*1. Write regular expressions for language that accept all strings of letters in which the letters are in ascending lexicographic order.*

**Solution:**

(A|a)\*(B|b)\*(C|c)\* (D|d)\*....... (Y|y)\* (Z|z)\*

2. *Consider the grammar:*

*S -> (L) | a*

*L -> L,S | S*

*Find parse trees for the following sentences:*

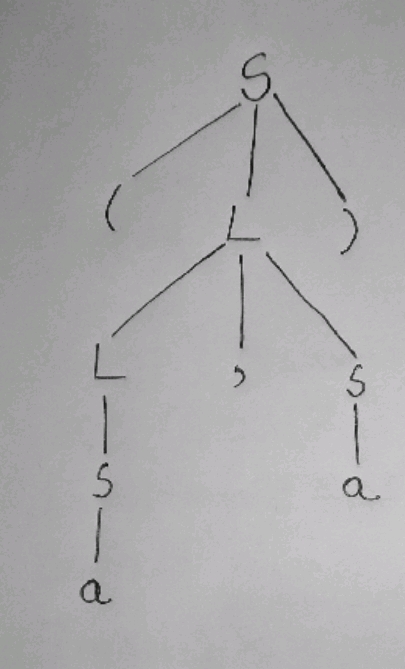
*i) (a, a),*

*ii) (a,(a,a)),*

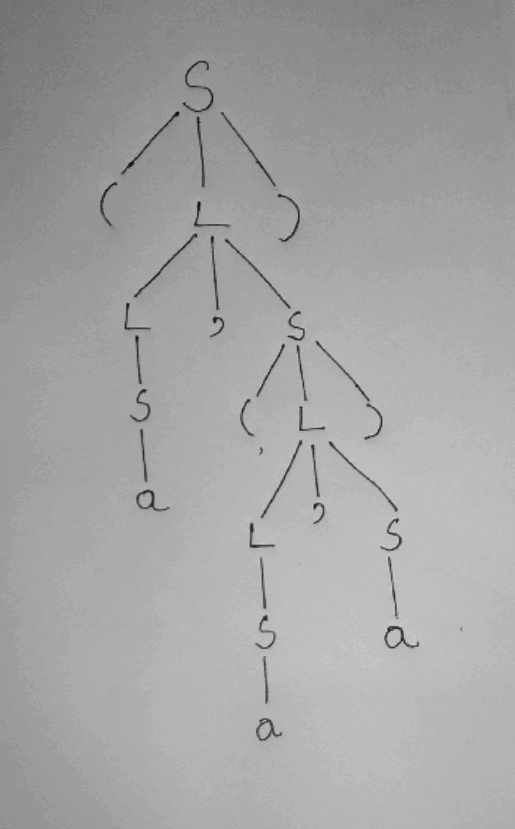
*iii) (a,((a,a),(a,a)))*

**Solution:**

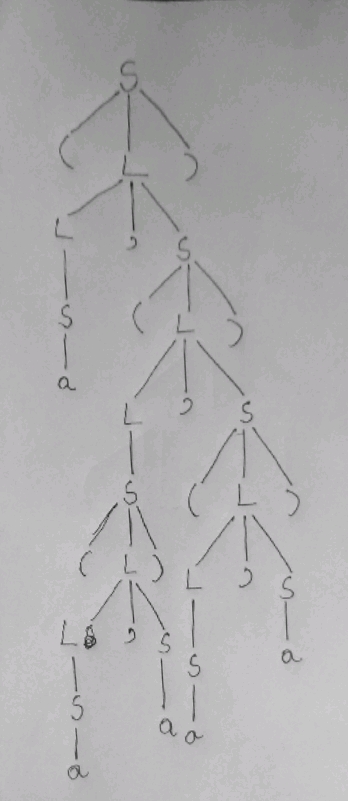
i)



ii)



iii)



*3. What language is generated by the following grammar:*

*S-> +SS | -SS | a*

**Solution:**

Prefix expression consisting of plus and minus signs

*4. Show that the grammar S-> aSbS | bSaS|*

*i) Is ambiguous by constructing two different leftmost derivations for the string abab.*

*ii) Construct the corresponding rightmost derivation for abab.*

*iii) Construct the parse tree for abab.*

**Solution:**

i) The two different leftmost deviation is constructed below with deviation tree for each of them:

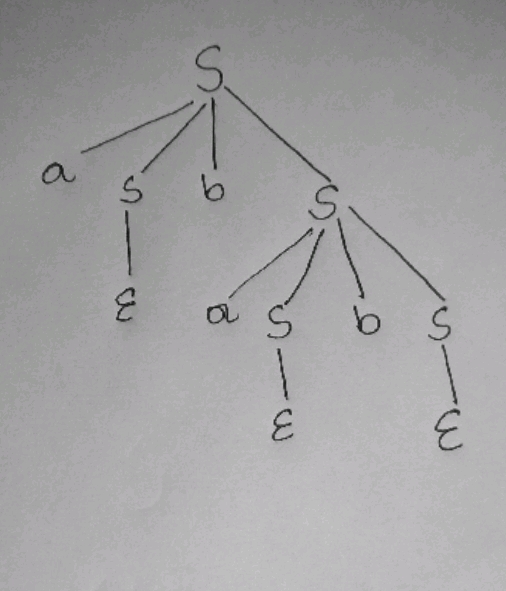
[1] S-> aSbS

-> abS (by S −> ε)

-> abaSbS (by S −> aSbS)

-> ababS (by S −> ε)

-> abab (by S −> ε)



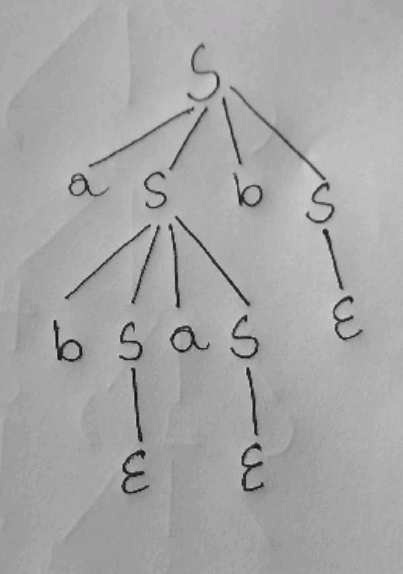
[2] S-> aSbS

-> abSaSbS (by S −> bSaS)

-> abaSbS (by S −> ε)

-> ababS (by S −> ε)

-> abab (by S −> ε)



As, the string **abab** constructs two different trees, the grammar S

−>aSbS |bSaS |ε is ambiguous.

ii) For rightmost derivation, derivation should be started from right side. For the string **abab**, this can be done in two ways –

a) S −> aSbS

−> aSbaSbS (by S −>aSbS)

−>aSbaSb (by S −>ε)

−>aSbab (by S −>ε)

−>abab (by S −>ε)

b) S −> aSbS

−> aSb (by S −> ε)

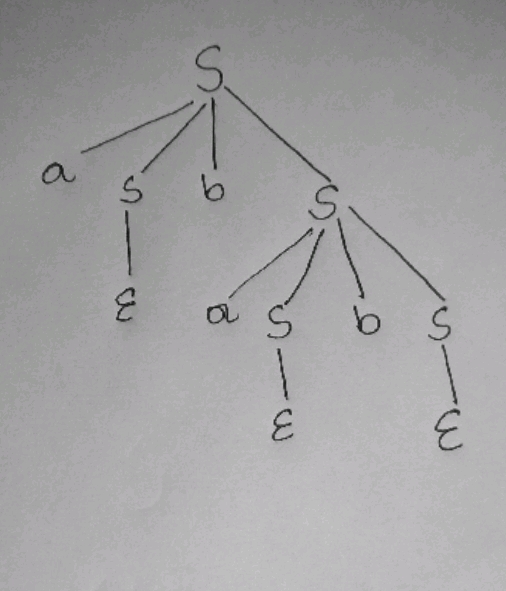
−>abSaSb (by S −> bSaS)

−>aSab (by S −>ε)

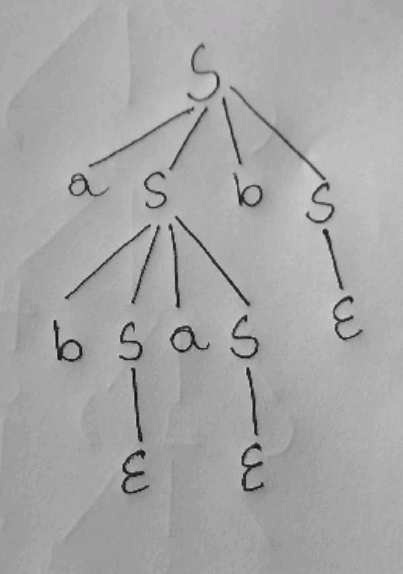
−>abab (by S −>ε)

iii) Parse tree for the string **abab** –

a)



b)



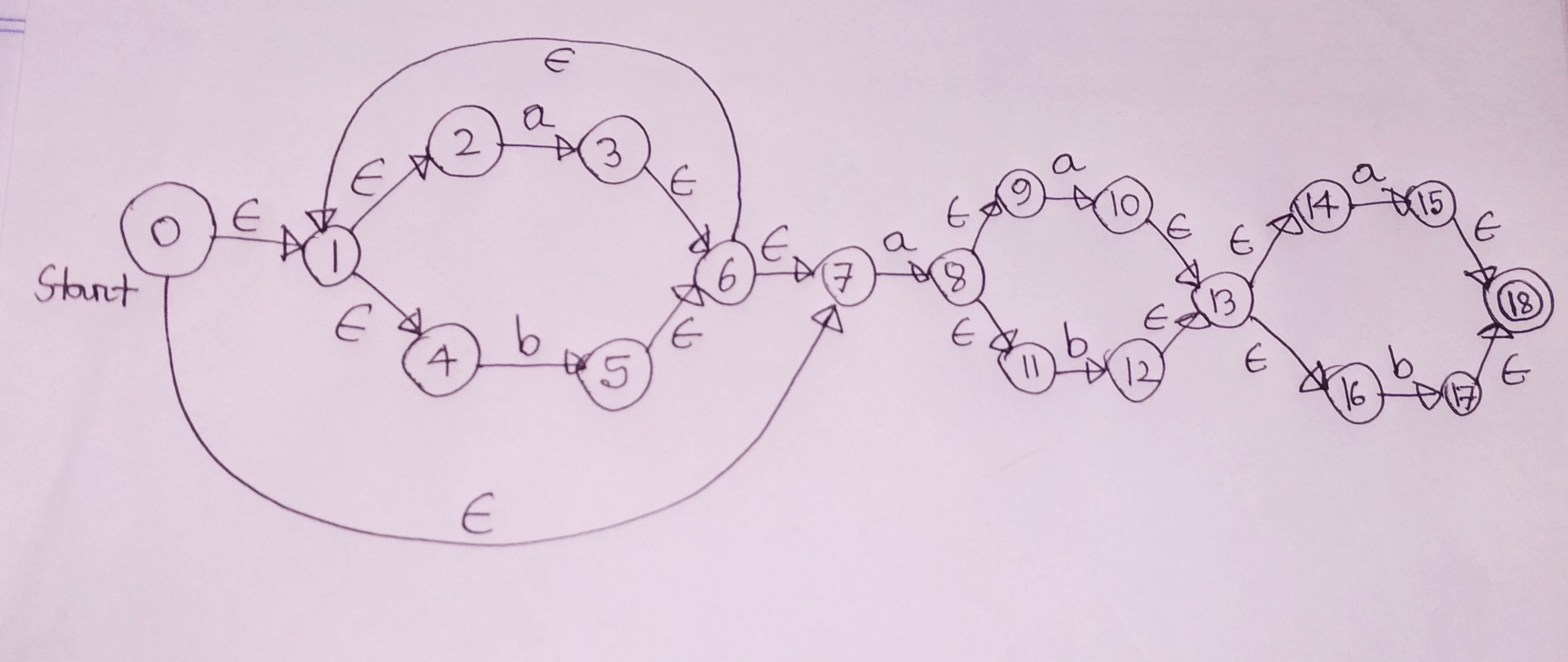
*5. Construct an NFA using Thompson’s construction from the following regular expression:*

*(a|b)\*a(a|b)(a|b)*

*Convert the resulting NFA to DFA using Subset construction rule.*

**Solution:**

**NFA:**



**DFA:**

**∈**-closure {0} = {0,1,2,4,7} = A

Move (A, a) = {3,8}

**∈** -closure {3,8} = {1,2,3,4,6,7,8,9,11} = B

Move (A, b) = {5}

**∈** -closure {5} = {1,2,4,5,6,7} = C

Move (B, a) = {3,8,10}

**∈** -closure {3,8,10} = {1,2,3,4,6,7,8,9,10,11,13,14,16} = D

Move (B, b) = {5,12}

**∈** -closure {5,12} = {1,2,4,5,6,7,12,13,14,16} =E

Move (C, a) = {3,8}

**∈** -closure {3,8} = B

Move (C, b) = {5}

**∈** -closure {5} = C

Move (D, a) = {3,8,10,15}

**∈** -closure {3,8,10,15} = {1,2,3,4,6,7,8,9,10,11,13,14,15,16,18} = F

Move (D, b) = {5,12,17}

**∈** -closure {5,12,17} = {1,2,4, 5,6,7,12,13,14,16,17,18} = G

Move (E, a) = {3,8,15}

**∈** -closure {3,8,15} = {1,2,3,4,6,7,8,9,11,15,18} = H

Move (E, b) = {5,17}

**∈** -closure {5,17} = {1,2,4,5,6,7,17,18} = I

Move (F, a) = {3,8,10,15}

**∈** -closure {3,8,10,15} = F

Move (F, b) = {5,12,17}

**∈** -closure {5,12,17} = G

Move (G, a) = {3,8,15}

**∈** -closure {3,8,15} = H

Move (G, b) = {5,17}

**∈** -closure {5,17} = I

Move (H, a) = {3,8,10}

**∈** -closure {3,8,10} = D

Move (H, b) = {5,12}

**∈** -closure {5,12} = E

Move (I, a) = {3,8}

**∈** -closure {3,8} = B

Move (I, b) = {5}

**∈** -closure {5} = C

Transition Table

|  |  |  |
| --- | --- | --- |
| **State** | **a** | **b** |
| A | B | C |
| B | D | E |
| C | B | C |
| D | F | G |
| E | H | I |
| F | F | G |
| G | H | I |
| H | D | E |
| I | B | C |

