NAME: VAIBHAV BANKA

REG NO: 21BCE1955

1. Check whether the given expression is a valid relational expression or not.

LEX CODE

```
%{
 #include "y.tab.h"
%}

%%
[a-zA-Z][a-zA-Z0-9]* {return chr;}
[0-9] {return num;}
\n {return 0;}
[\t]+ ;
.  {return yytext[0];}
%%

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

YACC PROGRAM

```
%{
  #include<stdio.h>
  #include<stdlib.h>
%}
%token chr num
%%
stmt: S   { printf("valid string\n");
            return 0; }
```

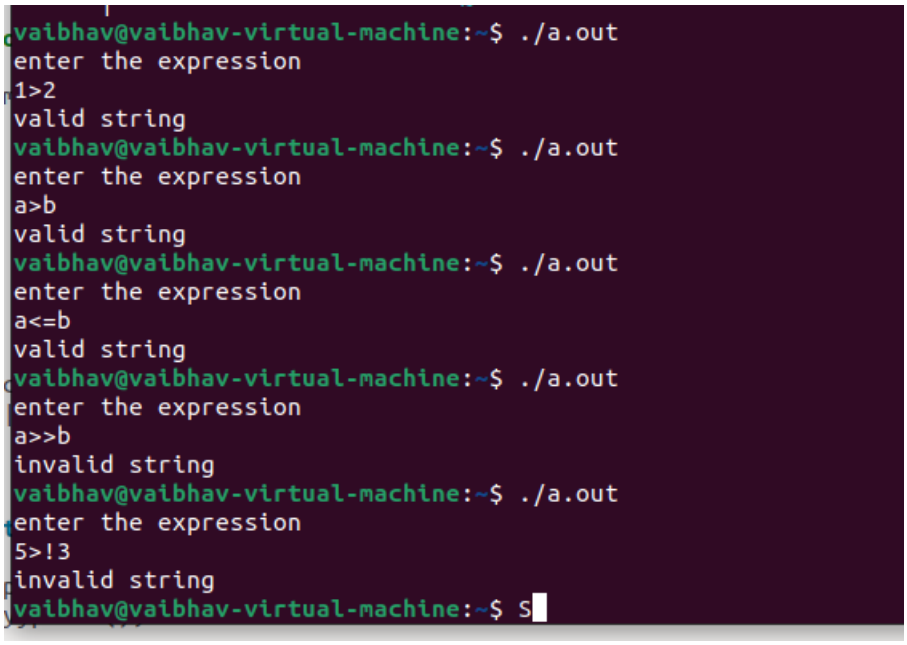
```
;
S:  T'>'T
    |T'<'T
    |T'>''='T
    |T'<''='T
    |T'!''='T
    |T'=''='T
    |T'='T
;
T:chr
  |num
;

%%

int yyerror(char *msg)
 {
  printf("invalid string\n");
  exit(0);
 }

main()
 {
  printf("enter the string\n");
  yyparse();
 }
```

OUTPUT


vaibhav@vaibhav-virtual-machine:~$ ./a.out
enter the expression
1>2
valid string
vaibhav@vaibhav-virtual-machine:~$ ./a.out
enter the expression
a>b
valid string
vaibhav@vaibhav-virtual-machine:~$ ./a.out
enter the expression
a<=b
valid string
vaibhav@vaibhav-virtual-machine:~$ ./a.out
enter the expression
a>>b
invalid string
vaibhav@vaibhav-virtual-machine:~$ ./a.out
enter the expression
5>!3
invalid string
vaibhav@vaibhav-virtual-machine:~$ S

2. **Write lex code and yacc code to check if the relational operators are correct for ∑= {0-9} * and print the correct relation if the given input is incorrect The output should be like this, For , input : 7>=5 , Output : Yes, 7 is >= 5**
For , input : 5>=7 , Output : No, 5 is not >= 7

LEX CODE

```
%{
  #include "y.tab.h"
  extern int yylval;
%}
%%
[0-9]+ {yylval=atoi(yytext);return NUM;}
">=" {return GE;}
"<=" {return LE;}
"!=" {return NE;}
">" {return GT;}
"<" {return LT;}
"==" {return EE;}
\n {return 0;}
. {return yytext[0];}
%%
int yywrap(){}
```

YACC PROGRAM

```
%{
 #include <stdio.h>
 int yylex(void);
 int yyerror(char* s);
%}
%token NUM GE LE EE GT LT NE
%left GE LE EE GT LT NE
%%
T:
```

```
 NUM GE NUM {if($1>=$3) {
 printf("Yes, %d is >= %d\n",$1,$3);}
 else {printf("No, %d is not greater than equal %d\n",$1,$3);}}
 |NUM LE NUM {if($1<=$3) {
 printf("Yes, %d is <= %d\n",$1,$3);}
 else {printf("No, %d is not greater than equal to %d\n",$1,$3);}}
 |NUM NE NUM {if($1!=$3) {
 printf("Yes, %d is != %d\n",$1,$3);}
 else {printf("No, %d is not equal to %d\n",$1,$3);}}
 |NUM EE NUM {if($1==$3) {
 printf("Yes, %d is == %d\n",$1,$3);}
 else {printf("No, %d is not same as %d\n",$1,$3);}}
 |NUM GT NUM {if($1>$3) {
 printf("Yes, %d is greater than %d\n",$1,$3);}
 else {printf("No, %d is not greater %d\n",$1,$3);}}
 |NUM LT NUM {if($1<$3) {
 printf("Yes, %d is < %d\n",$1,$3);}
 else {printf("No, %d is not lesser than %d\n",$1,$3);}}
;
%%
int main()
{
 printf("Enter the expression:\n");
 yyparse();
 return 0;
}
int yyerror(char* s)
{
 printf("\nThe expression is invalid\n");
}
```

OUTPUT

```
vaibhav@vaibhav-virtual-machine:~$ lex ex5q2.l
vaibhav@vaibhav-virtual-machine:~$ yacc -d ex5q2.y
ex5q2.y:29 parser name defined to default :"parse"
vaibhav@vaibhav-virtual-machine:~$ cc lex.yy.c y.tab.c
vaibhav@vaibhav-virtual-machine:~$ ./a.out
Enter the expression:
5>2
Yes, 5 is greater than 2
vaibhav@vaibhav-virtual-machine:~$ ./a.out
Enter the expression:
2>5
No, 2 is not greater 5
```

3. Make a program to check whether a given expression is valid for loop statement or not.

LEX CODE

```
%{
  #include "y.tab.h"
%}
alpha [A-Za-z]
digit [0-9]
%%
[\t \n]
"for" return FOR;
{digit}+ return NUM;
{alpha}({alpha}|{digit})* return ID;
"<=" return LE;
">=" return GE;
"==" return EQ;
"!=" return NE;
"||" return OR;
"&&" return AND;
. return yytext[0];
%%
int yywrap(){}
```

YACC PROGRAM

```
%{
  #include <stdio.h>
  #include <stdlib.h>
  int yylex(void);
  int yyerror(char *s);
%}
%token ID NUM FOR LE GE EQ NE OR AND
%right '='
%left OR AND
```

```
%left '>' '<' LE GE EQ NE
%left '+' '-'
%left '*' '/'
%right UMINUS
%left '!'
%%
S: ST {printf("Input accepted\n");return 0;}
ST : FOR '(' E ';' E2 ';' E ')' DEF
;
DEF: '{' BODY '}'
  | E';'
  | ST
  |
;
BODY: BODY BODY
  | E ';'
  | ST
  |
;
E: ID '=' 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
  | E '+' '+'
  | E '-' '-'
  | ID
  | NUM
;
E2: E'<'E
  | E'>'E
  | E LE E
  | E GE E
  | E EQ E
  | E NE E
  | E OR E
  | E AND E
;
%%
int main() {
  printf("Enter the expression:\n");
  yyparse();
  return 0;
}
int yyerror(char *s)
{
  printf("invalid\n");
}
```

OUTPUT

```
vaibhav@vaibhav-virtual-machine:~$ lex ex5q3.l
vaibhav@vaibhav-virtual-machine:~$ yacc -d ex5q3.y
ex5q3.y:56 parser name defined to default :"parse"
conflicts:  13 shift/reduce, 4 reduce/reduce
vaibhav@vaibhav-virtual-machine:~$ cc lex.yy.c y.tab.c
vaibhav@vaibhav-virtual-machine:~$ ./a.out
Enter the expression:
for(int i=0;i<10;i++){a=a*25}
invalid
vaibhav@vaibhav-virtual-machine:~$ ./a.out
Enter the expression:
for(int i=0;i<10;i++){a=a+25;}
invalid
vaibhav@vaibhav-virtual-machine:~$ ./a.out
Enter the expression:
for(i=0;i<10;i++){a=a*25;}
Input accepted
vaibhav@vaibhav-virtual-machine:~$ S
```