```
%{
#include<iostream>
#include<cstdlib>
#include<cstring>
#include "SymbolTable.h"
#define YYSTYPE SymbolInfo*

using namespace std;

int yyparse(void);
int yylex(void);

SymbolTable *table1=new SymbolTable();
SymbolTable *table2=new SymbolTable();
extern int line_count;
extern int error_count;
FILE *out;
FILE *out1;
SymbolInfo *sm = new SymbolInfo();
SymbolInfo *multiple = new SymbolInfo();
SymbolInfo *arr_ind = new SymbolInfo();
SymbolInfo *pr_id = new SymbolInfo();
SymbolInfo *asgn = new SymbolInfo();
int scope=1;


void yyerror(const char *s)
{
	//write your code
	//fprintf(stderr,"****error at line %d: %s\n",line_count,s);
	fprintf(out,"****error at line %d: %s\n",line_count,s);
	error_count++;
	return;
}


%}

%token delim newlinw IF ELSE FOR WHILE INT FLOAT DOUBLE CHAR RETURN VOID
MAIN CONTINUE PRINTLN
%token ADDOP MULOP ASSIGNOP RELOP LOGICOP NOT SEMICOLON COMMA LPAREN
RPAREN LCURL RCURL LTHIRD
%token RTHIRD INCOP DECOP CONST_INT CONST_FLOAT ID
%error-verbose

//%left
//%right

//%nonassoc


%%
```

```
start : func_declaration global_var_declaration Program
           {fprintf(out, "\n\nStart : func_declaration
global_var_declaration Program\n");}

        | global_var_declaration func_declaration Program
           {fprintf(out, "\n\nStart : global_var_declaration
func_declaration Program\n");}

           | func_declaration Program
           {fprintf(out, "\n\nStart : func_declaration Program\n");}

           | global_var_declaration Program
           {fprintf(out, "\n\nStart : global_var_declaration
Program\n");}

           | Program
           {fprintf(out, "\n\nStart : Program\n");}

           ;


global_var_declaration : type_specifier global_dec_list SEMICOLON
                 {fprintf(out, "\n\nglobal_var_declaration :
type_specifier global_dec_list SEMICOLON\n");}

                    | global_var_declaration type_specifier
global_dec_list SEMICOLON
                 {fprintf(out, "\n\nglobal_var_declaration :
global_var_declaration type_specifier global_dec_list SEMICOLO\n");}

                 ;

global_dec_list : global_dec_list COMMA ID
                 {fprintf(out, "\n\nglobal_dec_list : global_dec_list
COMMA ID\n");

                      fprintf(out,"%s\n",$3->name.c_str());
                      fprintf(out1,"%s\n",$3->name.c_str());
                      multiple = table1->Lookup($3->name);
                      if(multiple==NULL)
                      {
                            if(strcmp(sm->name.c_str(),"int")==0)
                            {
                                  $3->dataType = 0;
                                  $3->v.i = -99999;
                            }
                            else if(strcmp(sm->name.c_str(),"float")==0)
                            {
                                  $3->dataType = 1;
                                  $3->v.f = -99999.000000;
                            }

                            table1->insert($3);
                      }
```

```
                        else
                        {
                              char msg[30] = "Multiple Declaration For ID:
";
                              strcat(msg,$3->name.c_str());
                              yyerror(msg);
                        }
                  }
            | global_dec_list COMMA ID LTHIRD CONST_INT RTHIRD
                  {fprintf(out, "\n\nglobal_dec_list : global_dec_list
COMMA ID LTHIRD CONST_INT RTHIRD\n");

                        fprintf(out,"%s\n",$3->name.c_str());
                        fprintf(out1,"%s\n",$3->name.c_str());
                        multiple = table1->Lookup($3->name);
                        if(multiple==NULL)
                        {
                              if(strcmp(sm->name.c_str(),"int")==0)
                              {
                                    $3->dataType = 0;
                                    $3->sz = $5->v.i;
                                    $3->v.i_Arr = (int*)malloc(($3-
>sz)*sizeof(int));
                                    for(int k = 0; k<$3->sz; k++) $3-
>v.i_Arr[k] = -1;
                              }
                              else if(strcmp(sm->name.c_str(),"float")==0)
                              {
                                    $3->dataType = 1;
                                    $3->sz = $5->v.i;
                                    $3->v.f_Arr = (float*)malloc(($3-
>sz)*sizeof(float));
                                    for(int k = 0; k<$3->sz; k++) $3-
>v.f_Arr[k] = -1.00000;
                              }

                              table1->insert($3);
                        }
                        else
                        {
                              char msg[30] = "Multiple Declaration For ID:
";
                              strcat(msg,$3->name.c_str());
                              yyerror(msg);
                        }
                  }
            | global_dec_list COMMA ID LTHIRD error RTHIRD
            | ID  {fprintf(out, "\n\nglobal_dec_list : ID\n");

                        fprintf(out,"%s\n",$1->name.c_str());
                        fprintf(out1,"%s\n",$1->name.c_str());
                        multiple = table1->Lookup($1->name);
                        if(multiple==NULL)
                        {
```

```
                              if(strcmp(sm->name.c_str(),"int")==0)
                              {
                                    $1->dataType = 0;
                                    $1->v.i = -99999;
                              }
                              else if(strcmp(sm->name.c_str(),"float")==0)
                              {
                                    $1->dataType = 1;
                                    $1->v.f = -99999.000000;
                              }
                              table1->insert($1);
                        }
                        else
                        {
                              char msg[30] = "Multiple Declaration For ID:
";
                              strcat(msg,$1->name.c_str());
                              yyerror(msg);
                        }
                  }
            | ID LTHIRD CONST_INT RTHIRD
                  {fprintf(out, "\n\nglobal_dec_list : ID LTHIRD CONST_INT
RTHIRD\n");

                        fprintf(out,"%s\n",$1->name.c_str());
                        fprintf(out1,"%s\n",$1->name.c_str());
                        multiple = table1->Lookup($1->name);
                        if(multiple==NULL)
                        {
                              if(strcmp(sm->name.c_str(),"int")==0)
                              {
                                    $1->dataType = 0;
                                    $1->sz = $3->v.i;
                                    $1->v.i_Arr = (int*)malloc(($1-
>sz)*sizeof(int));
                                    for(int k = 0; k<$1->sz; k++) $1-
>v.i_Arr[k] = -1;
                              }
                              else if(strcmp(sm->name.c_str(),"float")==0)
                              {
                                    $1->dataType = 1;
                                    $1->sz = $3->v.i;
                                    $1->v.f_Arr = (float*)malloc(($1-
>sz)*sizeof(float));
                                    for(int k = 0; k<$1->sz; k++) $1-
>v.f_Arr[k] = -1.000000;
                              }
                              table1->insert($1);
                        }
                        else
                        {
                              char msg[30] = "Multiple Declaration For ID:
";
                              strcat(msg,$1->name.c_str());
```

```
                        yyerror(msg);
                    }
                }
            | ID LTHIRD error RTHIRD
            ;


func_declaration : type_specifier ID LPAREN Parameter_List RPAREN
SEMICOLON
                {fprintf(out, "\n\nfunc_declaration : type_specifier ID
LPAREN Parameter_List RPAREN SEMICOLON\n");}

             | type_specifier error LPAREN error RPAREN error SEMICOLON
             | type_specifier ID LPAREN RPAREN SEMICOLON
                {fprintf(out, "\n\nfunc_declaration : type_specifier ID
LPAREN RPAREN SEMICOLON\n");}

             ;

Parameter_List : Parameter_List COMMA Parameter
                {fprintf(out, "\n\nParameter_List : Parameter_List COMMA
Parameter\n");}

              | Parameter  {fprintf(out, "\n\nParameter_List :
Parameter\n");}

              ;

Parameter : type_specifier ID
            {fprintf(out, "\n\nParameter : type_specifier ID\n");}

        | type_specifier ID LTHIRD RTHIRD
                {fprintf(out, "\n\nParameter : type_specifier ID LTHIRD
CONST_INT RTHIRD\n");}

        | type_specifier error LTHIRD error RTHIRD
        ;

Program : INT MAIN LPAREN RPAREN compound_statement
            {fprintf(out, "\n\nProgram: INT MAIN LPAREN RPAREN
compound_statement\n");}

      ;

compound_statement : LCURL var_declaration statements RCURL
                        {fprintf(out, "\n\ncompound_statement: LCURL
var_declaration statements RCURL\n");

                        scope = 2;
                        }
               | LCURL statements RCURL
                        {fprintf(out, "\n\ncompound_statement: LCURL
statements RCURL\n");
```

```
                        scope = 2;
                    }
            | LCURL RCURL
                    {fprintf(out, "\n\ncompound_statement: LCURL
RCURL\n");

                        scope = 2;
                    }
            ;

var_declaration  : type_specifier declaration_list SEMICOLON
                 {fprintf(out, "\n\nvar_declaration: type_specifier
declaration_list SEMICOLON\n");}

           |  var_declaration type_specifier declaration_list SEMICOLON
                {fprintf(out, "\n\nvar_declaration: var_declaration
type_specifier declaration_list SEMICOLON\n");}

           ;


type_specifier   : INT      {fprintf(out, "\n\ntype_specifier: INT\n");

                        sm->name = "int";
                    }
           | FLOAT          {fprintf(out, "\n\ntype_specifier:
FLOAT\n");

                        sm->name = "float";
                    }
           | VOID           {fprintf(out, "\n\ntype_specifier: VOID\n");

                        sm->name = "void";
                    }
           ;

declaration_list : declaration_list COMMA ID  {fprintf(out,
"\n\ndeclaration_list: declaration_list COMMA ID\n");

                                        fprintf(out,"%s\n",$3-
>name.c_str());
                                        fprintf(out1,"%s\n",$3-
>name.c_str());
                                        multiple = table2->Lookup($3-
>name);
                                        if(multiple==NULL)
                                        {
                                              if(strcmp(sm-
>name.c_str(),"int")==0)
                                              {
                                                    $3->dataType = 0;
                                                    $3->v.i = -99999;
                                              }
```

```
                                                else if(strcmp(sm-
>name.c_str(),"float")==0)
                                                {
                                                      $3->dataType = 1;
                                                      $3->v.f = -
99999.000000;
                                                }
                                                table2->insert($3);
                                          }
                                          else
                                          {
                                                char msg[30] = "Multiple
Declaration For ID: ";
                                                strcat(msg,$3-
>name.c_str());
                                                yyerror(msg);
                                          }
                                    }
            | declaration_list COMMA ID LTHIRD CONST_INT RTHIRD
                {fprintf(out, "\n\ndeclaration_list: declaration_list
COMMA ID LTHIRD CONST_INT RTHIRD\n");

                        fprintf(out,"%s\n",$3->name.c_str());
                        fprintf(out1,"%s\n",$3->name.c_str());
                        multiple = table2->Lookup($3->name);
                        if(multiple==NULL)
                        {
                              if(strcmp(sm->name.c_str(),"int")==0)
                              {
                                    $3->dataType = 0;
                                    $3->sz = $5->v.i;
                                    $3->v.i_Arr = (int*)malloc(($3-
>sz)*sizeof(int));
                                    for(int k = 0; k<$3->sz; k++) $3-
>v.i_Arr[k] = -1;
                              }
                              else if(strcmp(sm->name.c_str(),"float")==0)
                              {
                                    $3->dataType = 1;
                                    $3->sz = $5->v.i;
                                    $3->v.f_Arr = (float*)malloc(($3-
>sz)*sizeof(float));
                                    for(int k = 0; k<$3->sz; k++) $3-
>v.f_Arr[k] = -1.00000;
                              }

                              table2->insert($3);
                        }
                        else
                        {
                              char msg[30] = "Multiple Declaration For ID:
";
                              strcat(msg,$3->name.c_str());
                              yyerror(msg);
```

```
                        }
                    }
                | ID   {fprintf(out, "\n\ndeclaration_list: ID\n");

                        fprintf(out,"%s\n",$1->name.c_str());
                        fprintf(out1,"%s\n",$1->name.c_str());
                        multiple = table2->Lookup($1->name);
                        if(multiple==NULL)
                        {
                              if(strcmp(sm->name.c_str(),"int")==0)
                              {
                                    $1->dataType = 0;
                                    $1->v.i = -99999;
                              }
                              else if(strcmp(sm->name.c_str(),"float")==0)
                              {
                                    $1->dataType = 1;
                                    $1->v.f = -99999.000000;
                              }
                              table2->insert($1);
                        }
                        else
                        {
                              char msg[30] = "Multiple Declaration For ID:
";
                              strcat(msg,$1->name.c_str());
                              yyerror(msg);
                        }
                    }
                | ID LTHIRD CONST_INT RTHIRD
                    {fprintf(out, "\n\ndeclaration_list: ID LTHIRD CONST_INT
RTHIRD\n");

                        fprintf(out,"%s\n",$1->name.c_str());
                        fprintf(out1,"%s\n",$1->name.c_str());
                        multiple = table2->Lookup($1->name);
                        if(multiple==NULL)
                        {
                              if(strcmp(sm->name.c_str(),"int")==0)
                              {
                                    $1->dataType = 0;
                                    $1->sz = $3->v.i;
                                    $1->v.i_Arr = (int*)malloc(($1-
>sz)*sizeof(int));
                                    for(int k = 0; k<$1->sz; k++) $1-
>v.i_Arr[k] = -1;
                              }
                              else if(strcmp(sm->name.c_str(),"float")==0)
                              {
                                    $1->dataType = 1;
                                    $1->sz = $3->v.i;
                                    $1->v.f_Arr = (float*)malloc(($1-
>sz)*sizeof(float));
```

```
                                    for(int k = 0; k<$1->sz; k++) $1-
>v.f_Arr[k] = -1.000000;
                              }
                              table2->insert($1);
                        }
                        else
                        {
                              char msg[30] = "Multiple Declaration For ID:
";
                              strcat(msg,$1->name.c_str());
                              yyerror(msg);
                        }
                  }
             | declaration_list COMMA ID error LTHIRD error RTHIRD
             | ID error LTHIRD error RTHIRD
             ;

statements : statement       {fprintf(out, "\n\nstatements:
statement\n");

                        }
         | statements statement    {fprintf(out, "\n\nstatements:
statements statement\n");

                              }
         ;

statement  : expression_statement {fprintf(out, "\n\nstatement:
expression_statement\n");}
         | compound_statement           {fprintf(out, "\n\nstatement:
compound_statement\n");}
         | FOR LPAREN expression_statement expression_statement
expression RPAREN statement
            {fprintf(out, "\n\nstatement: FOR LPAREN expression_statement
expression_statement expression RPAREN statement\n");

            }
         | FOR error LPAREN expression_statement expression_statement
error RPAREN statement
         | IF LPAREN expression RPAREN statement
            {fprintf(out,"\n\nstatement: IF LPAREN expression RPAREN
statement\n");

            }
         | IF error LPAREN error RPAREN statement

         | IF LPAREN expression RPAREN statement ELSE statement
            {fprintf(out,"\n\nstatement: IF LPAREN expression RPAREN
statement ELSE statement\n");

            }
         | IF error LPAREN error RPAREN statement ELSE statement
         | WHILE LPAREN expression RPAREN statement
```

```
            {fprintf(out,"\n\nstatement: WHILE LPAREN expression RPAREN
statement\n");

            }
          | WHILE error LPAREN error RPAREN statement
          | PRINTLN LPAREN ID RPAREN SEMICOLON
            {fprintf(out,"\n\nstatement: PRINTLN LPAREN ID RPAREN
SEMICOLON\n");

                  pr_id = table2->Lookup($3->name);
                  if(pr_id!=NULL)
                  {
                        if(pr_id->dataType==0)printf("%d\n",pr_id->v.i);
                        else if(pr_id->dataType==1)printf("%f\n",pr_id-
>v.f);
                  }
                  else
                  {
                        char msg[30] = "Undeclared Identifier: ";
                        strcat(msg,$3->name.c_str());
                        yyerror(msg);
                  }

            }
          | RETURN expression SEMICOLON
            {fprintf(out,"\n\nstatement: RETURN expression SEMICOLON\n");

            }
          | RETURN error SEMICOLON
          ;

expression_statement   : SEMICOLON{fprintf(out,
"\n\nexpression_statement: SEMICOLON\n");

                              }
                  | expression SEMICOLON  {fprintf(out,
"\n\nexpression_statement: expression SEMICOLON\n");

                                    }
                  | error SEMICOLON
                  ;

variable : ID                       {fprintf(out, "\n\nvariable: ID\n");
fprintf(out,"%s\n",$1->name.c_str());
                                    asgn = table2->Lookup($1->name);
                                    if(asgn!=NULL)
                                    {
                                          if(asgn->sz!=0)
yyerror("identifier to an array");
                                          else $$ = table2->Lookup($1-
>name);
                                    }
                                    else
                                    {
```

```
                                        char msg[30] = "Undeclared
Identifier: ";
                                        strcat(msg,$1->name.c_str());
                                        yyerror(msg);
                                  }
                            }
      | ID LTHIRD expression RTHIRD  {fprintf(out, "\n\nvariable: ID
LTHIRD expression RTHIRD\n");
                                  fprintf(out,"%s\n",$1->name.c_str());
                                  asgn = table2->Lookup($1->name);
                                  if(asgn!=NULL)
                                  {
                                        if(asgn->sz>0)
                                        {
                                              arr_ind = $3;
                                              if(($3->dataType==0)&&($3-
>v.i>=0&&$3->v.i<asgn->sz))
                                              {
                                                    $$ = table2-
>Lookup($1->name);
                                              }
                                              else if($3-
>dataType==1)yyerror("index can not be a float");
                                              else yyerror("array index
out-of-bound");
                                        }
                                        else
                                        {
                                              yyerror("not an array");
                                        }
                                  }
                                  else
                                  {
                                        char msg[30] = "Undeclared
Identifier: ";
                                        strcat(msg,$1->name.c_str());
                                        yyerror(msg);
                                  }
                            }
      | ID error LTHIRD error RTHIRD
      ;

expression : logic_expression           {fprintf(out,"\n\nexpression:
logic_expression\n");
                              fprintf(out1,"\n\nexpression:
logic_expression\n");
                            }
        | variable ASSIGNOP logic_expression
           {fprintf(out,"\n\nexpression: variable ASSIGNOP
logic_expression\n");
            fprintf(out1,"\n\nexpression: variable ASSIGNOP
logic_expression\n");
                if($3->dataType == 3) {}
                else if(!($1->dataType==0 && $3->dataType==1))
```

```
                {
                        if($1->sz == 0)
                        {
                                if($1->dataType == 0 && $3->dataType == 0)
                                        $1->v.i=$3->v.i;

                                else if($1->dataType == 1 && $3->dataType ==
0)
                                        $1->v.f=$3->v.i;
                                else if($1->dataType == 1 && $3->dataType ==
1)
                                        $1->v.f=$3->v.f;

                        }
                        else
                        {

                                if($1->dataType == 0 && $3->dataType == 0)
                                        $1->v.i_Arr[arr_ind->v.i]=$3->v.i;

                                else if($1->dataType == 1 && $3->dataType ==
0)
                                        $1->v.f_Arr[arr_ind->v.i]=$3->v.i;
                                else if($1->dataType == 1 && $3->dataType ==
1)
                                        $1->v.f_Arr[arr_ind->v.i]=$3->v.f;

                        }
                        $$=$1;
                        table2->print(out);
                        table2->print(out1);
                }
                else yyerror("Type Mismatch");
           }
         ;

logic_expression : rel_expression {fprintf(out,"\n\nlogic_expression:
rel_expression\n");
                                fprintf(out1,"\n\nlogic_expression:
rel_expression\n");
                                }
            | rel_expression LOGICOP rel_expression
                {fprintf(out,"\n\nlogic_expression: rel_expression
LOGICOP rel_expression\n");
                 fprintf(out1,"\n\nlogic_expression: rel_expression
LOGICOP rel_expression\n");
                        $$->dataType = 0;
                        if(strcmp($2->name.c_str(),"&&")==0)
                        {
                                if($1->dataType == 0 && $3->dataType == 0)
                                        $$->v.i = $1->v.i&&$3->v.i;
                                else if($1->dataType == 0 && $3->dataType ==
1)
                                        $$->v.i = $1->v.i&&$3->v.f;
```

```
                              else if($1->dataType == 1 && $3->dataType ==
0)
                                    $$->v.i = $1->v.f&&$3->v.i;
                              else if($1->dataType == 1 && $3->dataType ==
1)
                                    $$->v.i = $1->v.f&&$3->v.f;

                        }
                        else if(strcmp($2->name.c_str(),"||")==0)
                        {
                              if($1->dataType == 0 && $3->dataType == 0)
                                    $$->v.i = $1->v.i||$3->v.i;
                              else if($1->dataType == 0 && $3->dataType ==
1)
                                    $$->v.i = $1->v.i||$3->v.f;

                              else if($1->dataType == 1 && $3->dataType ==
0)
                                    $$->v.i = $1->v.f||$3->v.i;
                              else if($1->dataType == 1 && $3->dataType ==
1)
                                    $$->v.i = $1->v.f||$3->v.f;

                        }
                  }
             ;

rel_expression   : simple_expression    {fprintf(out,"\n\nrel_expression:
simple_expression\n");
                              fprintf(out1,"\n\nrel_expression:
simple_expression\n");
                              }
            | simple_expression RELOP simple_expression
                  {fprintf(out,"\n\nrel_expression: simple_expression
RELOP simple_expression\n");
                   fprintf(out1,"\n\nrel_expression: simple_expression
RELOP simple_expression\n");
                        $$->dataType = 0;
                        if(strcmp($2->name.c_str(),"<")==0)
                        {
                              if($1->dataType == 0 && $3->dataType == 0)
                                    $$->v.i = $1->v.i<$3->v.i;
                              else if($1->dataType == 0 && $3->dataType ==
1)
                                    $$->v.i = $1->v.i<$3->v.f;

                              else if($1->dataType == 1 && $3->dataType ==
0)
                                    $$->v.i = $1->v.f<$3->v.i;
                              else if($1->dataType == 1 && $3->dataType ==
1)
                                    $$->v.i = $1->v.f<$3->v.f;
```

```
                        }
                        else if(strcmp($2->name.c_str(),">")==0)
                        {
                              if($1->dataType == 0 && $3->dataType == 0)
                                    $$->v.i = $1->v.i>$3->v.i;
                              else if($1->dataType == 0 && $3->dataType ==
1)
                                    $$->v.i = $1->v.i>$3->v.f;

                              else if($1->dataType == 1 && $3->dataType ==
0)
                                    $$->v.i = $1->v.f>$3->v.i;
                              else if($1->dataType == 1 && $3->dataType ==
1)
                                    $$->v.i = $1->v.f>$3->v.f;

                        }
                        else if(strcmp($2->name.c_str(),"<=")==0)
                        {
                              if($1->dataType == 0 && $3->dataType == 0)
                                    $$->v.i = $1->v.i<=$3->v.i;
                              else if($1->dataType == 0 && $3->dataType ==
1)
                                    $$->v.i = $1->v.i<=$3->v.f;

                              else if($1->dataType == 1 && $3->dataType ==
0)
                                    $$->v.i = $1->v.f<=$3->v.i;
                              else if($1->dataType == 1 && $3->dataType ==
1)
                                    $$->v.i = $1->v.f<=$3->v.f;

                        }
                        else if(strcmp($2->name.c_str(),">=")==0)
                        {
                              if($1->dataType == 0 && $3->dataType == 0)
                                    $$->v.i = $1->v.i>=$3->v.i;
                              else if($1->dataType == 0 && $3->dataType ==
1)
                                    $$->v.i = $1->v.i>=$3->v.f;

                              else if($1->dataType == 1 && $3->dataType ==
0)
                                    $$->v.i = $1->v.f>=$3->v.i;
                              else if($1->dataType == 1 && $3->dataType ==
1)
                                    $$->v.i = $1->v.f>=$3->v.f;

                        }
                        else if(strcmp($2->name.c_str(),"==")==0)
                        {
                              if($1->dataType == 0 && $3->dataType == 0)
                                    $$->v.i = $1->v.i==$3->v.i;
```

```
                              else if($1->dataType == 0 && $3->dataType ==
1)
                                    $$->v.i = $1->v.i==$3->v.f;

                              else if($1->dataType == 1 && $3->dataType ==
0)
                                    $$->v.i = $1->v.f==$3->v.i;
                              else if($1->dataType == 1 && $3->dataType ==
1)
                                    $$->v.i = $1->v.f==$3->v.f;

                        }
                        else if(strcmp($2->name.c_str(),"!=")==0)
                        {
                              if($1->dataType == 0 && $3->dataType == 0)
                                    $$->v.i = $1->v.i!=$3->v.i;
                              else if($1->dataType == 0 && $3->dataType ==
1)
                                    $$->v.i = $1->v.i!=$3->v.f;

                              else if($1->dataType == 1 && $3->dataType ==
0)
                                    $$->v.i = $1->v.f!=$3->v.i;
                              else if($1->dataType == 1 && $3->dataType ==
1)
                                    $$->v.i = $1->v.f!=$3->v.f;

                        }
                  }
            ;

simple_expression : term          {fprintf(out,"\n\nsimple_expression:
term\n");
                               fprintf(out1,"\n\nsimple_expression:
term\n");
                              }
              | simple_expression ADDOP term
{fprintf(out,"\n\nsimple_expression: simple_expression ADDOP term\n");
                                    fprintf(out1,"\n\nsimple_expression:
simple_expression ADDOP term\n");

                                          if(strcmp($2-
>name.c_str(),"+")==0)
                                          {
                                                if($1->dataType == 0 && $3-
>dataType == 0)
                                                      {$$->dataType = 0;$$-
>v.i = $1->v.i+$3->v.i;}
                                                else if($1->dataType == 0
&& $3->dataType == 1)
                                                      {$$->dataType = 1;$$-
>v.f = $1->v.i+$3->v.f;}
```

```
                                                else if($1->dataType == 1
&& $3->dataType == 0)
                                                      {$$->dataType = 1;$$-
>v.f = $1->v.f+$3->v.i;}
                                                else if($1->dataType == 1
&& $3->dataType == 1)
                                                      {$$->dataType = 1;$$-
>v.f = $1->v.f+$3->v.f;}

                                          }
                                          else if(strcmp($2-
>name.c_str(),"-")==0)
                                          {
                                                if($1->dataType == 0 && $3-
>dataType == 0)
                                                      {$$->dataType = 0;$$-
>v.i = $1->v.i-$3->v.i;}
                                                else if($1->dataType == 0
&& $3->dataType == 1)
                                                      {$$->dataType = 1;$$-
>v.f = $1->v.i-$3->v.f;}

                                                else if($1->dataType == 1
&& $3->dataType == 0)
                                                      {$$->dataType = 1;$$-
>v.f = $1->v.f-$3->v.i;}
                                                else if($1->dataType == 1
&& $3->dataType == 1)
                                                      {$$->dataType = 1;$$-
>v.f = $1->v.f-$3->v.f;}

                                          }
                                    }
            ;

term :     unary_expression       {fprintf(out,"\n\nterm:
unary_expression\n");
                                fprintf(out1,"\n\nterm:
unary_expression\n");
                               }
     |  term MULOP unary_expression     {fprintf(out,"\n\nterm: term
MULOP unary_expression\n");
                                fprintf(out1,"\n\nterm: term MULOP
unary_expression\n");

                                    if(strcmp($2->name.c_str(),"*")==0)
                                    {
                                          if($1->dataType == 0 && $3-
>dataType == 0)
                                                {$$->dataType = 0; $$->v.i
= $1->v.i*$3->v.i;}
                                          else if($1->dataType == 0 && $3-
>dataType == 1)
```

```
                                        {$$->dataType = 1; $$->v.f
= $1->v.i*$3->v.f;}

                                  else if($1->dataType == 1 && $3-
>dataType == 0)
                                        {$$->dataType = 1; $$->v.f
= $1->v.f*$3->v.i;}
                                  else if($1->dataType == 1 && $3-
>dataType == 1)
                                        {$$->dataType = 1; $$->v.f
= $1->v.f*$3->v.f;}

                              }
                              else if(strcmp($2-
>name.c_str(),"/")==0)
                              {
                                  if($1->dataType == 0 && $3-
>dataType == 0)
                                        {$$->dataType = 0; $$->v.i
= $1->v.i/$3->v.i;}
                                  else if($1->dataType == 0 && $3-
>dataType == 1)
                                        {$$->dataType = 1; $$->v.f
= $1->v.i/$3->v.f;}

                                  else if($1->dataType == 1 && $3-
>dataType == 0)
                                        {$$->dataType = 1; $$->v.f
= $1->v.f/$3->v.i;}
                                  else if($1->dataType == 1 && $3-
>dataType == 1)
                                        {$$->dataType = 1; $$->v.f
= $1->v.f/$3->v.f;}

                              }
                              else
                              {
                                  if(!($1->dataType==1 || $3-
>dataType==1))
                                  {
                                        if($1->dataType == 0 && $3-
>dataType == 0)
                                              {$$->dataType = 0;
$$->v.i = $1->v.i%$3->v.i;}

                                  }
                                  else
                                  {
                                        $$->dataType = 3;
                                        yyerror("invalid operands
for modulo operation");
                                  }
                              }
                        }
```

```
     ;

unary_expression : ADDOP unary_expression
{fprintf(out,"\n\nunary_expression: ADDOP unary_expression\n");
                                fprintf(out1,"\n\nunary_expression: ADDOP
unary_expression\n");
                                   if(strcmp($1->name.c_str(),"-")==0)
                                   {
                                         if($1->dataType == 0) {$2->v.i=-
$2->v.i; $$=$2;}
                                         else if($1->dataType == 1) {$2-
>v.f=-$2->v.f; $$=$2;}

                                   }
                                }
             | NOT unary_expression   {fprintf(out,"\n\nunary_expression:
NOT unary_expression\n");
                                fprintf(out1,"\n\nunary_expression: NOT
unary_expression\n");
                                   if($1->dataType == 0) {$2->v.i=!$2-
>v.i; $$=$2;}
                                   else if($1->dataType == 1) {$2-
>v.f=!$2->v.f; $$=$2;}

                             }
             | factor {fprintf(out,"\n\nunary_expression: factor\n");
                     fprintf(out1,"\n\nunary_expression: factor\n");
                        $$ = new SymbolInfo();
                        $$->name = $1->name;
                        $$->type = $1->type;
                        $$->dataType = $1->dataType;
                        $$->index = $1->index;
                        $$->sz = $1->sz;
                        if($1->sz == 0)
                        {
                              if($1->dataType==0)$$->v.i = $1->v.i;
                              else if($1->dataType==1)$$->v.f = $1->v.f;

                        }
                        else
                        {
                              if($1->dataType==0)$$->v.i = $1-
>v.i_Arr[arr_ind->v.i];
                              else if($1->dataType==1)$$->v.f = $1-
>v.f_Arr[arr_ind->v.i];

                        }
                  }
             ;

factor      : variable {fprintf(out,"\n\nfactor: variable\n");
                 fprintf(out1,"\n\nfactor: variable\n");
               }
```

```
      | LPAREN expression RPAREN   {fprintf(out,"\nfactor: LPAREN
expression RPAREN\n");
                                  fprintf(out,"\nfactor: LPAREN expression
RPAREN\n");
                                  $$=$2;
                                 }
      | CONST_INT    {fprintf(out,"\n\nfactor: CONST_INT\n");
                  fprintf(out1,"\n\nfactor: CONST_INT\n");
                        fprintf(out,"%d\n",$1->v.i);
                        fprintf(out1,"%d\n",$1->v.i);
                  }
      | CONST_FLOAT  {fprintf(out,"\n\nfactor: CONST_FLOAT\n");
                  fprintf(out1,"\n\nfactor: CONST_FLOAT\n");
                        fprintf(out,"%f\n",$1->v.f);
                        fprintf(out1,"%f\n",$1->v.f);
                  }
      | factor INCOP {fprintf(out,"\n\nfactor: factor INCOP\n");
                  fprintf(out1,"\n\nfactor: factor INCOP\n");
                        if($1->sz==0)
                        {
                              if($1->dataType == 0) {$1->v.i++; $$=$1;}
                              else if($1->dataType == 1) {$1->v.f++;
$$=$1;}

                        }
                        else
                        {
                              if($1->dataType==0){$1->v.i_Arr[arr_ind-
>v.i]++; $$=$1;}
                              else if($1->dataType==1){$1-
>v.f_Arr[arr_ind->v.i]++; $$=$1;}

                        }
                  }
      | factor DECOP {fprintf(out,"\n\nfactor: factor DECOP\n");
                  fprintf(out1,"\n\nfactor: factor DECOP\n");
                        if($1->sz==0)
                        {
                              if($1->dataType == 0) {$1->v.i--; $$=$1;}
                              else if($1->dataType == 1) {$1->v.f--;
$$=$1;}

                        }
                        else
                        {
                              if($1->dataType==0){$1->v.i_Arr[arr_ind-
>v.i]--; $$=$1;}
                              else if($1->dataType==1){$1-
>v.f_Arr[arr_ind->v.i]--; $$=$1;}

                        }
                  }
      ;
```

```
%%
int main(int argc,char *argv[])
{

      extern FILE *yyin;
      if(argc!=2){
            printf("Please provide input file name and try again\n");
            return 0;
      }

      FILE *fin=fopen(argv[1],"r");
      if(fin==NULL){
            printf("Cannot open specified file\n");
            return 0;
      }

      out= fopen("log.txt","w");
      out1=fopen("parser.txt","w");

      yyin= fin;
      yyparse();
      fprintf(out,"\n*******global Symbol table******\n");
      table1->print(out);
      fprintf(out,"\n*******local symbol table******\n");
      table2->print(out);
      fprintf(out1,"\n*******global Symbol table******\n");
      table1->print(out1);
      fprintf(out1,"\n*******local symbol table******\n");
      table2->print(out1);
      fprintf(out,"\nTotal Lines: %d\n",line_count);
      fprintf(out,"\nTotal Errors: %d\n",error_count);
      fclose(yyin);
      fclose(out);
      return 0;
}
```