ourdiu.com | Shared by Rabiul Awayl

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 → W X = Y Z X → id | var Y → W num | num | W W → int | float | char Z → ; | ,	[8]	CO2
	b)	Apply the left- factoring technique to the following grammar to remove ambiguity and make it suitable for a predictive parser. Statement → id = Expression ;| id = FunctionCall ; FunctionCall → 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 → AC | BC A → S c | d B → c e | f C → m | n	[7]	CO2

	b)	Consider the following code segment:
		a = b + c;
		d = b + c;
		e = d * 2;
		f = e ;
		if (2 > 1)
		  a = a + 5;
		else
		  a = a - 5;
		Apply appropriate code optimization techniques to optimize the given code and rewrite the optimized code using suitable methods and justify the applied techniques.
			[3]	

ourdiu.com | Shared by Rabiul Awayl

3. *a)* Consider the following code and answer the questions. **[2+4]** **CO3**

```
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;
}
```

i. Convert the above code into three address codes (TAC) except the last statement.
ii. Represent the TAC into quadruples and triples.

b) Consider the following arithmetic expression: **[2+2]**

x - y * (-z) + w ^ x - y * (-z) * w ^ x + x - y *(-z)

i. Construct the Syntax Tree for the expression
ii. Construct a Directed Acyclic Graph (DAG) for the expression

4. *a)* Consider the following Three Address Code: **[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	

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?

b) Draw the flow graph for the above-mentioned instructions. **[4]**