Laboratory 9

Yacc specifications:

```
%{
#include <stdio.h>
#include <stdlib.h>
#define YYDEBUG 1
%}

%token ID
%token CONST
%token PLUS
%token MINUS
%token MULTIPLY
%token DIV
%token MOD
%token EQUALS
%token GREATER
%token GREATEROREQUAL
%token SMALLER
%token SMALLEROREQUAL
%token EQUALEQUAL
```

%token DIFFERENT

%token AND

%token OR

%token NOT

%token FALSE

%token TRUE

%token IF

%token ELSE

%token WHILE

%token INT

%token BOOL

%token STRING

%token GET

%token GIVE

%token START

%token STOP

%token MAIN

%token SEMI_COLON

%token SMALL

%token GREAT

%token ROUND_OPEN

%token ROUND_CLOSE

%token SQUARE_OPEN

%token SQUARE_CLOSE

%start program

%%

program : START MAIN SMALL statements GREAT STOP MAIN;

statements: declarationList statementList;

declarationList: declaration declarationList | declaration ;

declaration: type ID SEMI_COLON ;

type: INT|STRING|BOOL ;

statementList: statement statementList | statement ;

statement: ifStm | giveStm | getStm | whileStm | assignStmt ;

term: ID | CONST ;

assignStmt: term EQUALS term SEMI_COLON;

operator : PLUS | MINUS | MOD | DIV | MULTIPLY ;

ifStm : IF ROUND_OPEN condition ROUND_CLOSE SMALL statementList GREAT ;

```
giveStm : GIVE ID SEMI_COLON ;

getStm : GET ID SEMI_COLON ;

whileStm : WHILE ROUND_OPEN condition
ROUND_CLOSE SMALL statementList GREAT ;

condition : TRUE | FALSE | expr ;

expr : term op term ;

op : GREATER | GREATEROREQUAL | SMALLER |
SMALLEROREQUAL | EQUALEQUAL ;


%%


int yyerror(char *s)
{
  printf("%s\n", s);
}


extern FILE *yyin;


int main(int argc, char **argv)
{
  if (argc > 1)
    yyin = fopen(argv[1], "r");
```

```
  if ( (argc > 2) && ( !strcmp(argv[2], "-d") ) )
    yydebug = 1;
  if ( !yyparse() )
    fprintf(stderr,"\t U GOOOOOD !!!\n");
}
```

Flex (updated):

```
%{
#include <stdio.h>
#include <string.h>
#include "y.tab.h"
int lines = 0;
int correct=1;
int badLine=0;
%}

%option noyywrap
%option caseless

NUMBER              [+-]?[1-9][0-9]*|0
STRING          \"[a-zA-Z0-9]*\"
```

```
CONST          {NUMBER}|{STRING}
ID                  [a-zA-Z]+[a-zA-Z0-9_]*


%%
start              {printf("Reserved word: %s\n", yytext); return START;}
stop               {printf("Reserved word: %s\n", yytext); return STOP;}
get              {printf("Reserved word: %s\n", yytext); return GET;}
give         {printf("Reserved word: %s\n", yytext); return GIVE;}
int              {printf("Reserved word: %s\n", yytext); return INT;}
string           {printf("Reserved word: %s\n", yytext); return STRING;}
bool             {printf("Reserved word: %s\n", yytext); return BOOL;}
if               {printf("Reserved word: %s\n", yytext); return IF;}
else         {printf("Reserved word: %s\n", yytext); return ELSE;}
while            {printf("Reserved word: %s\n", yytext); return WHILE;}
```

```
true        {printf("Reserved word: %s\n", yytext); return TRUE;}
false       {printf("Reserved word: %s\n", yytext); return FALSE;}
main        {printf("Reserved word: %s\n", yytext); return MAIN;}
"<"         {printf("Separator: %s\n", yytext); return SMALL;}
">"         {printf("Separator: %s\n", yytext); return GREAT;}
"("         {printf("Separator: %s\n", yytext); return ROUND_OPEN;}
")"         {printf("Separator: %s\n", yytext); return ROUND_CLOSE;}
"["         {printf("Separator: %s\n", yytext); return SQUARE_OPEN;}
"]"         {printf("Separator: %s\n", yytext); return SQUARE_CLOSE;}
";"         {printf("Separator: %s\n", yytext); return SEMI_COLON;}

Plus        {printf( "Operator: %s\n", yytext ); return PLUS;}
```

```
Minus           {printf( "Operator: %s\n", yytext ); return
MINUS;}
Multiply        {printf( "Operator: %s\n", yytext ); return
MULTIPLY;}
Div         {printf( "Operator: %s\n", yytext ); return DIV;}
Mod             {printf( "Operator: %s\n", yytext ); return
MOD;}
Equals          {printf( "Operator: %s\n", yytext ); return
EQUALS;}
Greater  {printf( "Operator: %s\n", yytext ); return
GREATER;}
GreaterOrEqual  {printf( "Operator: %s\n", yytext );
return GREATEROREQUAL;}
Smaller  {printf( "Operator: %s\n", yytext ); return
SMALLER;}
SmallerOrEqual   {printf( "Operator: %s\n", yytext );
return SMALLEROREQUAL;}
EqualEqual   {printf( "Operator: %s\n", yytext ); return
EQUALEQUAL;}
Different       {printf( "Operator: %s\n", yytext ); return
DIFFERENT;}
And             {printf( "Operator: %s\n", yytext ); return
AND;}
```

```
Or              {printf( "Operator: %s\n", yytext );return
OR;}
Not             {printf( "Operator: %s\n", yytext ); return
NOT;}
{ID}        {printf( "Identifier: %s\n", yytext ); return ID;}
{CONST}{printf( "Constant: %s\n", yytext ); return
CONST;}
[ \t]+      {}
[\n]+               {lines++;}
[0-9][0-9]*{ID} {correct=0;
badLine=lines;printf("Incorrect:%s\n",yytext);}
. {correct=0; badLine= lines;
printf("Incorect:%s\n",yytext);}
%%
```

PROBLEM 1:

```
start main <
        int a ;
        int b ;
        int c ;
        int max ;
        get a ;
        get b ;
        get c ;
```

```
        if ( a Greater b ) <
                max Equals a ;>
        if (b Greater max) <
                max Equals b ;
        >

        if ( c Greater max ) <
                max Equals c ;
        >

        give max ;
> stop main
```

OUTPUT:

```
Reserved word: start
Reserved word: main
Separator: <
Reserved word: int
Identifier: a
Separator: ;
Reserved word: int
Identifier: b
Separator: ;
Reserved word: int
Identifier: c
Separator: ;
```

Reserved word: int

Identifier: max

Separator: ;

Reserved word: get

Identifier: a

Separator: ;

Reserved word: get

Identifier: b

Separator: ;

Reserved word: get

Identifier: c

Separator: ;

Reserved word: if

Separator: (

Identifier: a

Operator: Greater

Identifier: b

Separator:)

Separator: <

Identifier: max

Operator: Equals

Identifier: a

Separator: ;

Separator: >

Reserved word: if

Separator: (

Identifier: b

Operator: Greater

Identifier: max

Separator:)

Separator: <

Identifier: max

Operator: Equals

Identifier: b

Separator: ;

Separator: >

Reserved word: if

Separator: (

Identifier: c

Operator: Greater

Identifier: max

Separator:)

Separator: <

Identifier: max

Operator: Equals

Identifier: c

Separator: ;

Separator: >

Reserved word: give

Identifier: max

Separator: ;

Separator: >

Reserved word: stop

Reserved word: main

U GOOOOOD !!!

PROBLEM WITH ERROR:

```
start main <
        int 2a;
        int b;
        int c;
        int max;
        get a;
        get b;
        get c;
        if(a Greater b)<
                max Equals a;
        >
        else<
                max Equals b;
        >
        if(c Greater max)<
                max Equals c; >
        give max;
>stop main
```

OUTPUT:

Reserved word: start

Reserved word: main

Separator: <

Reserved word: int

Incorrect:2a

Separator: ;

syntax error