**Musterlösung Übungsblatt 4**

1. a)
FIRST(0S1) = {0}
FIRST(01) = {0}

D.h. Lookahead von nur einem Symbol reicht nicht aus. Umformung:

S → 0A
A → S1 | 1

FIRST(0A) = {0}
FIRST(S1) = {0}
FIRST(1) = {1}

```
procedure S;
begin
        match('0');
        A();
end;

procedure A;
begin
        if LA='0' then
        begin
                S();
                match('1');
        end
        else if LA='1' then match('1');
        else error();
end;
```

Es kann sein, dass Prozedur S terminiert aber nicht alle Tokens der Eingabe geparst wurden. In diesem Fall war die Syntax der Eingabe falsch.

b)
FIRST(+SS) = {+}
FIRST(-SS) = {-}
FIRST(a) = {a}

```
procedure S;
begin
        if LA='+' then
        begin
                match('+');
                S();
                S();
        end
        else if LA='-' then
        begin
                match('-');
                S();
                S();
        end
        else if LA='a' then
                match('a');
        else error();
end;
```

c)
Problem: Linksrekursion!!! Umformung nach Schema:

A → Aα | β | γ
…wird umgeformt zu…
A → βR | γR
R → αR | ε

S → R
R → (S) S R | ε

Vereinfachung durch Substitution von S durch R:

R → (R) R R | ε

FIRST((R) R R) = {(}

```
procedure R;
begin
        if LA='(' then
        begin
                match('(');
                R();
                match(')');
                R();
                R();
        end;
end;

procedure parse;
begin
        Eingabe :=
          Tokenliste des Ausdrucks
        S();
        if(not Eingabe leer)
                error();
end;
```

2.
Wir entwickeln einen Parser für die Grammatik korrekter Klammerausdrücke:

S → (S) S | [S] S | ε

FIRST((S) S) = {(}
FIRST([S] S) = {[}

```
procedure S;
begin
        if LA='(' then
        begin
                match('(');
                S();
                match(')');
                S();
        end
        else if LA='[' then
        begin
                match('[');
                S();
                match(']');
                S();
        end
end;

procedure prüfeKlammern;
begin
        Eingabe :=
          Tokenliste des Ausdrucks
        S();
        if(Eingabe leer)
                print("Ausdruck korrekt“);
        else
                print("Ausdruck falsch“);
end;
```

3.
Plus → Mal + Plus | Mal
Mal → Num * Mal | Num
Num → 0 | … | 9

Umformung da Lookahead eins nicht ausreicht:

Plus → Mal A
A → + Plus | ε
Mal → Num B
B → * Mal | ε
Num → 0 | … | 9

FIRST(Mal A) = {0, …, 9}
FIRST(+ Plus) = {+}
FIRST(Num B) = {0, …, 9}
FIRST(* Mal) = {*}
FIRST(Num) = {0, …, 9}
FIRST(0) = {0}
…

```
procedure Plus;
begin
        Mal();
        A();
end;

procedure A;
begin
        if LA='+' then
        begin
                match('+');
                Plus();
                println("Apply +");
        end;
end;

procedure Mal;
begin
        Num();
        B();
end;

procedure B;
begin
        if LA='*' then
        begin
                match('*');
                Mal();
                println("Apply *");
        end;
end;

procedure Num;
begin
        if LA='0' then
        begin
                match('0');
                println("push 0");
        end
        else …
        else if LA='9' then
        begin
```

```
            match('9');
            println("push 9");

        end
end;
```

4.
Beweis durch strukturelle Induktion.
Beh.: num = k*3 mit k є |N

- $11_b$ = 3, d.h. Beh. gilt.
- $1001_b$ = 9, d.h. Beh. gilt.
- Beh.: num' = k'*3 mit k є |N
  num = $num'_b$ $0_b$ = 2*num'
  = (2*k')*3, d.h. Beh. gilt.
- Beh.: num' = k'*3 mit k є |N
  und num'' = k''*3 mit k є |N
  num = $num'_b$ $num''_b$
  = $2^{|num''b|}$ * k'*3 + k'' * 3
  = 3 * ($2^{|num''b|}$ * k' + k'')

q.e.d.