Roll No. _____ **5th May 2025**

UCSE601: Compiler Design Time: **3 hours,** MM:**(Maximum Marks): 100**

Faculty: Dr. Ranjan Maity

Central Institute of Technology Kokrajhar

Computer Science & Engineering Department

END TERM EXAMINATION

Instructions: Attempt Q1 and Q2 and any two from the rest.

---

1. a) What type of derivation is used in LL and LR parsers?

   b) Match the following -

List1	List2
A: Buttom Up Parsing	P: Finite Automata
B: Lexical Analysis	Q: DAG
C: Runtime Environment	R: Augmented Grammar
D: Code Optimization	S: Activation Record

   c) Consider the following C Code -
   A= B+C
   D= A+B+C
   E=A+D
   What will be the number of nodes and edges in the DAG?

   d) Identify the tokens -
   printf("The %d is greater than %d", x,y);

   e) Consider the following grammar
   $P \longrightarrow pPQ|r$
   $Q \longrightarrow q$
   What number of reduction steps in a bottom-up parser to accept the string *ppprqqq* will be?

   f) Consider the following SDT:
   $P \longrightarrow pQ$ *print a*
   $P \longrightarrow p$ *print b*
   $Q \longrightarrow Pr$ *print c*
   For the given input *ppr* in a bottom-up parser, determine the output.

   g) Consider the following grammar
   $E \longrightarrow E{+}id \mid E{\times}\ id \mid id$
   for a given input $id{+}id{\times}id$, identify the handles in right sentential forms.

   h) Consider the SLR(1) parser has M states and LALR(1) constructed from it has N states. What will be the relations between M and N?

   i) Remove left recursion for the given grammar
   $P \longrightarrow PQrst|rst$

   j) Remove left factoring for the following grammar
   $P \longrightarrow pqrP|pqrt$ [10×2 = 20 marks]

2. a) Consider the following grammar G and convert it as an augmented grammar

$S \implies A + B|B$

$A \implies A * B|A|id$

$B \implies A - B|B|id$

b) Commpute the **FIRST** and **FOLLOW** of the grammar obtained in the previous step.

c) Construct the LR(1) items.

d) Construct the Canonical LR parsing table.

e) Design a LALR parsing table.

f) Use your parsing table to check whether *id+id+id-id* is accepted by your LALR parser.

[(2+8+10+10+5+5) = 40 marks]

---

3. a) Consider the following code and identify the leaders

1001: *i = 1*
1002: *j = 1*
1003: *t1 = 10*i*
1004: *t2 = t1+j*
1005: *t3 = 8*t2*
1006: *t4 = t3-88*
1007: *a[t4] = 0.0*
1008: *j = j+1*
1009: *if j <= 10 goto 1003*
1010: *i = i+1*
1011: *if i <= 10 goto 1002*
1012: *i = 1*
1013: *t5 = i-1*
1014: *t6 = 88*t5*
1015: *a[t6] = 1.0*
1016: *i = i+1*
1017: *if i <= 10 goto 1013*

b) Construct the basic blocks and design the flow graph of the previous problem mentioned in Question 3. a.

c) Represent the following instruction in three-address and two-address code - *d=a+b-c×e*

[(3+7+(5+5)) = 20 marks]

[**PTO**]

4. a) Consider the following grammar and derive the string *aaab* using leftmost and rightmost derivation -
   $S \Longrightarrow ABP$
   $A \Longrightarrow AB|a|Null$
   $B \Longrightarrow AB|b|Null$
   $P \Longrightarrow p|Null$
   b) Check whether the grammar is in **LL(1)** or not.
   c) Design a NFA for the regular expression $(a^*b)^*abc$ and convert the NFA to DFA using **McNaughton-Yamada-Thompson** algorithm.

[(5+5+10) = 20 marks]

---

5. Write briefly on - any four

a) Handle pruning.
b) Regular expression for tokens.
c) Type checking.
d) Dead code elimination.
e) Lex program to identify identifiers.

[(4×5) = 20*marks*]
