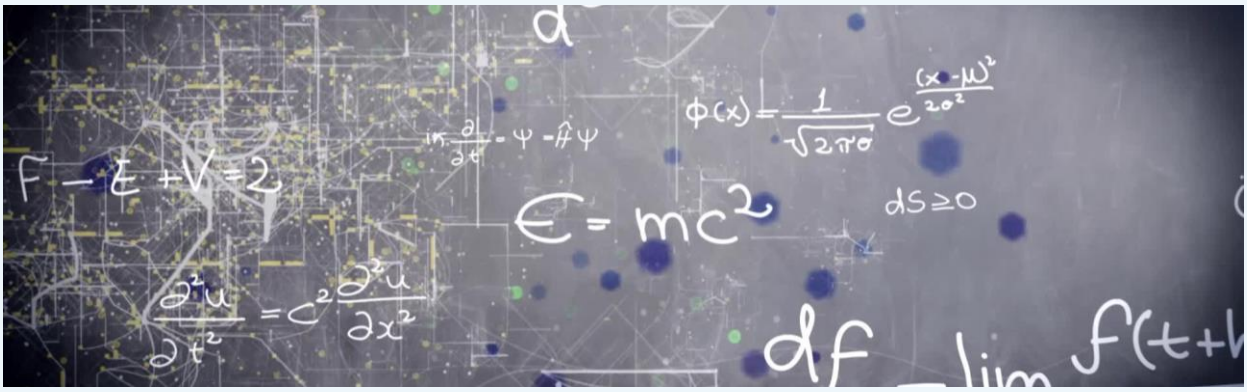

20BCE057
Practical 7
Aim : Implement Dangling Else

7.y file

```
%{

/* Definition Section*/

#include<stdio.h>

#include <stdlib.h>

%}


%token VAR NUMBER IF THEN ELSE LE GE NE EQ AND OR

%right '='
```

```
%left '<' '>' LE GE NE EQ AND OR

%left '+' '-' '*' '/' '%' '!'

%right UMINUS

%%


S:STMT {printf("Correct input\n");exit(0);};

STMT:IF '(' EXPR2 ')' THEN STMT1';' ELSE STMT1';'|IF '(' EXPR2 ')' THEN STMT1';'|STMT1;

STMT1:STMT|E;

E:VAR'='E|E'+'E|E'-'E|E'*'E|E'/'E|E'<'E|E'>'E|E LE E|E GE E|E EQ E|E NE E|E OR E|E AND E|VAR|NUMBER;

EXPR2:E'<'E|E'>'E|E LE E|E GE E|E EQ E|E NE E|E OR E|E AND E|VAR|NUMBER;

%%


#include "lex.yy.c"

main()


{

printf("\nNow you can enter expression:\n");

yyparse();

}


void yyerror()

{

printf("Invalid EXPR\n\n");
```

}

```

Now the 7.l file

%{

%}

alpha [A-Za-z]

digit [0-9]

/*Rules section*/

%%

[ \t\n]

if return IF;

else return ELSE;

then return THEN;

{digit}+ return NUMBER;

{alpha}({alpha}|{digit})* return VAR;

"<=" return LE;

">=" return GE;

"==" return EQ;
```

"!=" return NE;

"&&" return AND;

"||" return OR;

. return yytext[0];

%%

int yywrap(){return 1;}

```
(base) nirma@nirma-50:~/Desktop/7$ yacc 7.y
7.y: warning: 2 shift/reduce conflicts [-Wconflicts-sr]
7.y: warning: 1 reduce/reduce conflict [-Wconflicts-rr]
7.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples
(base) nirma@nirma-50:~/Desktop/7$ gcc y.tab.c -lfl -ly
y.tab.c: In function ‘yyparse’:
y.tab.c:1108:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-function-declaratio
 1108 |       yychar = yylex ();
      |                ^~~~~
y.tab.c:1249:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wim
 1249 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
      |       yyerrok
7.y: At top level:
7.y:21:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   21 | main()
      | ^~~~
7.y:26:6: warning: conflicting types for ‘yyerror’; have ‘void()’
   26 | void yyerror()
      |      ^~~~~~~
y.tab.c:1249:7: note: previous implicit declaration of ‘yyerror’ with type ‘void()’
 1249 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
(base) nirma@nirma-50:~/Desktop/7$ ./a.out < input2.txt

Enter expr:
Invalid EXPR

(base) nirma@nirma-50:~/Desktop/7$ ./a.out < input.txt

Enter expr:
Correct input
(base) nirma@nirma-50:~/Desktop/7$ ./a.out < input2.txt
```

Input.txt:

if (x > 5) then

y = x * 2;

else

y = x + 2;

Input2.txt:

if (x > 5) then if(z < -1) then y = x + 2;

else

y = x + 2;

7.l × 7.y × input.txt × in

```
%{
/* Definition Section*/
#include<stdio.h>
#include <stdlib.h>
%}
%token VAR NUMBER IF THEN ELSE LE GE NE EQ AND OR
%right '='
%left '<' '>' LE GE NE EQ AND OR
%left '+' '-' '*' '/' '%' '!'
%right UMINUS
%%

S:STMT {printf("Correct input\n");exit(0);};
STMT:IF '(' EXPR2 ')' THEN STMT1';' ELSE STMT1';'|IF '(' EXPR2 ')' THEN STMT1';'|STMT1;
STMT1:STMT|E;
E:VAR'='E|E'+'E|E'-'E|E'*'E|E'/'E|E'<'E|E'>'E|E LE E|E GE E|E EQ E|E NE E|E OR E|E AND E|VAR|NUMBER;
EXPR2:E'<'E|E'>'E|E LE E|E GE E|E EQ E|E NE E|E OR E|E AND E|VAR|NUMBER;
%%
#include "lex.yy.c"

main()
{
printf("\nNow you can enter expression:\n");
yyparse();
}
void yyerror()
{
printf("Invalid EXPR\n\n");
}
```

7.l × 7.y

```
%{

%}

alpha [A-Za-z]
digit [0-9]
/*Rules section*/
%%
[ \t\n]
if return IF;
else return ELSE;
then return THEN;
{digit}+ return NUMBER;
{alpha}({alpha}|{digit})* return VAR;
"<=" return LE;
">=" return GE;
"==" return EQ;
"!=" return NE;
"&&" return AND;
"||" return OR;
. return yytext[0];
%%

int yywrap(){return 1;}
```