**Name:** Paryusha Shah

**Roll No:** 22bce234

**Course:** Principles of Compiler Design

**Course Code:** 4CS501CC25

***Practical 7: Write semantic rules to generate three address codes for control statements.***

**File:** expr.h

#ifndef EXPR\_H

#define EXPR\_H

typedef struct Expr {

    char\* place;

} Expr;

/\* Label & temporary counters \*/

static int temp\_count = 0;

static int label\_count = 0;

static char\* newtemp() {

    static char temp[10];

    sprintf(temp, "t%d", ++temp\_count);

    return strdup(temp);

}

static char\* newlabel() {

    static char label[10];

    sprintf(label, "L%d", ++label\_count);

    return strdup(label);

}

#endif

**File:** prog7.l

%{

#include "expr.h"

#include "y.tab.h"

#include <stdlib.h>

#include <string.h>

%}

%%

"if"            { return IF; }

"else"          { return ELSE; }

"while"         { return WHILE; }

"int"           { return INT; }

[a-zA-Z\_][a-zA-Z0-9\_]\*    { yylval.id = strdup(yytext); return ID; }

[0-9]+                    { yylval.num = atoi(yytext); return NUM; }

"{"                       { return LBRACE; }

"}"                       { return RBRACE; }

"("                       { return LPAREN; }

")"                       { return RPAREN; }

"="                       { return ASSIGN; }

";"                       { return SEMI; }

"+"                       { return PLUS; }

"-"                       { return MINUS; }

"\*"                       { return MUL; }

"/"                       { return DIV; }

[ \t\n]+                  { /\* skip whitespace \*/ }

.                         { printf("Unknown symbol: %s\n", yytext); }

%%

int yywrap() { return 1; }

**File:** prog7.y

%{

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "expr.h"

int yylex(void);

extern FILE \*yyin;

void yyerror(char \*s) { fprintf(stderr, "Error: %s\n", s); }

%}

%union {

    int num;

    char\* id;

    Expr\* expr;

}

%token <id> ID

%token <num> NUM

%token IF ELSE WHILE

%token INT

%token PLUS MINUS MUL DIV ASSIGN SEMI LPAREN RPAREN LBRACE RBRACE

%type <expr> E

%nonassoc LOWER\_THAN\_ELSE

%nonassoc ELSE

%left PLUS MINUS

%left MUL DIV

%%

program:

    stmt\_list

    ;

stmt\_list:

    stmt\_list S

    | /\* empty \*/

    ;

S:

    ID ASSIGN E SEMI {

        printf("%s = %s\n", $1, $3->place);

    }

    | IF LPAREN E RPAREN S %prec LOWER\_THAN\_ELSE {

        char\* L1 = newlabel();

        char\* L2 = newlabel();

        printf("ifFalse %s goto %s\n", $3->place, L1);

        printf("goto %s\n", L2);

        printf("%s:\n", L1);

        printf("%s:\n", L2);

    }

    | IF LPAREN E RPAREN S ELSE S {

        char\* L1 = newlabel();

        char\* L2 = newlabel();

        printf("ifFalse %s goto %s\n", $3->place, L1);

        printf("goto %s\n", L2);

        printf("%s:\n", L1);

        printf("%s:\n", L2);

    }

    | WHILE LPAREN E RPAREN S {

        char\* L1 = newlabel();

        char\* L2 = newlabel();

        printf("%s:\n", L1);

        printf("ifFalse %s goto %s\n", $3->place, L2);

        printf("goto %s\n", L1);

        printf("%s:\n", L2);

    }

    | LBRACE stmt\_list RBRACE

    ;

E:

    E PLUS E {

        $$ = malloc(sizeof(Expr));

        $$->place = newtemp();

        printf("%s = %s + %s\n", $$->place, $1->place, $3->place);

    }

    | E MINUS E {

        $$ = malloc(sizeof(Expr));

        $$->place = newtemp();

        printf("%s = %s - %s\n", $$->place, $1->place, $3->place);

    }

    | E MUL E {

        $$ = malloc(sizeof(Expr));

        $$->place = newtemp();

        printf("%s = %s \* %s\n", $$->place, $1->place, $3->place);

    }

    | E DIV E {

        $$ = malloc(sizeof(Expr));

        $$->place = newtemp();

        printf("%s = %s / %s\n", $$->place, $1->place, $3->place);

    }

    | LPAREN E RPAREN { $$ = $2; }

    | NUM {

        $$ = malloc(sizeof(Expr));

        $$->place = malloc(20);

        sprintf($$->place, "%d", $1);

    }

    | ID {

        $$ = malloc(sizeof(Expr));

        $$->place = strdup($1);

    }

    ;

%%

int main() {

    FILE \*fp = fopen("input.c", "r");

    if(!fp) { perror("Cannot open file input.c"); return 1; }

    yyin = fp;

    yyparse();

    fclose(fp);

    return 0;

}

**File:** input.c

a = b + c \* 2;

if (a) {

    x = y + 1;

} else {

    y = y + 2;

}

while (x) {

    x = x - 1;

}

**Commands for run program:**

