Exercise 7 - Intermediate code generation using LEX and YACC

Harshini S
185001058

Program Code:

syntax.l

```
%{
#include<stdio.h>
#include<string.h>
#include "y.tab.h"
%}
term ([a-zA-Z\_][a-zA-Z\_0-9]*|[0-9]+)
relop ("<"|"<="|">"|">="|"=="|"!=")
op ("+"|"-"|"*"|"/"|"%")
%%
"if" { return IF;}
"else" { return ELSE;}
"while" { return WHILE; }
"do" { return DO; }
"switch" { return SWITCH; }
"case" { return CASE; }
"default" { return DEFAULT; }
"break" { return BREAK; }
{term} { yylval.str = strdup(yytext); return TERM; }
{relop} { yylval.str = strdup(yytext); return RELOP; }
{op} { yylval.str = strdup(yytext); return OP; }
[ \t\n]+ { }
. { return *yytext;}
%%
```

syntax.y

```
%{
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int yylex(void);
int yyerror(char *);
#include "y.tab.h"
int cc = 1, tc = 1, nc = 1, sc = 0;
%}
%token TERM RELOP OP WHILE DO SWITCH CASE DEFAULT
BREAK IF ELSE %union
```

```
{
        int intval;
        float floatval;
        char *str;
}
%type<str> TERM RELOP OP
%%
line: /* empty */
        | TERM '=' TERM OP TERM ';' { printf("\tt%d := %s %s %s\n\t%s :=
t%d\n", tc, $3, $4, $5, $1, tc); tc++; } line
        | TERM '=' TERM RELOP TERM ';' { printf("\tt%d := %s %s %s\n\t%s :=
t%d\n", tc, $3, $4, $5, $1, tc); tc++; } line
        | TERM '=' TERM ';' { printf("\t%s := %s\n", $1, $3); } line | WHILE TERM
RELOP TERM DO '{' { printf("LABEL%d: if not %s %s %s then goto
FALSE%d\nTRUE%d: ", cc, $2, $3, $4, cc, cc); } line '}' { printf("FALSE%d: ", cc);
cc++; } line
        | WHILE TERM OP TERM DO '{' { printf("LABEL%d: if not %s %s %s then
goto FALSE%d\nTRUE%d: ", cc, $2, $3, $4, cc, cc); } line '}' { printf("FALSE%d:
", cc); cc++; } line
        | WHILE TERM DO '{' { printf("LABEL%d: if not %s then goto
FALSE%d\nTRUE%d: ", cc, $2, cc, cc); } line '}' { printf("FALSE%d: ", cc); cc++; }
line
        | SWITCH '(' TERM RELOP TERM ')' '{' { printf("\tt%d := %s %s %s\n", tc,
$3, $4, $5); sc = tc; tc++; } cases '}' { printf("NEXT%d: ", cc); cc++; } line
        | SWITCH '(' TERM OP TERM ')' '{' { printf("\tt%d := %s %s %s\n", tc, $3, $4,
$5); sc = tc; tc++; } cases '}' { printf("NEXT%d: ", cc); cc++; } line
        | SWITCH '(' TERM ')' '{' { printf("\tt%d := %s\n", tc, $3); sc = tc; tc++; } cases
'}' { printf("NEXT%d: ", cc); cc++; } line | BREAK ';' line { printf("goto NEXT%d\n",
cc); }
        | if
if : IF TERM RELOP TERM '{' {printf("\nif not %s %s %s then goto
FALSE%d\nTRUE%d: ", $2, $3, $4, cc, cc); } line '}' { printf("NEXT%d: ", cc); cc++;
} else

else : line
        | ELSE '{' line '}' line
cases: /* empty */
          | CASE TERM ':' { printf("CASE%d: if not t%d == %s goto CASE%d\n",
nc, sc, $2, nc + 1); nc++; } line cases
          | DEFAULT { printf("CASE%d: ", nc); nc++; } ':' line
%%
int yyerror(char* s)
{
   fprintf(stderr, "%s\n", s);
```

```
    return 0;
}
int yywrap()
{
    return 1;
}
int main()
{
    yyparse();
    printf("\n");
    return 0;
}
```