1)

a) [A-Z]+ { return new Token(Token.NOMEPROP, yytext()); }

ou

[a-z][a-z]* { return new Token(Token.NOMEPROP, yytext()); }

[a-z]+ { return new Token(Token.TERM, yytext()); }

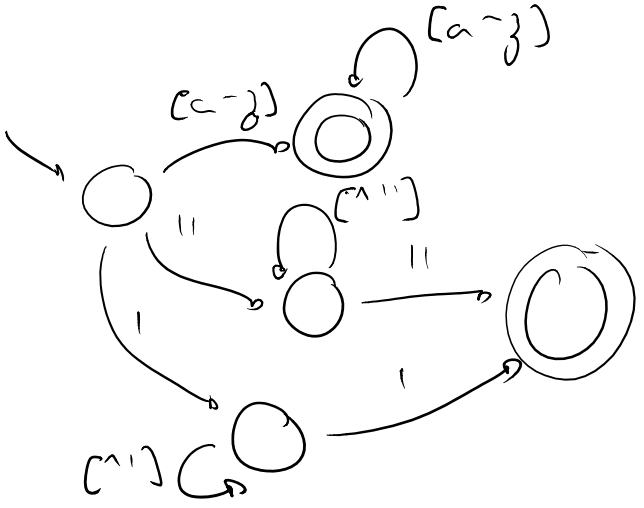
| ['][^']*['] <repete>
| ['][^']*['] <repete>

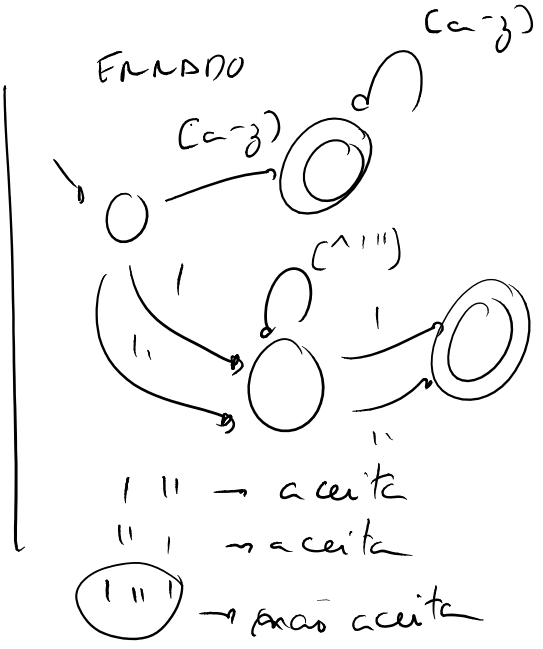
ou

[a-z]+ | ['][^']*['] | ['][^']['] { return new... }

['] ~ [']
[ ] ~ [ ]

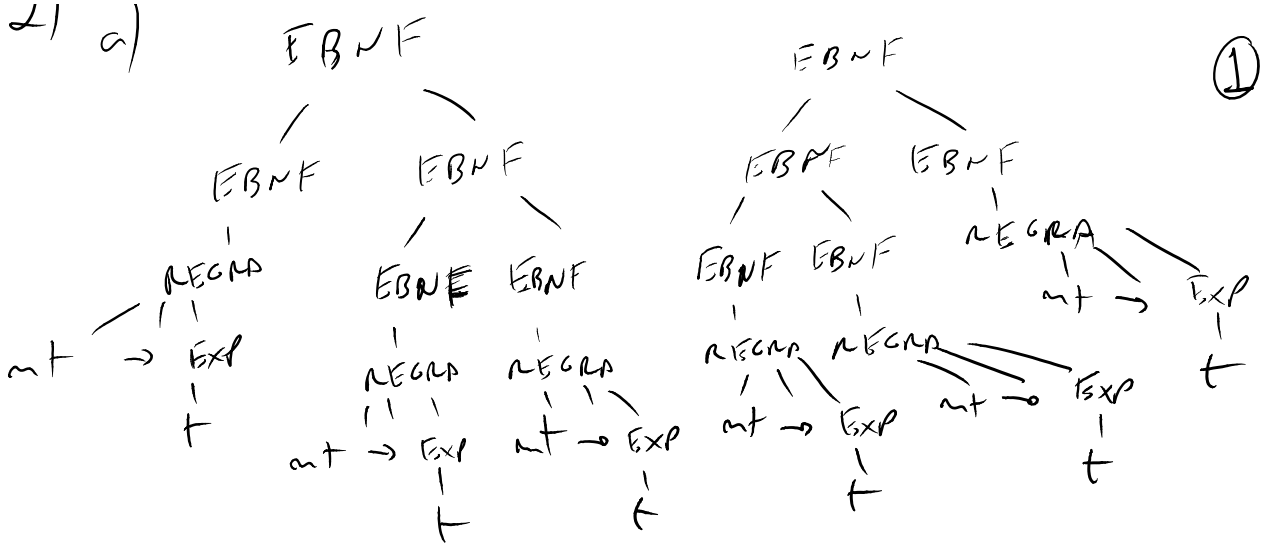
b)





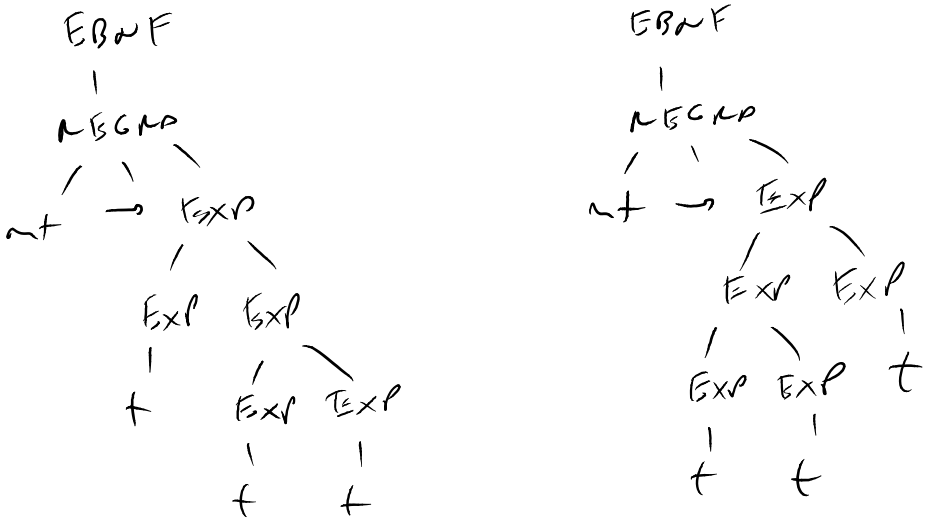
2) a) EBNF EBNF ①

21 a)



①

②



e outros!

b) A ⇒
A → ... | ○ vazio } regras com produção vazia

REGRA → não term "→" EXP
| não term "→"
| não term "→" EXP "|"

ou

REGRA → não term "→" [EXP ["|"]]

REGRA → não term "→" (EXP) ("|") X

ou

parcial EXP → EXP EXP | EXP "|" EXP | naoterm | term |

c) EXP → EXP EXP
| EXP "|" EXP
| naoterm
| term
| "(" EXP ")"
| "C" EXP "J"
| "{" EXP "}"

d) EXP → EXP "|" JEXP
| JEXP

assoc. esquerda

JEXP → AEXP JEXP
| AEXP

assoc. direita

AEXP → naoterm
| term

3) a) Produções 3 e 4 de CMD começam com id.
ou Produções 3 e 4 de CMD têm conjuntos de lookahead {"id"}, logo a interseção dos conjuntos não é vazia
ou Produções 3 e 4 de CMD têm id como prefixo comum

CMD → read id
| write id
| id CMD'

ou

CMD → read id
| write id
| id / id ":=" EXP .

| num | id

| id CMD'

CMD' → ":=" EXP
| "(" ")"

ou

| num | id

| id (":=" EXP
| "(" ")")

b) Tree EXP() {

Tree res = new Tree("EXP");
switch (la) {
case "num":
res.filho(match("num"));
break;
case "id":
res.filho(match("id"));
break;
case "(":
res.filho(match("("));
res.filho(EXP());
res.filho(match("+"));
res.filho(EXP());
res.filho(match(")"));
break;
default:
erro("expressão inválida");
}

ordem não importa

ou

default: → retira

}

return res;

}