Zuname _____ Vorname _____ Matr. Nr. _____

Übungsgruppe
- ❑ 1 (Wöß) Do 10^{15}-11^{45}
- ❑ 2 (Wöß) Do 12^{45}-14^{15}
- ❑ 3 (Rammerstorfer) Do 14^{30}-16^{15}

Punkte _____ korr. _____

Letzter Abgabetermin
Donnerstag, 11.10.2001, 8^{15} Uhr

Grammatiken

1. Grundbegriffe (1+3+3+6 Punkte)

Die Grammatik der Sprache *MicroJava* sieht folgendermaßen aus:

```
Program    = "class" ident {ConstDecl | VarDecl | ClassDecl} "{" {MethodDecl}
"}".
ConstDecl  = "final" Type ident "=" (number | charConst) ";".
VarDecl    = Type ident ["[" "]"] {"," ident ["[" "]"]} ";".
ClassDecl  = "class" ident "{" {VarDecl} "}".
MethodDecl = (Type | "void") ident "(" [FormPars] ")"
             {VarDecl} "{" {Statement} "}".
FormPars   = Type ident ["[" "]"] {"," Type ident ["[" "]"]}.
Type       = ident.
Statement  = Designator ("=" Expr | "(" [ActPars] ")" | "++" | "--") ";"
           | "if" "(" Condition ")" Statement ["else" Statement]
           | "while" "(" Condition ")" Statement
           | "break" ";"
           | "return" [Expr] ";"
           | "read" "(" Designator ")" ";"
           | "print" "(" Expr ["," number] ")" ";"
           | "{" {Statement} "}".
ActPars    = Expr {"," Expr}.
Condition  = CondTerm {"||" CondTerm}.
CondTerm   = CondFact {"&&" CondFact}.
CondFact   = Expr Relop Expr.
Expr       = ["-"] Term {Addop Term}.
Term       = Factor {Mulop Factor}.
Factor     = Designator ["(" [ActPars] ")"]
           | number
           | charConst
           | "new" Type ["[" Expr "]"]
           | "(" Expr ")".
Designator = ident {"." ident | "[" Expr "]"}.
Relop      = "==" | "!=" | ">" | ">=" | "<" | "<=".
Addop      = "+" | "-".
Mulop      = "*" | "/" | "%".
```

a) Geben Sie den kürzesten Satz an, den man mit dieser Grammatik erzeugen kann.

b) Betrachten Sie die Nonterminalsymbole *Statement*, *ActPars* und *Expr*. Geben Sie für jedes dieser Nonterminalsymbole an, ob es links-, zentral- oder rechtsrekursiv und direkt oder indirekt rekursiv ist.

c) Zeichnen Sie den Syntaxbaum für folgenden Satz:

```
class Point int x; int y; { void moveTo(Point p) { x = p.x; y = p.y; } }
```

Gibt es mehrere Syntaxbäume für diesen Satz? (Mit Begründung)

d) Bestimmen Sie alle terminalen Anfänge und Nachfolger für die Regeln *ActPars*, *Expr*, *CondTerm* und *Statement.*

2. Konstruktion einer Grammatik (3 Punkte)

Geben Sie eine Grammatik (in EBNF) für die Bezeichner einer fiktiven Programmiersprache gemäß den folgenden Bedingungen an:

Die Bezeichner dürfen aus Buchstaben (Terminalklasse *b*), Ziffern (Terminalklasse *z*) und dem Unterstreichungszeichen (*"_"*) bestehen.

Ein Bezeichner muß mit einem Buchstaben beginnen und darf anschließend beliebig viele Buchstaben und Unterstreichungszeichen in beliebiger Kombination enthalten. Auch Ziffernfolgen sind erlaubt, allerdings müssen sie von Unterstreichungszeichen umgeben sein.

Bsp.

gültige Bezeichner:	ungültige Bezeichner
b	*bz*
*b_*	*b_zzzz*
b_z_b	*_zzz_*
bb__b_zzz_z_b	*bz_b*

3. Beseitigung von Linksrekursionen (4 Punkte)

Gegeben sei folgender Auszug aus einer fiktiven Grammatik, der Methodenaufrufe beschreibt:

```
MethodCall =  MethodCall "." Ident "(" [ Parameter ] ")"
            | Ident "." Ident "(" [ Parameter ] ")" .
Parameter  =  MethodCall | Ident .
Ident      =  Ident letter | Ident digit | letter .
```

Beseitigen Sie alle Linkrekursivitäten und geben Sie die transformierte Grammatik in EBNF an. *letter* und *digit* sind Terminalklassen und definieren die Mengen der Buchstaben und der Ziffern.