Consider the following regular expression: \((a+b)^*(c+d)^*\)

1. Transform this regular expression to an NFA, from there to a right-linear regular grammar, and from there back to the original regular expression.
2. Transform the NFA from part 1, to a DFA.
3. Minimize the DFA obtained in Part 2.
4. Write (in pseudo-code) a lexical analyzer for the language given by the above regular expression. Write three versions of the lexical analyzer:
   - Table-driven
   - Hard-coded
4. Table-driven (non-minimal DFA)  Token_scan() {
    Token t = empty();
    S = 1;
    while c ≠ null {
        if ST[S,c] = "-"
            { Error(c); c = getchar(); continue; }
        else
            { S = ST[S,c]; t = t + c; c = getchar(); }
    }
    return t;
}

Hand-coded:

Token_scan() {
    S = 1;
    while c ≠ null {
        case S of
            1: case c of
                a, b: S = 2;
                c, d: S = 3;
            end;
            2: case c of
                c, d: S = 3;
            end;
            3: case c of
                a, b: Error(c);
            end;
            end;
            t = t + c; c = getchar();
        }
    return t;
}