Name: Rachit Shah
Roll No: 20BCE266
Course Name and Course Code: 2CS701 Compiler Construction
Practical No: 7
Aim: To implement grammar rules for control statements, and Loop control.

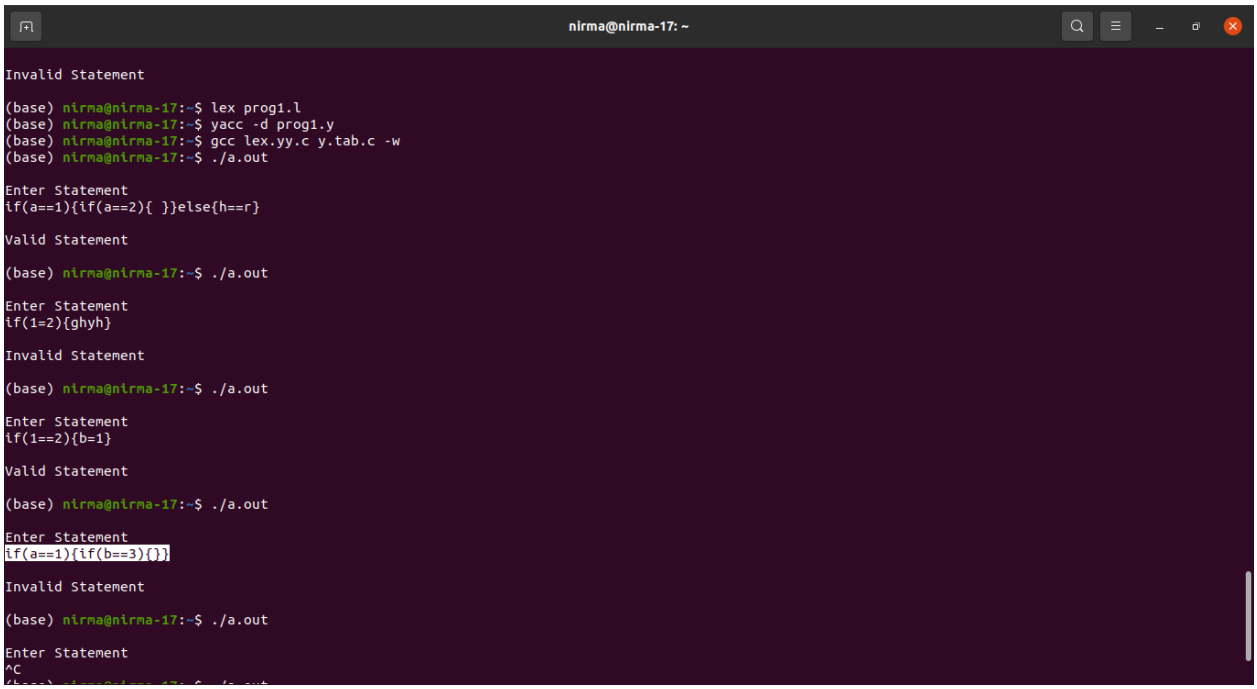
Code:
prac7.l file

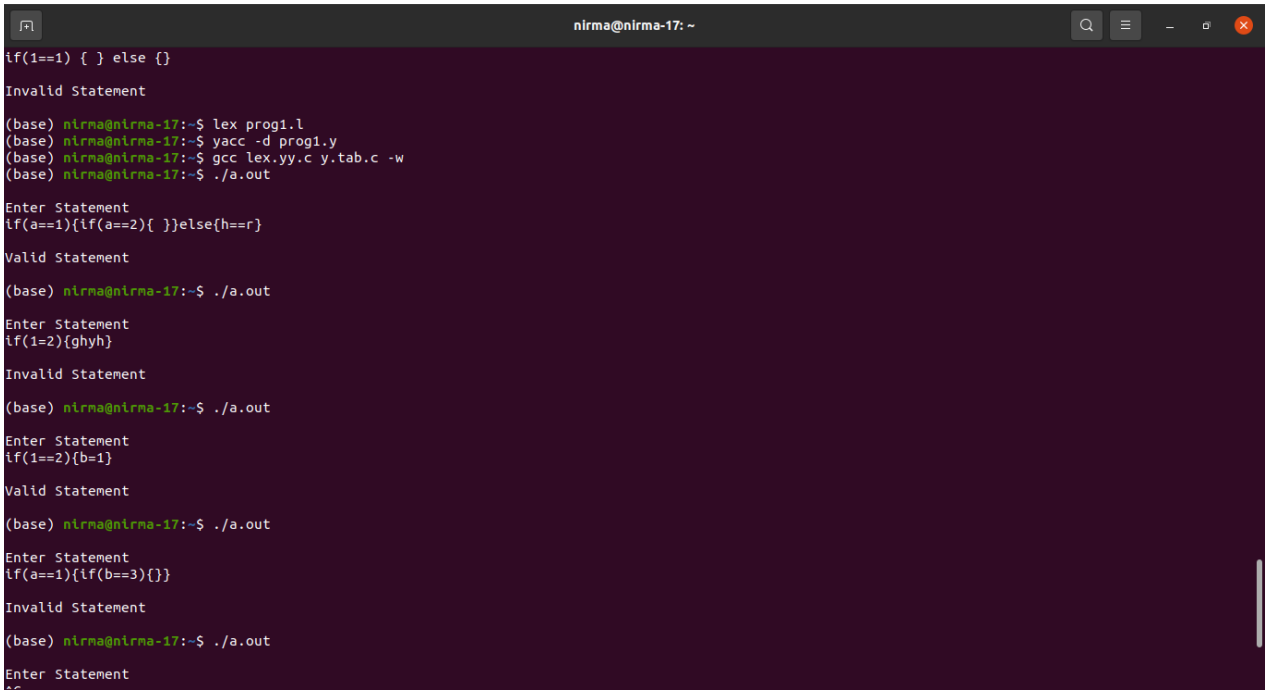
```
%{
  #include<stdio.h>
  #include "y.tab.h"
%}

%%
"if" {return IF;}
"else" {return ELSE;}
"<"|">"|"<="|">="|"=="|"="|"&&"|"||"|"and"|"or" {return OPR;}
[0-9]+ {return NUMBER;}
[A-Za-z][A-Za-z0-9_]* {return ID;}
" " {return S;}

[\t] ;
[\n] return 0;
. return yytext[0];

%%

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

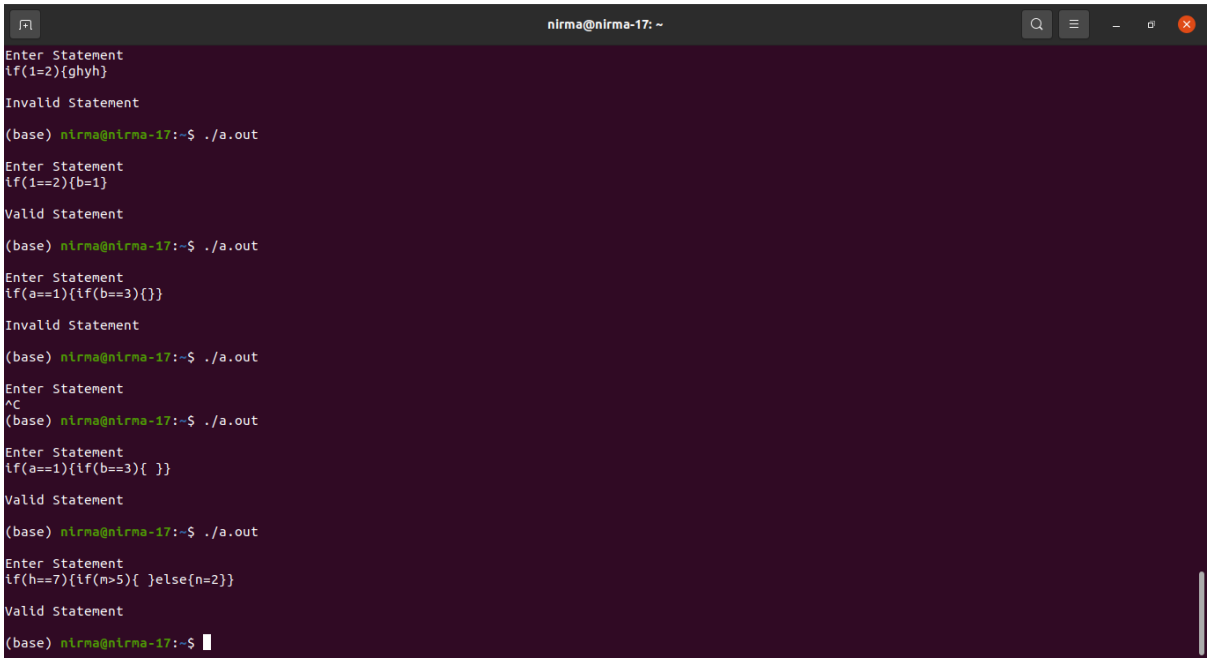
Prac7.y

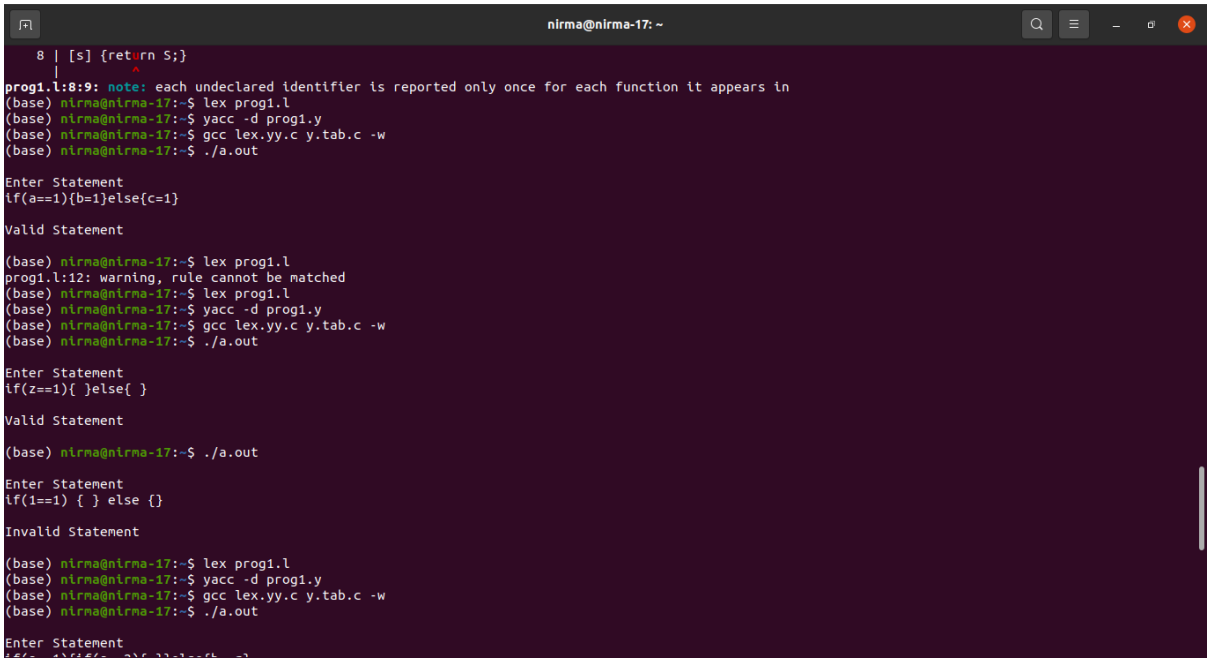
```
%{
  #include<stdio.h>
  int flag=0;
%}

%token IF OPR NUMBER ID ELSE S

%%
IfExpression: E
  {
        //printf("\nResult=%d\n", $$);
        return 0;
  };

 E: IF '('C')"{'C'}' | IF '('C')"{'C'}' ELSE '{'C'}';
 C: T OPR T | S |E;
 T: NUMBER
 | ID;
%%

void main()
{
   printf("\nEnter Statement\n");
   yyparse();
   if(flag==0)
   printf("\nValid Statement\n\n");
}

void yyerror()
{
   printf("\nInvalid Statement\n\n");
   flag=1;
}
```

Output:


nirma@nirma-17: ~
Invalid Statement
(base) nirma@nirma-17:~$ lex prog1.l
(base) nirma@nirma-17:~$ yacc -d prog1.y
(base) nirma@nirma-17:~$ gcc lex.yy.c y.tab.c -w
(base) nirma@nirma-17:~$ ./a.out
Enter Statement
if(a==1){if(a==2){ }}else{h==r}
Valid Statement
(base) nirma@nirma-17:~$ ./a.out
Enter Statement
if(1=2){ghyh}
Invalid Statement
(base) nirma@nirma-17:~$ ./a.out
Enter Statement
if(1==2){b=1}
Valid Statement
(base) nirma@nirma-17:~$ ./a.out
Enter Statement
if(a==1){if(b==3){}}
Invalid Statement
(base) nirma@nirma-17:~$ ./a.out
Enter Statement
^C


nirma@nirma-17: ~
if(1==1) { } else {}
Invalid Statement
(base) nirma@nirma-17:~$ lex prog1.l
(base) nirma@nirma-17:~$ yacc -d prog1.y
(base) nirma@nirma-17:~$ gcc lex.yy.c y.tab.c -w
(base) nirma@nirma-17:~$ ./a.out
Enter Statement
if(a==1){if(a==2){ }}else{h==r}
Valid Statement
(base) nirma@nirma-17:~$ ./a.out
Enter Statement
if(1=2){ghyh}
Invalid Statement
(base) nirma@nirma-17:~$ ./a.out
Enter Statement
if(1==2){b=1}
Valid Statement
(base) nirma@nirma-17:~$ ./a.out
Enter Statement
if(a==1){if(b==3){}}
Invalid Statement
(base) nirma@nirma-17:~$ ./a.out
Enter Statement


nirma@nirma-17: ~
Enter Statement
if(1=2){ghyh}
Invalid Statement
(base) nirma@nirma-17:~$ ./a.out
Enter Statement
if(1==2){b=1}
Valid Statement
(base) nirma@nirma-17:~$ ./a.out
Enter Statement
if(a==1){if(b==3){}}
Invalid Statement
(base) nirma@nirma-17:~$ ./a.out
Enter Statement
^C
(base) nirma@nirma-17:~$ ./a.out
Enter Statement
if(a==1){if(b==3){ }}
Valid Statement
(base) nirma@nirma-17:~$ ./a.out
Enter Statement
if(h==7){if(m>5){ }else{n=2}}
Valid Statement
(base) nirma@nirma-17:~$


nirma@nirma-17: ~
8 | [s] {return S;}
| ^
prog1.l:8:9: note: each undeclared identifier is reported only once for each function it appears in
(base) nirma@nirma-17:~$ lex prog1.l
(base) nirma@nirma-17:~$ yacc -d prog1.y
(base) nirma@nirma-17:~$ gcc lex.yy.c y.tab.c -w
(base) nirma@nirma-17:~$ ./a.out
Enter Statement
if(a==1){b=1}else{c=1}
Valid Statement
(base) nirma@nirma-17:~$ lex prog1.l
prog1.l:12: warning, rule cannot be matched
(base) nirma@nirma-17:~$ lex prog1.l
(base) nirma@nirma-17:~$ yacc -d prog1.y
(base) nirma@nirma-17:~$ gcc lex.yy.c y.tab.c -w
(base) nirma@nirma-17:~$ ./a.out
Enter Statement
if(z==1){ }else{ }
Valid Statement
(base) nirma@nirma-17:~$ ./a.out
Enter Statement
if(1==1) { } else {}
Invalid Statement
(base) nirma@nirma-17:~$ lex prog1.l
(base) nirma@nirma-17:~$ yacc -d prog1.y
(base) nirma@nirma-17:~$ gcc lex.yy.c y.tab.c -w
(base) nirma@nirma-17:~$ ./a.out
Enter Statement