# MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL

**Paper Code : PCC-CS501 Compiler Design**

**UPID : 005506**

*Time Allotted : 3 Hours* *Full Marks :70*

*The Figures in the margin indicate full marks.*
*Candidate are required to give their answers in their own words as far as practicable*

## Group-A (Very Short Answer Type Question)

1. Answer *any ten* of the following : [ 1 x 10 = 10 ]

(I) What is postfix SDT?

(II) What is equivalence of type expression?

(III) The actual parameters are not used by the calling procedure. (true/false)

(IV) "goto L" is an unconditional jump(to L) three address instruction.(true/false)

(V) A basic block is a sequence of consecutive statements with single entry/single exit. (true/false)

(VI) Flex is a lexical analyzer generator tool. (True/False)

(VII) Write the name of translator that translates assembly code to relocatable machine code.

(VIII) 12_Name is a lexeme of pattern Identifier in C language. (True/False)

(IX) Write the rule for converting left recursive grammar to right recursive grammar.

(X) What is Annotated -parse tree?

(XI) Reduce/Reduce conflict happens during bottom up parsing. (True/False)

(XII) How lexical analyzer recognize a token?

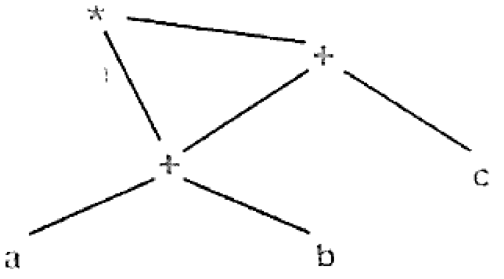
## Group-B (Short Answer Type Question)

Answer *any three* of the following : [ 5 x 3 = 15 ]

2. Write the difference between synthesized and inherited attributes with examples. [5]

3. Describe the purpose of two pointers of Buffer Pairs in Lexical Analyzer. [5]

4. Convert the DAG into Three address code. [5]



**Directed Acyclic Graph**

5. Consider the postfix SDT: Here expr, expr$_1$ both are same, and for differentiate between left expr and right expr, we use expr$_1$ in right. [5]

expr → expr$_1$ + term { print(' + ') }

expr → expr$_1$ - term { print(' - ') }

expr → term

term → 0 { print(' 0 ') }

term → 1 { print(' 1 ') }

.......................

term → 9 { print(' 9 ') }

Draw the parse tree with action embedded for the expression **9 + 4 - 3.**

6. Convert the C code into three address instruction. [5]
while( A[i] ≥ v ){ i = i + 1; } where array elements are integers of 4 bytes in sized.

## Group-C (Long Answer Type Question)

Answer *any three* of the following : [ 15 x 3 = 45 ]

7. (a) Suppose ε-closure(q) is a set of states which are reachable from q with zero or more ε-moves, where q is a state in NFA. You are given a Regular Expression R = (b|a)*baa and the set of input symbols is {a,b}U{ ε }. Convert this Regular Expression R to NFA N. [ 3 ]

(b) Convert this NFA N to DFA D using the definition of ε-closure(q). [ 7 ]

(c) Convert this DFA D to Minimal DFA. [ 5 ]

8. (a) Suppose you have given a grammar of certain kind of statements and first & follow sets: [ 5 ]

A→B A' ; A'→ + B A' | ε ; B→ C B' ; B'→ * C B' | ε ; C→ (A) | id

| Non-terminals | First sets | Follow sets |
|---|---|---|
| A | ( , id | ), # |
| A' | + , ε | ), # |
| B | ( , id | +, ), # |
| B' | * , ε | +, ), # |
| C | ( , id | +, *, ), # |

Where # is an end marker representing the end of input string.

Build the predicting parsing table for the above grammar.

(b) Consider a new set for error recovery of predicting parsing, called synchronizing set of non-terminal A, is a set where each symbol of synchronizing set of non-terminal A is taken from follow(A) set. I.e synchronizing(C) = follow(C) = {+, *, ), #}. [ 5 ]

Instead of writing "error" in M[A, a] in parsing table, you use "syn" in that cell if a belongs to follow(A).

Example: If M[ A, ) ] = "error" in table M, then you use "syn" in M[ A, ) ] cell of M, since ")" belongs to follow(A).

The solution of (b) can be obtained from the following rules given below:

i) If the parser looks up entry M[ A, a ] and finds that is "error", then the input symbol a is skipped.

ii) If the entry is "sync", then it skip symbols from input until a terminal symbol is seen which belongs to first(A) to continue parsing, if A is the top of the STACK.

iii) If a token on top of the STACK does not match the input symbol, then you pop the token and resume parsing.

Build an error correcting non-recursive version of predicting parsing table for the above grammar.

(c) From the solution of (b), show the behavior of your parser on the following input: [ 5 ]

| Sl no | STACK | INPUT | Behavior/Action |
|---|---|---|---|
| 1. | #A | )id * + id# | ............ |
| 2. | ........... | .............. | ............... |

9. Consider the grammar with productions: [ 15 ]

S → A a | b A c | d c | b d a; A → d

Prove that the above grammar is CLR(1) but not SLR(1) by building the parsing table.

10. (a) Suppose you have been given the three address code of Quick Sort Algorithm [ 10 ]

| Three address code for Quick Sort | | | | | |
|---|---|---|---|---|---|
| 1 | i:= m–1 | 11 | t5:= a[t4] | 21 | a[t10]:= x |
| 2 | j:= n | 12 | If t5 > v goto (9) | 22 | goto (5) |
| 3 | t1:= 4*n | 13 | If i >= j goto (23) | 23 | t11:= 4* i |
| 4 | v:= a[t1] | 14 | t6:= 4*i | 24 | x:= a[t11] |
| 5 | i:= i+1 | 15 | x:= a[t6] | 25 | t12:= 4*i |
| 6 | t2:= 4*i | 16 | t7:= 4*i | 26 | t13:= 4*n |
| 7 | t3:= a[t2] | 17 | t8:= 4*j | 27 | t14:= a[t13] |
| 8 | If t3 < v goto (5) | 18 | t9:= a[t8] | 28 | a[t12]:= t14 |
| 9 | j:= j-1 | 19 | a[t7]:= t9 | 29 | t15:= 4*n |
| 10 | t4:= 4*j | 20 | t10:= 4*j | 30 | a[t15]:= x |

Find the leader codes and basic blocks including three address code of the above code.

(b) Build the flow graph for above code. [ 5 ]

11. (a) Describe in brief about the cousin of Compiler. [ 5 ]

(b) Describe the operation of different phases of Compiler with suitable example. [ 10 ]