COMPILER DESIGN

SOLUTIONS

S → aSAb | bSBc	⇨First(S) = {a, b}
A → +AB | ε	⇨First(A) = {+, ε}
B → *BC | ε	⇨First(B) = {*, ε}
C → aC | d	⇨First(C) = {a, d}

1. What is in the Follow(S)?

(a) {a, b, c, +, $}
(b) {a, c, +, *, $}
(c) {b, c, +, *, $}
(d) {a, b, d, *, $}

Solution: Option (c)

Explanation:

S → aSAb | bSBc
A → +AB | ε
B → *BC | ε
C → aC | d

Follow(S) = {First(A), First(B), b, c, $}

2. What is in the Follow(B)?

(a) {a, b, c, d, *}
(b) {a, b, d, ε, $}
(c) {a, c, d, *, $}
(d) {c, d, b, +, *}

Solution: Option (a)

Explanation:

Follow(B) = {C, Follow(A), First(C)}
⬇
b, First(B)

= {c, b, *, a, d}

3. Choose the False statement.

(a) No left recursive/ ambiguous grammar can be LL(1)
(b) The class of grammars that can be parsed using LR methods is proper subset of the class of grammar that can be parsed by LL method
(c) LR parsing is non-backtracking method
(d) LR parsing can describe more languages than LL parsing

Solution: Option (b)

Explanation:

FALSE, as LL(1) ⊆ LR(k)

4. Consider the following SDT.

A → BC *(I) B.i = f(A.i)
(II) B.i = f(A.S)
(III) A.S = f(B.s)

Which of the above is violating L – attributed definition?

(a) I only
(b) II only
(c) I, II
(d) I, II, III

Solution: Option (b)

Explanation:

It does not follow L-attribute definition.

5.

X → YZ

Y → Y + Z {print ('+');}
T {Y.val = T.val}

Z → *Y {print ('*');} Z
T {Z.val = T.val}
ε

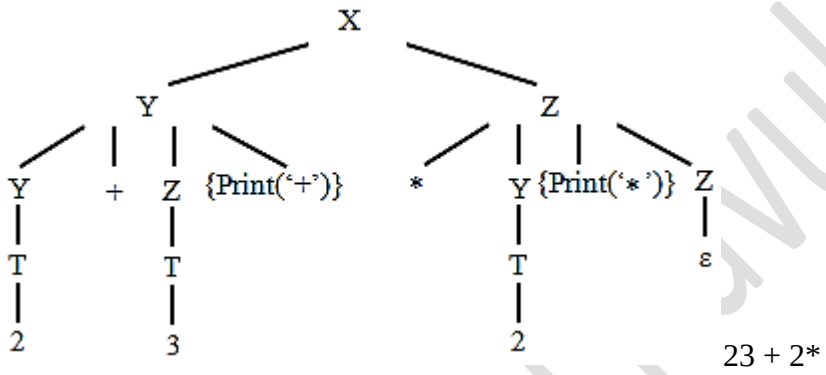
T → num {print(num.val);}

For 2+3*2, the above translation scheme prints

(a) 2+3*2 (b) 23+2*
(c) 232*+ (d) 23*2+

Solution: Option (b)

Explanation:



23 + 2*

6. Consider the following expression

x = a*b – c*d+e

For generating target code how many register will be required apart from accumulator A?

(a) 1 (b) 2
(c) 3 (d) 5

Solution: Option (a)

Explanation:

x = a * b – c * d + e

MOV A, a
MUL b
MOV R_1, A
MOV A, C
MUL d
SUB R_1, A
ADD R_1, e

So, One Register required.

7. Consider the following two grammars

G_1: A → A1 | 0A1 | 01
G_2: A → 0A | 1

Which of the following is True regarding above grammars?

(a) L_1 is LR(k)
(b) L_2 is LR(k)
(c) Both L_1 and L_2 is LR(k)
(d) None is LR(k)

Solution: Option (b)

Explanation:

G_1: A → A1 | 0A1 | 01 --- Ambiguous grammar
G_2: A → 0A | 1 --- Regular grammar

Ambiguous grammar is not LR(k)
Above Regular grammar is LR(k)

8. Consider the following grammar.

S → aB | aAb
A → bAb | a
B → aB | ε

How many back tracks are required to generate the string aab from the above grammar?

(a) 1
(b) 2
(c) 3
(d) 4

Solution: Option (b)

Explanation:

S ⇨ aB	S ⇨ aAb	S ⇨ aAb
⇨ aaB	⇨ abAbb	⇨ aab
⇨ aa	⇨ ababb	
Backtrack	*Backtrack*	

So, 2 backtracking is required.