Opgaveløsninger
(sæt 7)

Opgave 23: 10.2 (1 point)

(a)

H_{2C} og H_{2D} er gendrivelser af C_2, eftersom DRAW allerede er opnået ved trækket C_1.

(b)

Værdien af stillingen er DRAW.

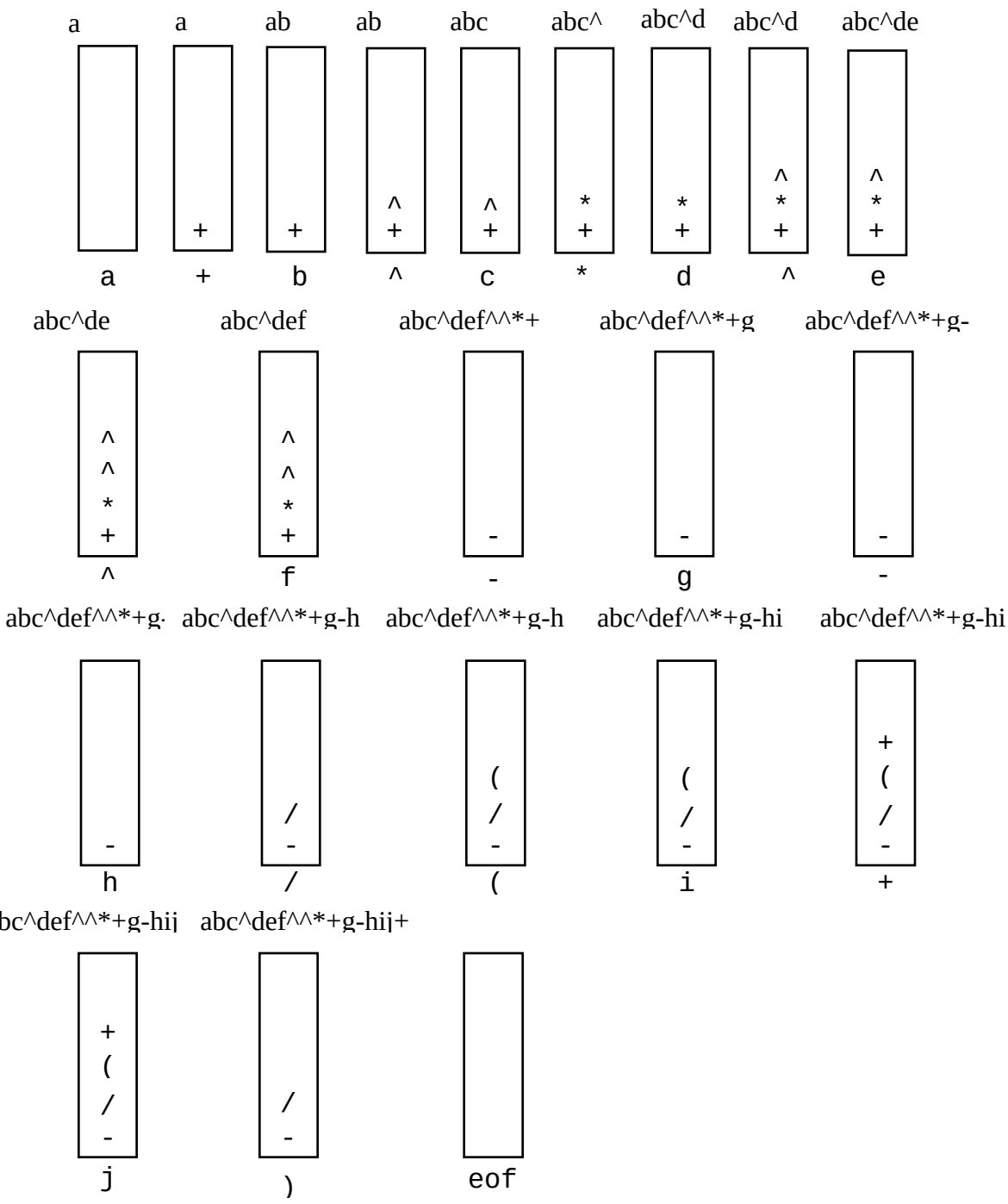
Opgave 24: 11.2 (1 point)

a. `1 2 + 3 4 ^ -`
b. `1 2 ^ 3 4 * -`
c. `1 2 3 * + 4 5 ^ - 6 +`
d. `1 2 + 3 * 4 5 6 - ^ -`

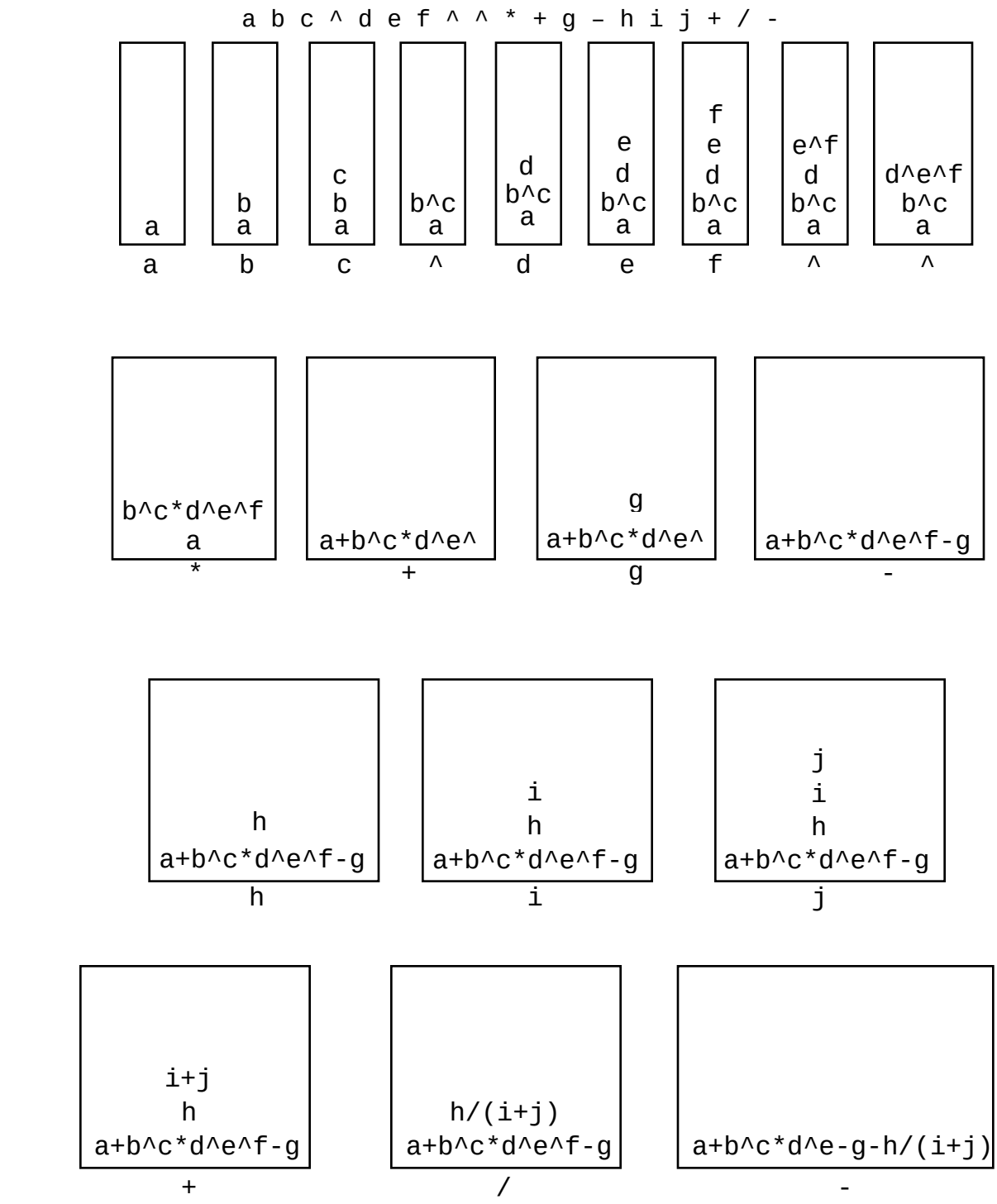
Opgave 25: 11.3 (2 point)

a.

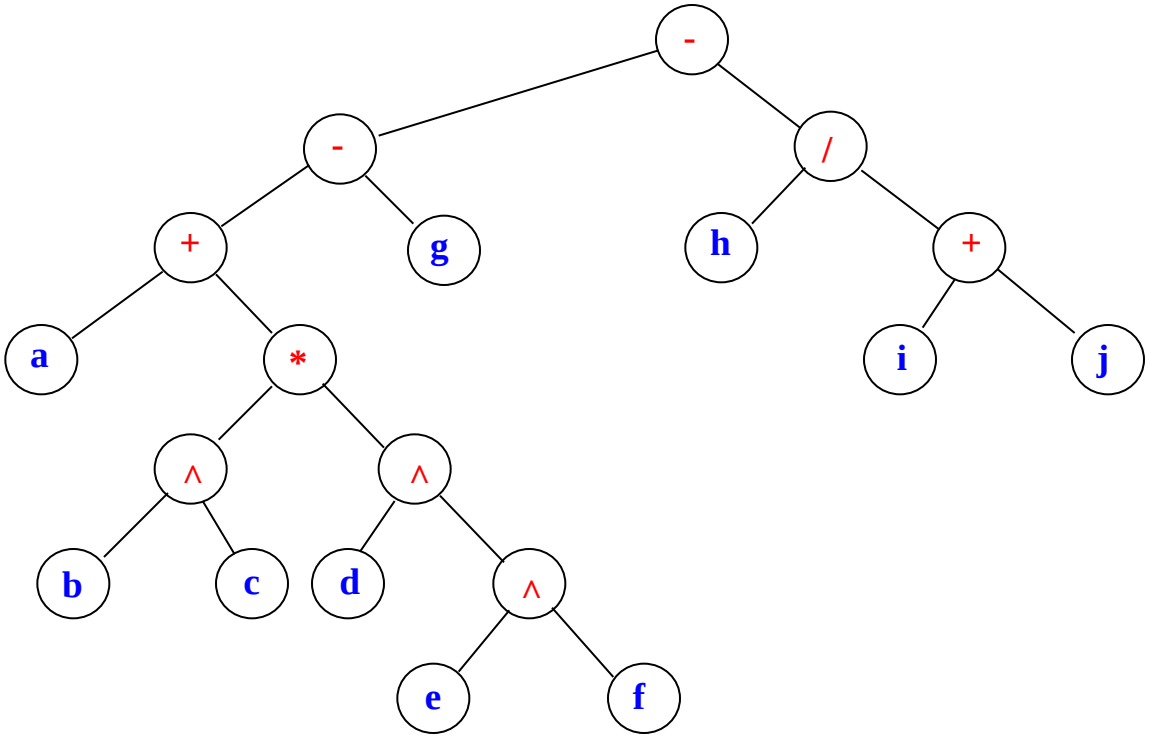
a + b ^ c * d ^ e ^ f – g – h / (i + j)



b.


a b c ^ d e f ^ ^ * + g – h i j + / -
a
b
a
c
b
a
b^c
a
d
b^c
a
e
d
b^c
a
f
e
d
b^c
a
e^f
d
b^c
a
d^e^f
b^c
a
a
b
c
^
d
e
f
^
^
b^c*d^e^f
a
*
a+b^c*d^e^
+
g
a+b^c*d^e^
g
a+b^c*d^e^f-g
-
h
a+b^c*d^e^f-g
h
i
h
a+b^c*d^e^f-g
i
j
i
h
a+b^c*d^e^f-g
j
i+j
h
a+b^c*d^e^f-g
+
h/(i+j)
a+b^c*d^e^f-g
/
a+b^c*d^e-g-h/(i+j)
-

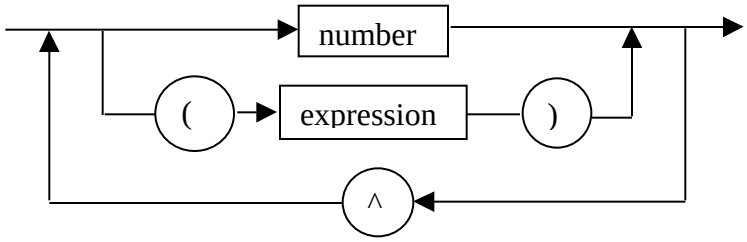
c.



Opgave 26: Calculator (ikke-obligatorisk, 3 point)

Syntaksdiagrammet for "factor" ændres til

factor



Ændringerne i klassen Calculator er angivet nedenfor med **fed** skrift.

```
import java.util.*;

public class Calculator {
    static final int PLUS = 0, MINUS = 1, MULT = 2,
                     DIV = 3, LPAR = 4, RPAR = 5,
                     CONST = 6, EOS = 7, POWER = 8;
    int token;
    double value;
    StringTokenizer str;

    double valueOf(String s) {
        str = new StringTokenizer(s, "+-*/()= ^", true);
        getToken();
        return expression();
    }

    double expression() {
        double v = term();
        while (token == PLUS || token == MINUS)
            if (token == PLUS)
              { getToken(); v += term(); }
            else
              { getToken(); v -= term(); }
        return v;
    }

    double term() {
        double v = factor();
        while (token == MULT || token == DIV)
            if (token == MULT)
              { getToken(); v *= factor(); }
            else
              { getToken(); v /= factor(); }
        return v;
    }
```

```
double factor() {
    double v = 0;
    if (token == CONST)
        v = value;
    else if (token == LPAR) {
        getToken();
        v = expression();
        if (token != RPAR)
            error("missing right paranthesis");
    } else
        error("illegal factor");
    getToken();
    if (token == POWER) {
        getToken();
        v = Math.pow(v, factor());
    }
    return v;
}
```

```
    void error(String msg) {
        throw new RuntimeException(msg);
    }

    void getToken() {
        String s;
        try {
            s = str.nextToken();
        } catch(NoSuchElementException e) {
            token = EOS;
            return;
        }
        if (s.equals(" ")) getToken();
        else if (s.equals("+")) token = PLUS;
        else if (s.equals("-")) token = MINUS;
        else if (s.equals("*")) token = MULT;
        else if (s.equals("/")) token = DIV;
        else if (s.equals("(")) token = LPAR;
        else if (s.equals(")")) token = RPAR;
        else if (s.equals("^")) token = POWER;
        else {
            try{
                value = Double.valueOf(s).doubleValue();
                token = CONST;
            } catch(NumberFormatException e) {
                error("constant expected");
            }
        }
    }

    public static void main(String arg[]) {
        Calculator calc = new Calculator();
        System.out.println(calc.valueOf("3*2+4*5"));
        System.out.println(calc.valueOf("3*2^2^3/2"));
    }
}
```