5.1 Typregeln I

Gegeben seien die Klassen aus der Vorlesung:

```
class Cl1 {
   Cl2 v;
}

class Cl2 { ...}
```

Leiten Sie mit den Typregeln aus der Vorlesung für den Block

```
{
    Cl1 x;
    return x.v;
}
```

den korrekten Typ her.

5.2 Typchecker (Prüfungsaufgabe)

Programmieren Sie die Funktionen

```
typecheckStmt::  Stmt -> [(String, Type)] -> [Class] -> Stmt
```

für die abstrakte Syntax folgender reduzierter Sprache:

statement	→	ifthenstatement
	|	ifthenelsestatement
	|	whilestatement
	|	block
	|	emptystatement
	|	returnstatement
ifthenstatement	→	IF LBRACE bexpression RBRACE statement
ifthenelsestatement	→	IF LBRACE bexpression RBRACE statement ELSE statement
whilestatement	→	WHILE LBRACE bexpression RBRACE statement
block	→	LBRACKET RBRACKET
	|	LBRACKET statements RBRACKET
statements	→	statement
	|	statements statement
emptystatement	→	SEMICOLON
returnstatement	→	RETURN SEMICOLON
	|	RETURN expression SEMICOLON
bexpression	→	TRUE
expression	→	1

Beispiel für `If`:

```
typecheckStmt :: Stmt -> [(String, Type)] -> [Class] -> Stmt
typecheckStmt (If(be, ifs, Nothing)) symtab cls =
  let
    bexp = typecheckExpr be symtab cls
    ifstmt = typecheckStmt ifs symtab cls
  in
    if ((getTypeFromExpr bexp) == "boolean") then
      TypedStmt(If(bexp, ifstmt, Nothing), getTypeFromStmt ifstmt)
    else
     error "boolean expected"

getTypeFromExpr :: Expr -> Type
getTypeFromExpr (TypedExpr(_, typ)) = typ

getTypeFromStmt :: Stmt -> Type
getTypeFromStmt (TypedStmt(_, typ)) = typ
```

Hinweis: Für die Expressions müssen Sie eine Funktion

```
typecheckExpr::  Expr -> [(String, Type)] -> [Class] -> Expr
```

programmieren.

5.3 Typregeln II (Prüfungsaufgabe)

Gegeben seien die Klassen aus der Vorlesung:

```
class Cl1 {                    class Cl2 {
   char m1 () {                  Cl3 v;
     int b;                      char m2(Cl3 v, int w) { ...}
     Cl2 x = new Cl2 ()         }
     return x.m2(x.v, b);
   }                           class Cl3 { ...}
}
```

Leiten Sie für den Block

```
{
    int b;
    Cl2 x = new Cl2 ()
    return x.m2(x.v, b);
}
```

mit den Regeln aus der Vorlesung den Typ `char` her.

Hinweis:

$$\dfrac{\dfrac{\vdots \qquad\qquad \vdots}{\{\, b : int \,\} \rhd_{Stmt} \mathsf{Block}(\,\mathsf{LVarDecl}(\,x, CL2\,);\mathsf{Assign}(\,LorFVar(x), \mathsf{New}(\,CL2\,)\,);\mathsf{Return}(\,\mathsf{MethodCall}(\,\mathsf{LorFVar}(\,x\,), m2, (\mathsf{InstVar}(\,\mathsf{LorFVar}(\,x\,), v\,), \mathsf{LorFVar}(\,b\,))\,)\,)\,) : char}\,[\textbf{BlockLVarD}]}{\emptyset \rhd_{Stmt} \mathsf{Block}(\,\mathsf{LVarDecl}(\,b, int\,);\mathsf{LVarDecl}(\,x, CL2\,);\mathsf{Assign}(\,LorFVar(x), \mathsf{New}(\,CL2\,)\,);\mathsf{Return}(\,\mathsf{MethodCall}(\,\mathsf{LorFVar}(\,x\,), m2, (\mathsf{InstVar}(\,\mathsf{LorFVar}(\,x\,), v\,), \mathsf{LorFVar}(\,b\,))\,)\,)\,) : char}\,[\textbf{BlockLVarD}]$$

Die Aufgabe muss entweder an der Tafel (handschiftlich) entwickelt oder als Präsentation so vorgestellt werden, dass ersichtlich ist, welche Schritte nacheinander erfolgen.