Homework 6: Abstract Syntax Tree (20 points)

Study the abstract syntax tree definition for the Mini-Pascal language available in the folder: **Homework 6: Solution/ast.h**.

1. Extend the YACC-based LALR(1) parser with an attribute grammar for creating the abstract syntax tree representation of a Mini-Pascal program; **(10 points)**

2. Verify the correctness of the abstract syntax tree by traversing it in preorder and generating a new program semantically equivalent with the original one. **(10 points)**

You may use the lexical analyser available at **Homework 6: Solution/minipas.l** and the YACC definitions on the next page.

```
%union {
  int integer;
  char *string;
  float number;
  struct tENTRY *symtab;
  struct tN_PROG *prog;
  struct tN_STMT *stmt;
  struct tN_CALL *call_stmt;
  struct tN_ASSIGN *assign_stmt;
  struct tN_IF *if_stmt;
  struct tN_WHILE *while_stmt;
  struct tN_EXPR *exp;
}

%token                  PROG
%token                  VAR ARRAY TO OF
%token                  FUNC PROC
%token                  INT REAL BOOL
%token                  START END
%token                  IF THEN ELSE
%token                  WHILE DO
%token <string>         IDENT
%token <number>         NUM
%token                  COLON SCOLON COMMA DOT
%token                  LBRA RBRA LPAR RPAR
%token                  ASSIGN
%token                  LT LE GT GE EQ NE
%token                  PLUS MINUS TIMES SLASH DIV MOD
%token                  NOT AND OR
%token                  TRUE FALSE

%type <symtab>          varDec varDecList identListType identList type
subProgHead args parList
%type <prog>            subProgList
%type <stmt>            compStmt stmtList stmt elsePart
%type <call_stmt>       subProgCall
%type <assign_stmt>     assStmt
%type <if_stmt>         ifStmt
%type <while_stmt>      whileStmt
%type <exp>             params exprList expr index simpleExpr term factor
%type <integer>         simpleType relOp addOp mulOp

%start                  program
```