Monday, October 21, 2013 2:43 PM

1)

a)

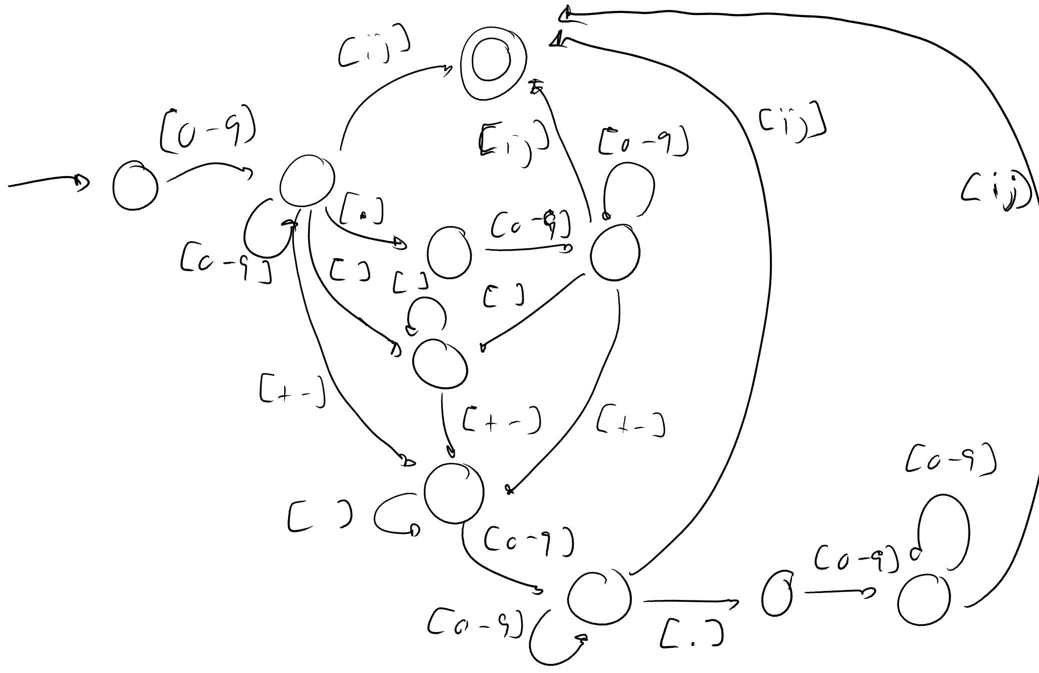
```
[0-9]+[ij]
[0-9]+[.][0-9]+[ij]
[0-9]+[ ]*[+\-][ ]*[0-9]+[ij]
[0-9]+[ ]*[+\-][ ]*[0-9]+[.][0-9]+[ij]
[0-9]+[.][0-9]+[ ]*[+\-][ ]*[0-9]+[ij]
[0-9]+[.][0-9]+[ ]*[+\-][ ]*[0-9]+[.][0-9]+[ij]
```

(i|j) ≡ [ij]

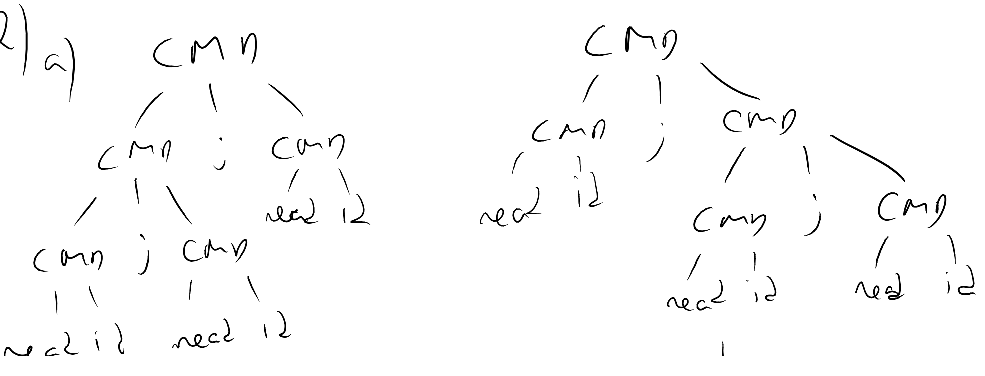
ou

```
([0-9]+([.][0-9]+)?[ ]*[+\-][ ]*)?[0-9]+([.][0-9]+)?[ij]
```

b)



2) a)



read id , read id ; read id   read id ; read id ; read id

b) CMD → CMD' ; CMD
        | CMD'

CMD' → if EXP then CMD end
        | if EXP then CMD else CMD end
        | repeat CMD until EXP
        | id := EXP
        | read id
        | write EXP

3) a) SEXP → LISTA → '(' SEXPS ')' → '(' SEXP SEXPS ')'
→ '(' ATOMO SEXPS ')' → '(' id SEXPS ')'
→ '(' id SEXP SEXPS ')' → '(' id LISTA SEXPS ')'
→ '(' id '(' SEXPS ')' SEXPS ')'
→ '(' id '(' SEXP SEXPS ')' SEXPS ')'
→ '(' id '(' ATOMO SEXPS ')' SEXPS ')'
→ '(' id '(' id SEXPS ')' SEXPS ')'
→ '(' id '(' id SEXP ')' SEXPS ')'
→ '(' id '(' id LISTA ')' SEXPS ')'
→ '(' id '(' id '(' SEXPS ')' ')' SEXPS ')'

```
→ '(' id '(' id '(' SEXP ')' ')' SEXPS ')'
→ '(' id '(' id '(' ATOMO ')' ')' SEXPS ')'
→ '(' id '(' id '(' num ')' ')' SEXPS ')'
→ '(' id '(' id '(' num ')' ')' SEXP ')'
→ '(' id '(' id '(' num ')' ')' LISTA ')'
→ '(' id '(' id '(' num ')' ')' '(' SEXPS ')' ')'
→ '(' id '(' id '(' num ')' ')' '(' SEXP ')' ')'
→ '(' id '(' id '(' num ')' ')' '(' ATOMO ')' ')'
→ '(' id '(' id '(' num ')' ')' '(' id ')' ')'
```

b)
```
Arvore lista() {
    Arvore res = new Arvore("LISTA");
    res.filho(match('('));
    res.filho(sexps());
    res.filho(match(')'));
    return res;
}

Arvore atomo() {
    Arvore res = new Arvore("ATOMO");
    if (la == NUM)
```

```
            res.filho(match(NUM));
        else
            res.filho(match(ID));
    } return res;

Arvore sexp() {
    Arvore res = new Arvore("SEXP");
    if(la == '(')
        res.filho(lista());
    else
        res.filho(atomo());
    return res;
}
```

c) SEXP é um prefixo em comum das regras de SEXPS.

$$\begin{array}{l} SEXPS \to SEXP\ SEXPS' \\ SEXPS' \to SEXPS \\ \qquad\qquad |\end{array} \xrightarrow{\text{simplificando}} \begin{array}{l} SEXPS \to SEXP\ SEXPS' \\ SEXPS' \to SEXP\ SEXPS' \\ \qquad\qquad |\end{array}$$

ou

$SEXPS \to SEXP \{ SEXP \}$

```
Arvore sexps() {
    Arvore res = new Arvore("SEXPS");
    res.filho(sexp());
    res.filho(sexps1());
    return res;
}
```

```
}

Arvore sexpsl() {
    Arvore res = new Arvore("SEXPSL");
    if(la == '(' || la == NUM || la == ID) {
        res.filho(sexp());
        res.filho(sexpsl());
    }
    return res;
}
```

OU

```
// SEXPS -> SEXP {SEXP}
Arvore sexps() {
    Arvore res = new Arvore("SEXPS");
    res.filho(sexp());
    while(la == '(' || la == NUM || la == ID) {
        res.filho(sexp());
    }
    return res;
}
```