2.1. $G_{Integer}$ for integer 4520

Integer => Integer Digit
→ Integer Digit Digit
→ Integer Digit Digit Digit
→ Digit Digit Digit Digit
→ 4 Digit Digit Digit
→ 45 Digit Digit
→ 452 Digit
→ 4520

for integer d_1

Integer → Digit
→ 1

d_2

Integer → Integer Digit
→ Digit Digit
→ 1 Digit
→ 12

Steps = $2d_n$

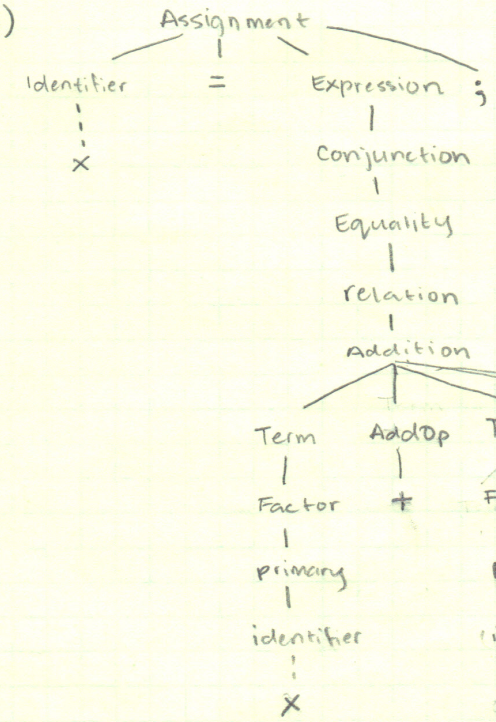
2.2 $G_{Integer}$ derive 4520 rightmost

Integer → Integer Digit
→ Integer 0
→ Integer Digit 0
→ Integer 2 0
→ Integer Digit 2 0
→ Integer 5 2 0
→ Digit 5 2 0
→ 4 5 2 0

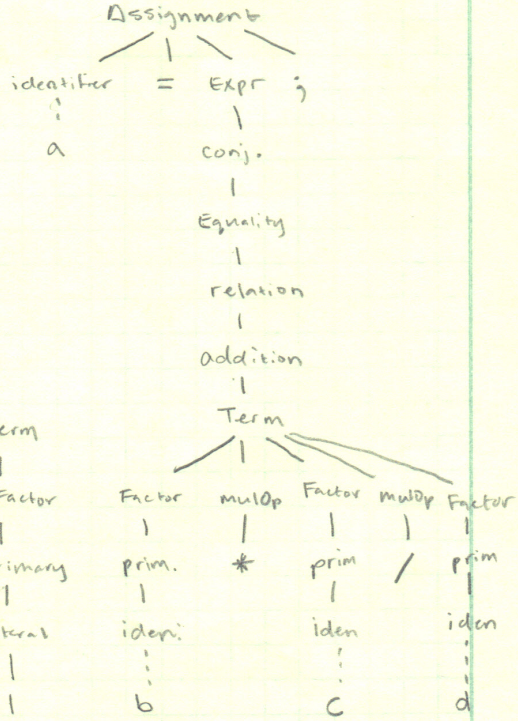
2.5 using $G_{f2.7}$ draw parse trees for

x = x + (a - 1); a = (b * (c / d)); i = i + j * k - 3

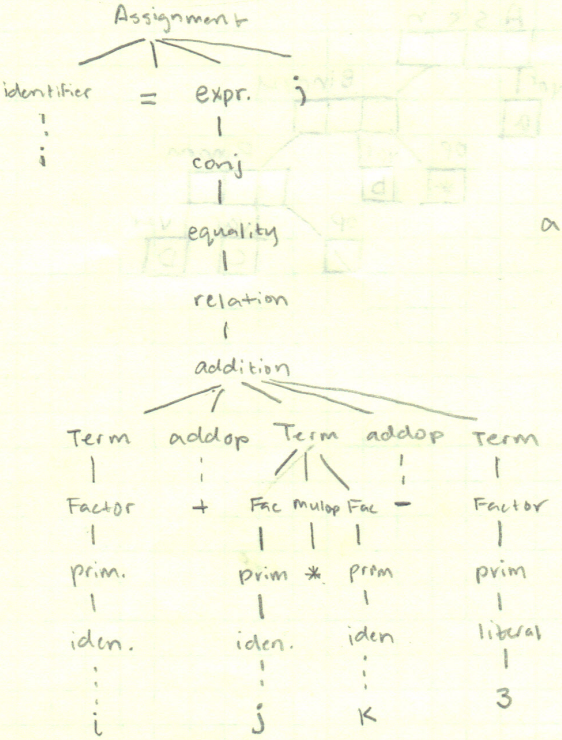
a)



b)



c) $i = i + (j * (k - 3))$

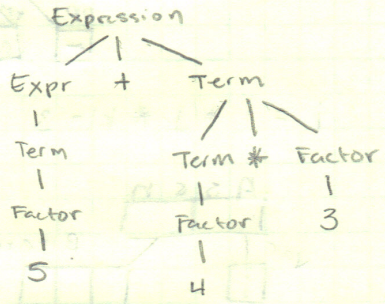


2.8

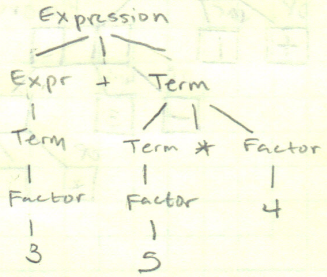
Expr → Expr + Term | Term
Term → Term * Factor | Factor
Factor → 0|...|9 | (Expr)

a) 5 + 4 * 3
b) 5 * 4 + 3

a)



b)



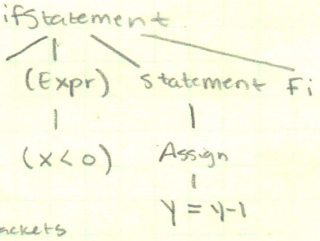
2.13 ① if (x<0) if (x==0) y = y-1; else y=0; fi

② if (x<0) if (x==0) y = y-1 ; fi else y=0; fi

①
```
if (x<0)
    if (x==0)
    y = y-1;
    else
    y = 0 fi
```

②
```
if (x<0)
    if (x==0)
    y = y-1; Fi
else
    y = 0; Fi
```

ifStatement → if (Expr) Statement [fi] |
if (Expr) Statement else State. [fi]



Adding fi to the grammar would ensure all if's were coupled with fi after a statement completed.

2.15 Give translation rules for EBNF metabrackets and Metaparenthesis.

BNF statement → Expr − Term | Expr + Term | Term
EBNF gives → Term { (+|−) Term }

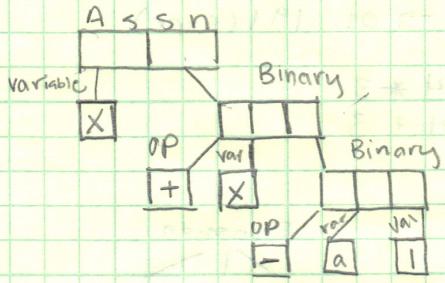
BNF → Addition | Addition Relop Addition
EBNF → Addition [Relop Addition]

when given a choice between two* terminals and some relop, binop, or *, sub (*|*) for redundant terminals terminals that are separated by (|) and have * inbetween can be omitted using [] metabrackets

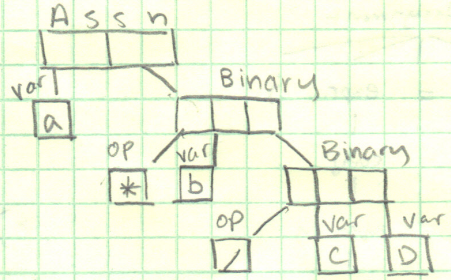
* anyop

2.20 back →

2.20 $x = x + a - 1$



$a = b * c / D$



$i = i + (j * k) - 3$