BITS, Pilani- K K Birla Goa Campus

Course: CS F363 Compiler construction | **Semester**-II, 2018-19**, Class Test 1 Marks: 30** | **Weightage:** 15% **Mode:** Open Book | Date: 04/03/2019 | Time: 7 PM to 8 PM | Duration: 60 min

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 _____ **2**

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

Ans.

Solution: Ans. 10 In a C source program, the basic element recognized by the compiler is the âĂIJtoken.âĂİ A token is source-program text that the compiler does not break down into component elements. There are 6 types of C tokens : identifiers, keywords, constants, operators, string literals and other separators. There are total 10 tokens in the above printf statement. Below are tokens in above program.

```
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.

Student Name : _____ **ID:** _____

Solution: (D) Both statements are incorrect.

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

Ans.

Solution: yylex() is used to accept input and call parser, but scanf() for only accepting data.

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

Solution: (D)

Explanation: 0 or 1 occurrence of the symbol x.

T1 : (b+c)* a + a(b+c)* a T2 : (a+c)* b + b(a+c)* b T3 : (b+a)* c + c(b+a)* c

Given String : bbaacabc
Longest matching prefix is âĂİ bbaac âĂİ (Which can be generated by T3)
The remaining part (after Prefix) âĂIJabcâĂİ (Can be generated by T3)

So, the answer is 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.

Student Name : _____ **ID:** _____

Solution: (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 **1**

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

Identify the compiler's 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

Solution: (C) There is âĂŸfroâĂŹ instead of âĂŸforâĂŹ. This is not a lexical error as lexical analysis typically involves Tokenization.

7. If lex .1 is a lex code then **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

Solution: (D) All of these

8. YACC builds up: **1**

(A) SLR parsing table
(B) canonical LR parsing table
(C) LALR parsing table
(D) none of these

Solution: (C) LALR parsing table

9. Consider the following expression grammar.

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

Student Name : _____ **ID:** _____

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? **1**

(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

Solution: (C) In a shift/reduce conflict, the default is to do the shift.

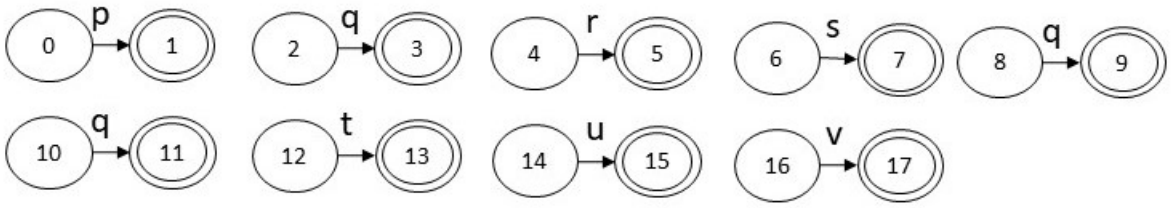
Student Name : _____ **ID:** _____

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,.... **5**

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

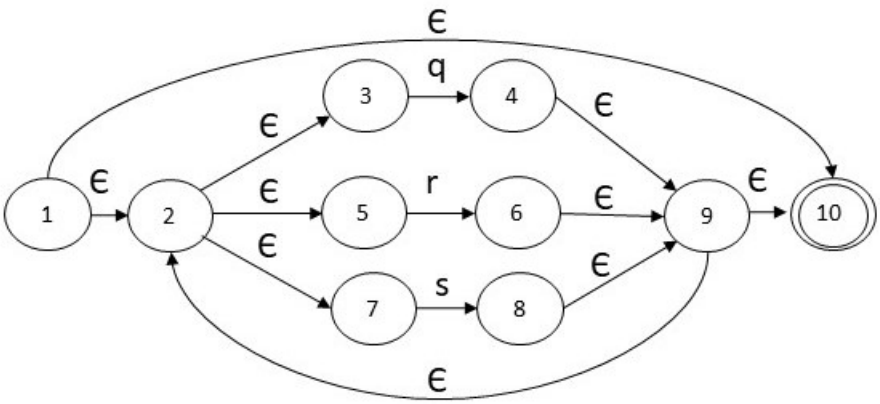
Solution:

Step 1

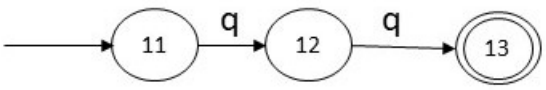


Step 2

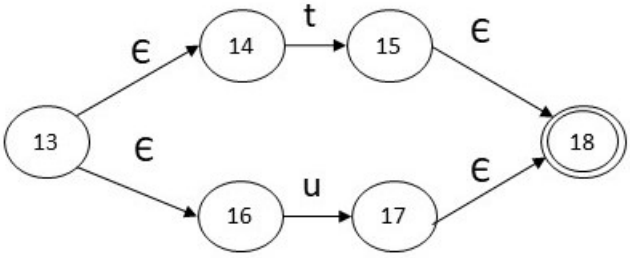
(q|r|s)*



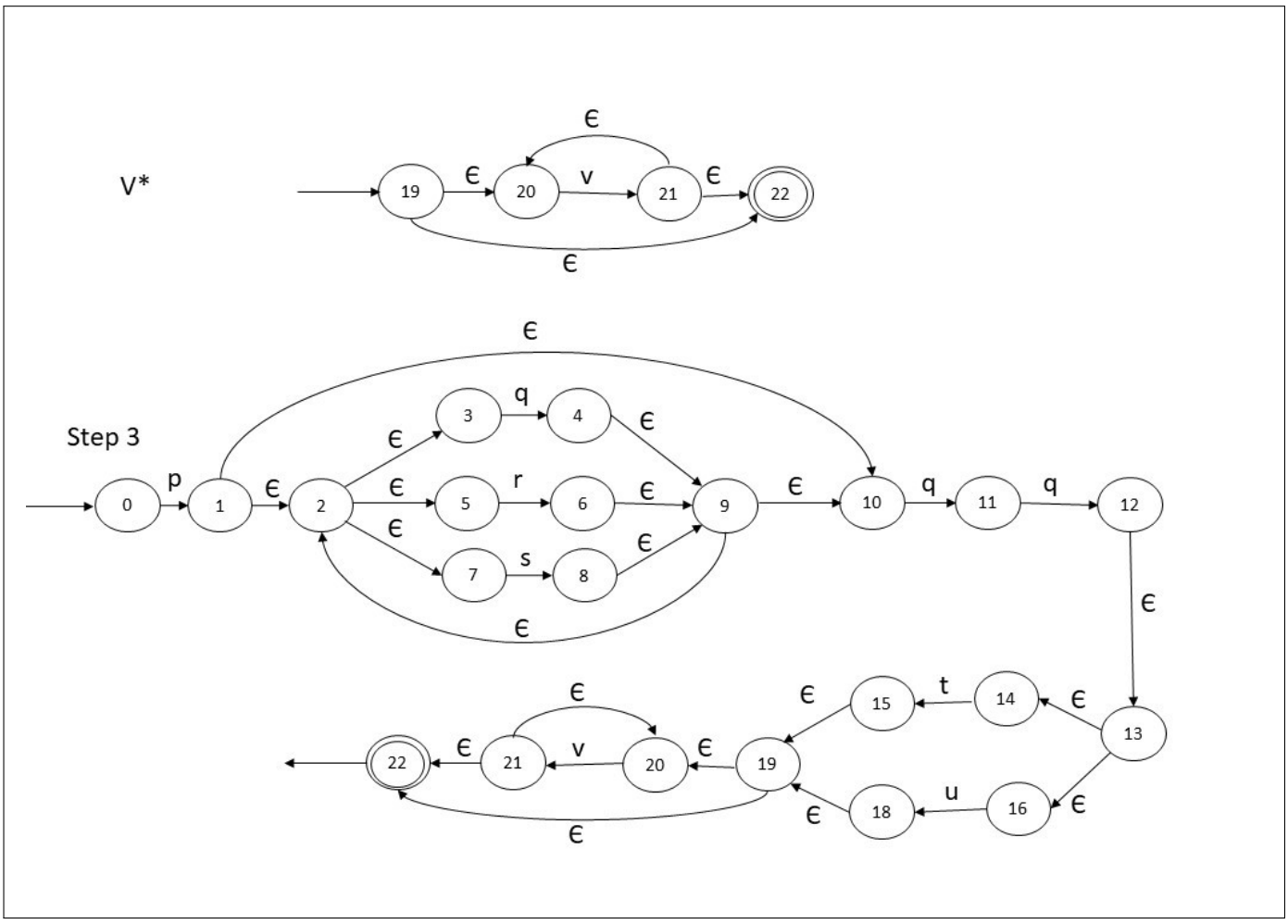
qq



t|u



Student Name : _____ **ID:** _____



Student Name : _____ **ID:** _____

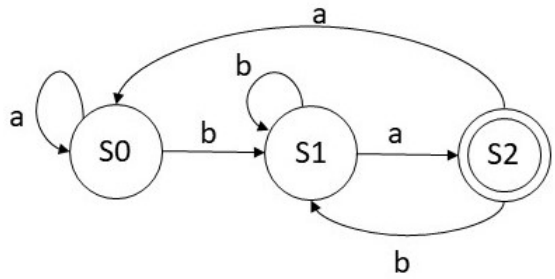
11. From the following regular expression construct a DFA directly (using firstpos, lastpos and followpos), without skipping any step. Show every intermediate data structure, tables, and their values while creating the DFA. **5**

`(a|b)*ba`

Solution:

Partial marks will be given for each and every steps.

Final DFA



Student Name : _____ **ID:** _____

12. Consider the following CFG, where the set of terminals is Îč = {a, b, #, %, !} and the set of non-terminals is V = {S,T,U}

    S → %aT | U !

    T → aS | baT | ϵ

    U → #aT U | ϵ

    Partial marks will be given for each and every step.

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

S	
T	
U	

Solution:

S	{%, #, !}
T	{a, b, ε }
U	{#, ε}

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

S	
T	
U	

Solution:

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

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

	a	b	#	%	!	$
S						
T						
U						

Student Name : _____ **ID:** _____

Solution:

	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	

Student Name : _____ **ID:** _____

(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

Solution:

Student Name : _____ **ID:** _____

Stack	Input	Action
S$	#abaa%aba!$	output S → U !
U !$	#abaa%aba!$	output U → #aT U
#aT U !$	#abaa%aba!$	match #
aT U !$	abaa%aba!$	match a
T U !$	baa%aba!$	output T → baT
baT U !$	baa%aba!$	match b
aT U !$	aa%aba!$	match a
T U !$	a%aba!$	output T → aS
aSU !$	a%aba!$	match a
SU !$	%aba!$	output S → %aT
%aT U !$	%aba!$	match %
aT U !$	aba!$	match a
T U !$	ba!$	output T → baT
baT U !$	ba!$	match b
aT U !$	a!$	match a
T U !$	!$	output T → ε
U !$	!$	output U → ε
!$	!$	match !
$	$	accept

END