



Daffodil International University
Faculty of Science & Information Technology
Department of Computer Science & Engineering
Final Examination, Summer 2026
Course Code: CSE313, Course Title: Compiler Design
Level: 3 Term: 1 Batch: 67

Time: 02:00 Hrs

Marks: 40

Answer ALL Questions

[The figures in the right margin indicate the full marks and corresponding course outcomes. All portions of each question must be answered sequentially.]

1.	a)	Consider the following grammar to produce an LR(0) parser and canonical table and identify any shift-reduce conflicts, if present. $S \rightarrow W X = Y Z$ $X \rightarrow id \mid var$ $Y \rightarrow W num \mid num \mid W$ $W \rightarrow int \mid float \mid char$ $Z \rightarrow ; \mid ,$	[8]	CO2
	b)	Apply the left- factoring technique to the following grammar to remove ambiguity and make it suitable for a predictive parser. Statement $\rightarrow id = Expression ; \mid id = FunctionCall ;$ FunctionCall $\rightarrow id (Arguments)$	[2]	
2.	a)	Given the following grammar, use predictive parsing techniques to determine the FIRST() and FOLLOW() sets of each non-terminal. Then, use these sets to build the LL(1) parsing table. $S \rightarrow AC \mid BC$ $A \rightarrow S c \mid d$ $B \rightarrow c e \mid f$ $C \rightarrow m \mid n$	[7]	CO2
	b)	Consider the following code segment: <pre> a = b + c; d = b + c; e = d * 2; f = e; if (2 > 1) a = a + 5; else a = a - 5; </pre> Apply appropriate code optimization techniques to optimize the given code and rewrite the optimized code using suitable methods and justify the applied techniques.	[3]	

3.	a)	Consider the following code and answer the questions. <pre> int main() { int i, sum; sum = 0; for(i = 1; i <= 5; i++) { sum = sum + i * 2; } sum = sum + sum + sum + sum - sum + sum + sum + i; } </pre> i. Convert the above code into three address codes (TAC) except the last statement. ii. Represent the TAC into quadruples and triples.	[2+4]	CO3																																										
	b)	Consider the following arithmetic expression: $x - y * (-z) + w \wedge x \oplus y * (-z) * w \wedge x + x - y * (-z)$ i. Construct the Syntax Tree for the expression ii. Construct a Directed Acyclic Graph (DAG) for the expression	[2+2]																																											
4.	a)	Consider the following Three Address Code: <table border="1"> <tr> <td>1: i = 0</td> <td>15: A[i] = t3</td> <td>29: z = z + 1</td> </tr> <tr> <td>2: s = 0</td> <td>16: goto L2</td> <td>30: s = s * 2</td> </tr> <tr> <td>3: x = 10</td> <td>17: L5: t3 = t1 - t2</td> <td>31: C[j] = z</td> </tr> <tr> <td>4: y = 20</td> <td>18: x = x + 2</td> <td>32: j = j + 1</td> </tr> <tr> <td>5: z = 30</td> <td>19: y = y - 2</td> <td>33: goto L2</td> </tr> <tr> <td>6: L1: if i >= 100 goto L4</td> <td>20: B[i] = t3</td> <td>34: L3: i = i + 1</td> </tr> <tr> <td>7: t1 = x * 2</td> <td>21: goto L2</td> <td>35: x = x + z</td> </tr> <tr> <td>8: t2 = y / 2</td> <td>22: L2: if j >= 10 goto L3</td> <td>36: y = y - z</td> </tr> <tr> <td>9: j = 0</td> <td>23: t4 = C[j]</td> <td>37: goto L1</td> </tr> <tr> <td>10: k = i + 5</td> <td>24: t5 = t4 * x</td> <td>38: L4: res = s + x</td> </tr> <tr> <td>11: if k > 50 goto L5</td> <td>25: s = s + t5</td> <td>39: res = res - y</td> </tr> <tr> <td>12: t3 = t1 + t2</td> <td>26: t6 = s - y</td> <td>40: return res</td> </tr> <tr> <td>13: x = x + 1</td> <td>27: D[j] = t6</td> <td></td> </tr> <tr> <td>14: y = y - 1</td> <td>28: if t6 < 0 goto L3</td> <td></td> </tr> </table> Answer the following question: i. Which lines in the code qualify as leaders by leader selection rules? ii. Is / Are there any instruction(s) designated as leaders more than twice? iii. What is the total number of basic blocks identified in the code?	1: i = 0	15: A[i] = t3	29: z = z + 1	2: s = 0	16: goto L2	30: s = s * 2	3: x = 10	17: L5: t3 = t1 - t2	31: C[j] = z	4: y = 20	18: x = x + 2	32: j = j + 1	5: z = 30	19: y = y - 2	33: goto L2	6: L1: if i >= 100 goto L4	20: B[i] = t3	34: L3: i = i + 1	7: t1 = x * 2	21: goto L2	35: x = x + z	8: t2 = y / 2	22: L2: if j >= 10 goto L3	36: y = y - z	9: j = 0	23: t4 = C[j]	37: goto L1	10: k = i + 5	24: t5 = t4 * x	38: L4: res = s + x	11: if k > 50 goto L5	25: s = s + t5	39: res = res - y	12: t3 = t1 + t2	26: t6 = s - y	40: return res	13: x = x + 1	27: D[j] = t6		14: y = y - 1	28: if t6 < 0 goto L3		[3*2]	CO3
1: i = 0	15: A[i] = t3	29: z = z + 1																																												
2: s = 0	16: goto L2	30: s = s * 2																																												
3: x = 10	17: L5: t3 = t1 - t2	31: C[j] = z																																												
4: y = 20	18: x = x + 2	32: j = j + 1																																												
5: z = 30	19: y = y - 2	33: goto L2																																												
6: L1: if i >= 100 goto L4	20: B[i] = t3	34: L3: i = i + 1																																												
7: t1 = x * 2	21: goto L2	35: x = x + z																																												
8: t2 = y / 2	22: L2: if j >= 10 goto L3	36: y = y - z																																												
9: j = 0	23: t4 = C[j]	37: goto L1																																												
10: k = i + 5	24: t5 = t4 * x	38: L4: res = s + x																																												
11: if k > 50 goto L5	25: s = s + t5	39: res = res - y																																												
12: t3 = t1 + t2	26: t6 = s - y	40: return res																																												
13: x = x + 1	27: D[j] = t6																																													
14: y = y - 1	28: if t6 < 0 goto L3																																													
	b)	Draw the flow graph for the above-mentioned instructions.	[4]																																											