CO5	BT3

a) Use the semantic actions for array references and annotate the parse tree for **x = a[b[i][i]][c[k]].**	5	CO4, CO5	BT3

b) Consider the following nested loop in a matrix multiplication function:
 	
`void matrix_multiply(int A[100] [100], int B[100] [100], int C[100] [100]) {`
`for (int i = 0; i < 100; i++) {`
`for (int j = 0; j < 100; j++) {`
`c[i][j] = 0;`
`for (int k=0; k < 100; k++)`
`{ c[i][j] += A[i][k] * B[k][j];`
`}}}`
Identify and discuss three loop optimization techniques that can be applied to improve the performance of this function. Explain how each technique would specifically enhance the execution efficiency. Illustrate the impact of these optimizations on the computational complexity of the function. Would they affect the time complexity, space complexity, or both? Provide a detailed analysis.
	5		