EPT-TEST- 56(CD, Lexical, Parsers)
Total Questions: 15
Time: 60 Minutes

Q1. [MCQ]

Consider the following ANSI C program:

```
int main ()
{
    Integer x;
    return 0;
}
```

Which one of the following phases of C compiler will throw an error?(If any tie in answer select first phase in compiler phases)

A. Lexical analyzer
B. Syntax analyzer
C. Semantic analyzer
D. Machine dependent optimizer

Q2. [MCQ]

Which of the following error will not come under Lexical error,

I. Spelling error
II. Exceeding length of an identifier or numeric constants
III. Appearance of illegal characters
IV. Missing semicolon

A. I , II only

B. II , III only

C. I, II, III only

D. 1, IV only

Q3. [MCQ]

Consider the following statements related to compiler construction :

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 a finite-state machine.

Which of the above statement(s) is/are correct ?

A. Only I
B. Only II
C. Both I and
D. None

Q4. [MCQ]

The number of tokens in the following C statement is printf("i=%d, &i=%x", i,&i);

A. 13
B. 6
C. 10
D. 9

Q5. [MCQ]

What is the maximum number of reduce moves that can be taken by a bottom-up parser for a grammar with no epsilon and unit production(i.e., of type A -> ϵ and A -> a)

to parse a string with N tokens?

A. N/2
B. N-1
C. 2N-1
D. 2^N

Q6. [MCQ]

For the grammar below, a partial LL(1) parsing table is also presented along with the grammar. Entries that need to be filled are indicated as E1, E2, and E3.

epsilon(∈) **is the empty string, $ indicates end of input, and, | separates alternate right hand sides of productions.**

S → a A b B | b A a B | ε
A → S
B → S

	a	b	$
S	**E1**	**E2**	S → ε
A	A → S	A → S	error
B	B → S	B → S	**E3**

(A) FIRST(A) = {a, b, ε} = FIRST(B)
FOLLOW(A) = {a, b}
FOLLOW(B) = {a, b, $}

(B) FIRST(A) = {a, b, $}
FIRST(B) = {a, b, ε}
FOLLOW(A) = {a, b}
FOLLOW(B) = {$}

(C) FIRST(A) = {a, b, ε} = FIRST(B)
FOLLOW(A) = {a, b}
FOLLOW(B) = ∅

(D) FIRST(A) = {a, b} = FIRST(B)
FOLLOW(A) = {a, b}
FOLLOW(B) = {a, b}

Q7. [MCQ]

The grammar S → aSa | bS | c is

A. LL(1) but not LR(1)
B. LR(1)but not LL(1)
C. Both LL(1)and LR(1)
D. Neither LL(1)nor LR(1)

Q8. [MCQ]

Match all items in Group 1 with correct options from those given in Group 2.

Group 1	Group 2
P. Regular expression	1. Syntax analysis
Q. Pushdown automata	2. Code generation

R. Dataflow analysis 3. Lexical analysis

S. Register allocation 4. Code optimization

A. P-4. Q-1, R-2, S-3
B. P-3, Q-1, R-4, S-2
C. P-3, Q-4, R-1, S-2
D. P-2, Q-1, R-4, S-3

Q9. [MCQ]

Which of the following describes a handle (as applicable to LR-parsing) appropriately?

A. It is the position in a sentential form where the next shift or reduce operation will occur
B. It is non-terminal whose production will be used for reduction in the next step
C. It is a production that may be used for reduction in a future step along with a position in the sentential form where the next shift or reduce operation will occur
D. It is the production p that will be used for reduction in the next step along with a position in the sentential form where the right hand side of the production may be found

Q10. [MCQ]

Which one of the following is a top-down parser?

A. Recursive descent parser.
B. Operator precedence parser.
C. An LR(k) parser.
D. An LALR(k) parser

Q11. [MCQ]

Consider the grammar with non-terminals

N = {S,C,S1 }, terminals T={a,b,i,t,e}, with S as the start symbol, and the following set of rules:

S --> iCtSS$_1$|a

S$_1$ --> eS|ϵ

C --> b

The grammar is NOT LL(1) because:

A. it is left recursive
B. it is right recursive
C. it is ambiguous
D. None

Q12. [MCQ]

Consider the context-free grammar G below

$S \rightarrow aSb \mid X$

$X \rightarrow aX \mid Xb \mid a \mid b$, where S and X are non-terminals, and a and b are terminal symbols. The starting non-terminal is S.

Which one of the following statements is CORRECT?

A. The language generated by G is $(a + b)^*$

B. The language generated by G is $a^*(a + b)b^*$

C. The language generated by G is $a^*b^*(a + b)$

D. The language generated by G is not a regular language

Q13. [MCQ]

Consider the following grammar.

S -> S * E

S -> E

E -> F + E

E -> F

F -> id

Consider the following LR(0) items corresponding to the grammar above.

(i) S -> S * .E

(ii) E -> F. + E

(iii) E -> F + .E

Given the items above, which two of them will appear in the same set in the canonical sets-of-items for the grammar of LR(0) parser?

A. (i) and (ii)
B. (ii) and (iii)
C. (i) and (iii)
D. None of the above

Q14. [MCQ]

A canonical set of items is given below

S --> L. > R

Q --> R.

On input symbol < the set has

A. **a shift-reduce conflict and a reduce-reduce conflict.**
B. **a shift-reduce conflict but not a reduce-reduce conflict.**
C. **a reduce-reduce conflict but not a shift-reduce conflict.**
D. **neither a shift-reduce nor a reduce-reduce conflict.**

Q15. [MCQ]

Consider the grammar defined by the following production rules, with two operators ∗ and +

S --> T * P

T --> U | T * U

P --> Q + P | Q

Q --> Id

U --> Id

Which one of the following is TRUE?

A. + is left associative, while $*$ is right associative
B. + is right associative, while $*$ is left associative
C. Both + and $*$ are right associative
D. Both + and $*$ are left associative

Q16. [MCQ]

Consider the following context-free grammar where the set of terminals is {a,b,c,d,f}.

$$\begin{aligned} \mathrm{S} &\rightarrow d\,a\,\mathrm{T} \mid \mathrm{R}\,f \\ \mathrm{T} &\rightarrow a\,\mathrm{S} \mid b\,a\,\mathrm{T} \mid \epsilon \\ \mathrm{R} &\rightarrow c\,a\,\mathrm{T}\,\mathrm{R} \mid \epsilon \end{aligned}$$

The following is a partially-filled LL(1) parsing table.

	a	b	c	d	f	$\$$
S			(1)	$\mathrm{S} \rightarrow d\,a\,\mathrm{T}$	(2)	
T	$\mathrm{T} \rightarrow a\,\mathrm{S}$	$\mathrm{T} \rightarrow b\,a\,\mathrm{T}$	(3)		$\mathrm{T} \rightarrow \epsilon$	(4)
R			$\mathrm{R} \rightarrow c\,a\,\mathrm{T}\,\mathrm{R}$		$\mathrm{R} \rightarrow \epsilon$	

Which one of the following choices represents the correct combination for the numbered cells in the parsing table ("blank" denotes that the corresponding cell is empty)

A. ① $S \rightarrow R\ f$ ② $S \rightarrow R\ f$ ③ $T \rightarrow \epsilon$ ④ $T \rightarrow \epsilon$

B. ① blank ② $S \rightarrow R\ f$ ③ $T \rightarrow \epsilon$ ④ $T \rightarrow \epsilon$

C. ① $S \rightarrow R\ f$ ② blank ③ blank ④ $T \rightarrow \epsilon$

D. ① blank ② $S \rightarrow R\ f$ ③ blank ④ blank

?

ANSWERS

A1. B

A2. D

A3. D

A4. C

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

1 2 3 4 5 6 7 8 9 10

A5. B

Given in the question, a grammar with no epsilon- and unit-production (i.e., of type A -> ϵ and A -> a).

To get the maximum number of Reduce moves, we should make sure that in each sentential form only one terminal is reduced. Since there is no unit production, the last 2 tokens will take only 1 move.

So To Reduce input string of n tokens, first Reduce n-2 tokens using n-2 reduce moves and then Reduce last 2 tokens using production which has . So a total of n-2+1 = n-1 Reduce moves.

Suppose the string is abcd. (n = 4).

We can write the grammar which accepts this string as follows:

S->aB

B->bC

C->cd

The Right Most Derivation for the above is:

S -> aB (Reduction 3)

-> abC (Reduction 2)

-> abcd (Reduction 1)

We can see here that no production is for unit or epsilon. Hence 3 reductions here. We can get less number of reductions with some other grammar which also doesn't produce unit or epsilon productions,

S->abA

A-> cd

The Right Most Derivation for the above as:

S -> abA (Reduction 2)

-> abcd (Reduction 1)

Hence 2 reductions.

But we are interested in knowing the maximum number of reductions which comes from the 1st grammar. Hence total 3 reductions is maximum, which is (n - 1) as n = 4 here.

Thus, Option B.

A6. A

FIRST(S) = { a, b, epsilon}

FIRST(A) = FIRST(S) = { a, b, epsilon}

FIRST(B) = FIRST(S) = { a, b, epsilon}

FOLLOW (A) = { b , a }

FOLLOW (S) = { $ } U FOLLOW (A) = { b , a , $ }

FOLLOW (B) = FOLLOW (S) = { b ,a , $ }

epsilon corresponds to an empty string.

A7. C

First(aSa) = a

First(bS) = b

First(c) = c

All are mutually disjoint i.e no common terminal between them, the given grammar is LL(1).

As the grammar is LL(1) so it will also be LR(1) as LR parsers are more powerful than LL(1) parsers and all LL(1) grammar are also LR(1)

So option C is correct.

A8. B

Regular expressions are used in lexical analysis.

Pushdown automata is related to context free grammar which is related to syntax analysis.

Dataflow analysis is done in code optimization.

Register allocation is done in code generation.

A9. D

A10. A

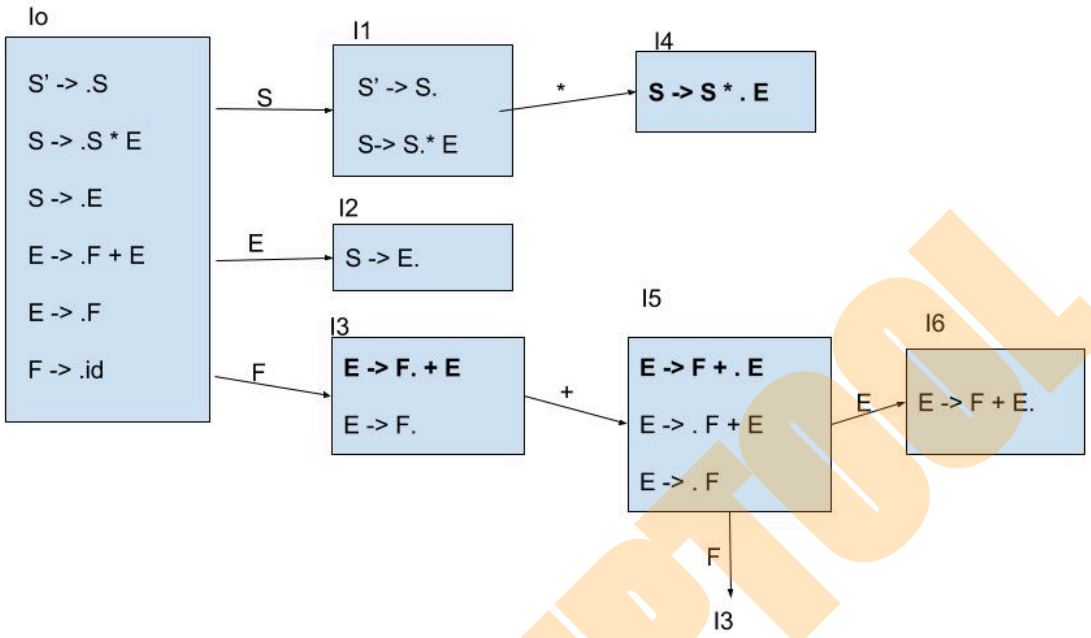
A11. C

for the string " ibtibtaea " it is ambiguous, because more than 1-parse tree possible

A12. B

A13. D

Below are the LR(0) sets of items.



In the LR(0) sets of items formed so far, we can see that the given three items belong to the different sets (I3, I4, I5).

Here, S' -> .S is an augmented production rule.

A14. D

The question is asked with respect to the symbol < which is not present in the given canonical set of items. Hence it is neither a shift-reduce conflict nor a reduce-reduce conflict on symbol <. Hence D is the correct option.

But if the question would have been asked with respect to the symbol > then it would have been a shift-reduce conflict.

A15. B

From the grammar we can find out associative by looking at grammar.

Let us consider the 2nd production

T -> T * U

T is generating T*U recursively (left recursive) so * is left associative.

Similarly

P -> Q + P

Right recursion so + is right associative.

So option B is correct.

A16. A

Lets Compute First and Follow of non terminals.

	First	Follw
S	d,c,f	c,f,$
T	a,b, ∈	c,f,$
R	c,∈	f

Now using first and follow we will construct an LL(1) parsing table.

1. S ->Rf production entry will come under the first of S terminals (c,f) in the parsing table.

2. T->∈ production entry will come under follow of T terminals (c,f,$) in the parsing table.