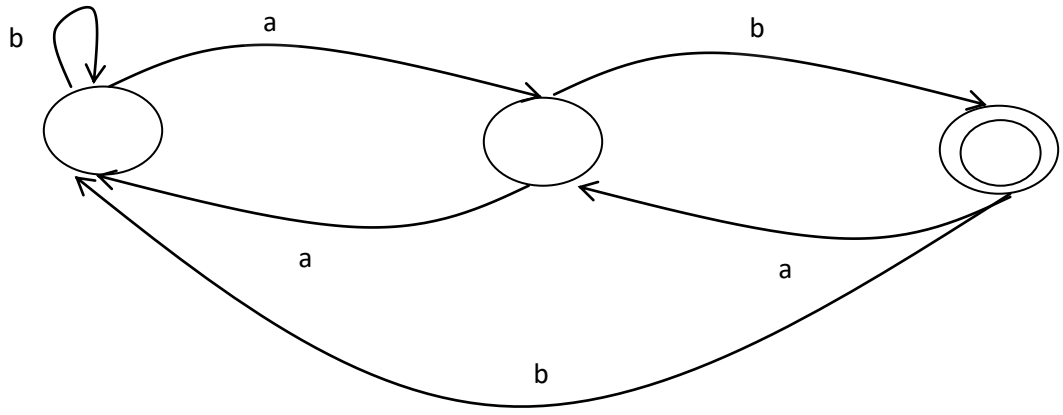
1) The regular language equivalent to
(a+b+c)*a(a+b+c)*

1. (c+b+a)*(c+b+a)*
2. (a+b+c)*(ab+bc+a)(a+b+c)*
3. (c+b+a)*a(c+b+a)*
4. (a+b+c)*(a+b+c)(a+b+c)*

2)



The regular language represented by this FA is:

1. (a+b)*
2. (a*+ab)(a+b)*ab
3. a*+(ab)*+(aab)*+(aaa*b)*
4. (a+b)*ab

3) Let S_i be the string consisting of i 0's followed by i 1's. Define the language $L_n=\{S_i \mid 1\le i\le n\}$. For example, $L_3=\{01,0011,000111\}$. Can you

represent L_n as a FA? If so, what is the smallest number of states needed for a DFA that recognizes L_n?

4) Given the following lexical specification

```
a(ba)*
b*(ab)*
abd
d+
```

i) dddabbabab is tokenized as ddd / a / bbabab

ii) ababddababa is tokenized as ab / abd / d / ababa

a) i and ii correct b) only i correct c) only ii correct d) neither i nor ii correct

5) a) Draw an NFA for the regular expression ((x+y)* a) + (x* a*)

b) Give the equivalent DFA for the NFA: draw state transition table

c) Reduce the DFA from step2 if possible.

6) Consider the Grammar:

```
1.S-> A$
2.A-> xBC
3.A->CB
4.B->yB
5.B->λ
```

a) What are the terminals and non-terminals of this language?
b) Describe the strings generated by this language with the help of a regular expression
c) What sequence of productions are applied to derive the string xyyx$? Draw the parse tree.
d) Compute the first and follow sets for all non-terminals.
e) Compute the predict set for each productions
f) Is this grammar LL(1)? If not, why not?

7) a) Draw an AST for the assignment statement `w := x + y * (z + 3)`

b) Give one advantage to generating ASTs before producing code, rather than producing code directly.

c) Give one possible three address code that would be generated for the above tree. Use the following

instructions: LD A, T loads from variable A into temporary T. OP T1, T2, T3 performs T3 = T1 OP T2. ST T, A stores from variable A into temporary T. OP are ADD, MUL.