Roll No							

End Semester Examination 2024

Name of the Course: B. Tech. *Semester:* 6th

Name of the Paper: Compiler Design *Paper Code: TCS-601*

Time: 3 Hour's Maximum Marks: 100

Note:

(i) All Questions are compulsory.
(ii) Answer any two sub questions among a,b and c in each main question.
(iii) Total marks in each main question are twenty.
(iv) Each question carries 10 marks.

Q1	(10 X2 = 20 Marks)	
(a)	Illustrate the structure of a compiler. Also Explain in detail the process of compilation for the statement: **a=b+c*70.**	
(b)	Explain the following with suitable example. i. Bootstrapping ii. CFG iii. Token with their types iv. Single pass and Multi-pass compiler.	
(c)	With suitable example explain the problems arises by left recursive and non-deterministic grammar. Remove the left recursion (if any) from the following grammar productions: **i. S → Aa/b** **A → Sd/c** **B →cd/e** **ii. S → Sda/ca/b** **S → Sab/Scd/a/b/c** **A → Aa/Acd/d/a** **iii. E → E - T/T** **T → T * F/F** **F → (E)/a/b/c/id**	CO1
Q2	**(10 X2 = 20 Marks)**	
(a)	What are the steps to be followed in order to create LL(1) parsing table? To verify whether the given grammar is LL(1) or not? **S → (L)/a** **L → L,S/S**	
(b)	Explain operator precedence parser. Consider the following grammar: **E → E + E** **E → E * E** **E → id** By using operator relation table create the parse tree for the input string **"2+3*4"** where precedence order is defined as **"id > * > + > $" and +, *** both are having left associative.	CO2

(c)	Consider the following grammar: **S→ Aa/bAc/dc/bda** **A→d** Construct the canonical collections of LR(1) items and to verify whether the given grammar is CLR(1) or not?	
Q3	**(10 X2 = 20 Marks)**	
(a)	List out the applications of SDT. Consider the following SDT schemes for a simple desk calculator. PRODUCTION — SEMANTIC RULES L→ E — { L.val= E.val; } E→ E + T — { E.val= E.val + T.val; } E→ T — { E.val= T.val; } T→ T * F — { T.val= T.val * F.val; } T→ F — { T.val= F.val; } F→ (E) — { F.val= E.val; } F→ digit — { F.val= digit.lexval; } By using the above SDT, construct the parse tree for the arithmetic expression "**3*4*5+6**" and also compute its **L.val.**	**CO3**
(b)	What do you mean by error handler? Illustrate the various types of errors detected by the error handler at the lexical and syntax analysis phase of compiler. Also illustrate some error handling techniques with an appropriate example.	
(c)	Briefly explain the various types of storage allocation strategies with suitable example.	
Q4	**(10 X2 = 20 Marks)**	
(a)	What do you mean by back patching? Generate the three address code for the following switch-case statement using back patching: **switch(i+j)** **{** **case (1): if(a<b)** **{ L=M+N*P }** **else { X=A+B*C }** **break;** **case (2): X=A+B*C** **break;** **default: X=(a+b)*(c+d)** **break;**	**CO4**
(b)	List out some benefits of having an intermediate code generated in compiler design. For the given high level expression write quadruple and triples. **a= -(x+y-z)+(b+c*d)*(x+y+g-h)**	

(c)	What do you mean by code optimization? Consider the given code snippet in a high-level programming language. According to you, what are the best suitable code optimization techniques you can apply to optimize the given code? Give the explanation for your answer.
	`#include <stdio.h>`
	`int main() {`
	`int a = 10;`
	`int b = 5;`
	`int X;`
	`int Z;`
	`for(i=1;i<=100;i++)`
	`{`
	`X= a*b+i;`
	`}`
	`printf("The output is: %d\n", X);`
	`return 0;`
	`}`	

Q5	(10 X2 = 20 Marks)	
(a)	What is an activation record? Draw diagram of general activation record and explain the purpose of different fields of an activation record.	

(b)	Consider the following source program.
	`fact(n)`
	`{`
	`int f=1;`
	`for(i=2; i≤n; i++)`
	`f=f*i;`
	`return f;`
	`}`
	For the above high level instructions, construct the program flow graph using control flow analysis and find out the number of basic blocks and number of leaders.
		CO5

(c)	What is common sub-expression elimination? Explain it with suitable example. Construct the DAG for the following three address statements. 1) T1= 4*i 2) T2= a[T1] 3) T3= 4*i 4) T4= b[T3] 5) T5= T2*T4 6) T6= prod + T5 8) T7= i + 10 9) X= T7+B 10) Y=X	