Student Name: ——————————————— **ID:** ———————————

Note: No marks will be given if the justification for your answer is not provided. Tick only the correct answer for MCQs. Answer in the space provided for problem solving and discriptive questions. Write your Name and ID on each sheet of paper.

1. The number of tokens in the following C statement is ———   `1`

   ```
   printf("i = %d, &i = %x", i, &i);
   ```

2. Consider the following statements related to compiler construction :   `1`

   I. Lexical Analysis is specified by context-free grammars and implemented by pushdown automata.
   II. Syntax Analysis is specified by regular expressions and implemented by finite-state machine.

   Select the correct option:
   (A) Both the statements are correct
   (B) I statement is correct, II is incorrect.
   (C) II statement is correct, I is incorrect.
   (D) Both the statements are incorrect.

3. What is the difference between yylex() and scanf()?   `1`

   Ans.

4. A lexical analyzer uses the following patterns to recognize three tokens T1, T2, and T3 over the alphabet (a,b,c).   `1`

   T1: a?(b|c)*a
   T2: b?(a|c)*b
   T3: c?(b|a)*c

   Note that "x?" means either x occurs once or doesn't occur at all.

   If the string `bbaacabc` is processes by the analyzer, which one of the following is the sequence of tokens it outputs?

   (A) T1T2
   (B) T2T1
   (C) T1T3
   (D) T3T3

5. What does yymore() do in lex?   `1`

   (A) It points to the matched text
   (B) It changes to a new lexical state

(C) It returns all but the first n characters of the current token back to the input stream

(D) It tells the scanner that the next time it matches a rule, the corresponding token should be appended onto the current value of yytext rather than replacing it.

6. Consider line number 3 of the following C- program $\boxed{1}$

```
int main ( ) {              /* Line 1 */
  int I, N;                 /* Line 2 */
  fro (I = 0, I < N, I++);  /* Line 3 */
}
```

Identify the compilerâĂŹs response about this line while creating the object-module

(A) No compilation error

(B) Only a lexical error

(C) Only syntactic errors

(D) Both lexical and syntactic errors

7. If lex .1 is a lex code then $\boxed{1}$

(A) The command lex lex .1 invokes lex to act on lex .1

(B) The command lex lex.1 writes its output to the file lex.yy.c

(C) lex.yy.c has the definition of the function yylex

(D) All of these

8. YACC builds up: $\boxed{1}$

(A) SLR parsing table

(B) canonical LR parsing table

(C) LALR parsing table

(D) none of these

9. Consider the following expression grammar.

```
E-> number
| E '+' E
| E 'ÃŮ' E
```

The above grammar is fed to a yaac tool for parsing and evaluating arithmetic expressions. Which one of the following is true about the action of yaac for the given grammar? $\boxed{2}$

(A) It detects recursion and eliminates recursion

(B) It detects reduce-reduce conflict, and resolves

(C) It detects shift-reduce conflict, and resolves the conflict in favor of a shift over a reduce action

(D) It detects shift-reduce conflict, and resolves the conflict in favor of a reduce over a shift action

10. Build an NFA for the following regular expression using Thompson construction method mechanically without skipping any step. You MUST name the states as 0,1,2,3,.... $\boxed{5}$

    `p(q|r|s)*qq(t|u)v*`

11. From the following regular expression construct a DFA directly (using firstpos, lastpos and followpos), without skipping any step. Show every intermediate data structure, tables, and their values while creating the DFA. **5**

   `(a|b)*ba`

12. Consider the following CFG, where the set of terminals is $\hat{I}\check{c} = \{a, b, \#, \%, !\}$ and the set of non-terminals is $V = \{S,T,U\}$

$S \rightarrow \%aT \mid U\,!$

$T \rightarrow aS \mid baT \mid \epsilon$

$U \rightarrow \#aT\,U \mid \epsilon$

Partial marks will be given for each and every step.

(a) Construct the FIRST sets for each of the non-terminals. **2**

S	{ %, #, ! }
T	{ a, b, ε }
U	{ #, ε }

(b) Construct the FOLLOW sets for each of the non-terminals. **2**

S	{ $, #, ! }
T	{ $, #, ! }
U	{ ! }

(c) Construct the LL(1) parsing table for the grammar. **3**

	a	b	#	%	!	$
S			S → U !	S → %aT	S → U !	
T	T → aS	T → baT	T → ε		T → ε	T → ε
U			U → #aT U		U → ε	

(d) Show the sequence of stack, input and action configurations that occur during an LL(1) parse of the string âĂIJ#abaa%aba!âĂİ. At the beginning of the parse, the stack should contain a single S. | **3** |

Stack	Input	Action