Register Number | 1 | 9 | 5 | 0 | 0 | 1 | 0 | 7 | 6 |

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110

(An Autonomous Institution, Affiliated to Anna University, Chennai)

Department of Computer Science and Engineering

Continuous Assessment Test – I

Question Paper

Degree & Branch	**BE (CSE)**				**Semester**	VI
Subject Code & Name	**UCS1602 – Compiler Design**				**Regulation:**	**2018**
Academic Year	2021-2022	**Batch**	2019-2023	**Date**	**31-03-2022**	FN
Time: 90 Minutes 8.30 – 10.00 am	**Answer All Questions**				**Maximum: 50 Marks**	

Part – A (6×2 = 12 Marks)

<KL3>	1.Estimate the correct number of LOC(lines of code) after applying appropriate code optimization techniques for the given three address code. t1=t1*30 t2=t1+0 t3=t2+c t4=t3	<CO1>
<KL1>	2. What is the correct sequence of processes involved in program execution?	<CO1>
<KL2>	3. Illustrate the use of the global variables yytext, yyleng and yylval used in LEX with examples.	<CO1>
<KL3>	4. Consider a language L generates the following: It starts with $ followed by float values with both whole number and fractional part. eg. $1234.56 It can start with $ followed by integer values. e.g $56 It can start with $ followed by float values with only fractional part e.g $.45 Construct a regular expression to generate L.	<CO1>
<KL3>	5. Consider a regular expression (a/b)*abb(a/b)*. Let the follow position table be Node followpos 1 1,2,3 2 1,2,3 3 4 4 5 5 6,7,8 6 6,7,8 7 6,7,8 8 – Apply DFA construction algorithm to find the next state for the input symbol 'a' from a state {1,2,3,5}.	<CO1>
<KL2>	6. Explain ambiguous grammar with an example.	<CO2>

Part – B (3×6 = 18 Marks)

<KL2>	7. Explain the phases of a compiler. Illustrate the output of each phase for the following code segment. **int a,b;** **float c,d;** **d=a+c*b-20;**	<CO1>
<KL2>	8. Write a LEX specification to recognize the identifier, numeric constants including fraction and exponentiation, keywords and operators.	<CO1>
<KL2>	9. Show that the grammar G1 is not suitable for implementing top down parser. Rewrite the grammar to overcome this problem. G1: A → AB1 | B0 | 1 B → B1 | A0 | 0	<CO2>

Part – C (2×10 = 20 Marks)

<KL3>	10. Apply direct method to construct DFA for the regular expression ((a|c)*)ac(ba)*	<CO1>
	(OR)	
<KL3>	11. Apply direct method to construct DFA for the regular expression (a|b|c)*(a|b)*.	<CO1>

<KL3>	12. Construct parse tree for the sentence S using grammar G.
	Grammar G:
	S->if E then S | if E then S else S | while E do S | begin L end | AS
	L -> L S | S
	E -> E R E | E A E | id
	R -> < | <= | > | >= | != | ==
	A -> + | - | * | / | %
	AS -> AS=E | id
	Sentence S:
	begin
	while a > b do
	begin
	x = y + z
	a = a - b
	end
	x = y - z
	end
		<CO2>

	(OR)	
<KL3>	13. Construct recursive descent parser for the grammar G. Parse the string **id / id - (id - id)** G: E →E – T | T T → T / F | F F → (E) | id	<CO2>
