1 Boolean expression:

Lex

```
%option noyywrap
%{
        #include<stdio.h>
        #include "y.tab.h"
        extern int yylval;
%}
%%
[0-9]+ {
        yylval=atoi(yytext);
        return NUMBER;
}
"&&" {
        return AND;
}
"||" {
        return OR;
}
[\t] ;
[\n] return 0;
. return yytext[0];
%%
```

Yacc

```
%{
#include<stdio.h>
int yylex();
int flag=0;
int yyerror();
%}
%token NUMBER AND OR
%left '!' AND OR
%%
booleanExpression: E{
printf("Result: %d\n", $$);
return 0;
};
E :E AND E {$$=$1&&$3;}
|E OR E {$$=$1||$3;}
|'!'E {$$=!$2;}
|NUMBER {$$=$1;}
;
%%
void main()
{
printf("Enter the Boolean Expression:");

yyparse();
if(flag==0)
printf("\nValid Boolean Expression\n");
}
int yyerror()
{
printf("\nInvalid Boolean Expression\n");
flag=1;
return 1;
}
```

Output:

```
C:\Flex Windows\EditPlusPortable>9_1.exe
Enter the Boolean Expression:5&&6
Result: 1

Valid Boolean Expression

C:\Flex Windows\EditPlusPortable>
```

2 Prefix expression

Lex

```
%{
        #include"y.tab.h"
%}
%%
[0-9] {
        yylval=atoi(yytext);
        return NUMBER;
}
[t];
\n return 0;
. {
        return yytext[0];
}
%%
int yywrap(){
        return 1;
}
```

Yacc

```
%{
        #include<stdio.h>
        int yyerror(char *s);
        int yylex();
        int c=0;
%}
%token NUMBER
%left '+' '-'
%left '*' '/'
%nonassoc UMINUS
%%
state : exp { c++;}
;
exp : NUMBER

| '+' exp exp {}
| '-' exp exp {}
| '*' exp exp {}
| '/' exp exp {}
;
%%
int yyerror(char * str){
        printf("\nInvalid Expression");
        c=0;
        return 1;
}
int main(){
        printf("\nEnter the Expression : ");
        yyparse();
        if(c>0)
        printf("THIS IS A VALID PREFIX EXPRESSION\n");
        return(0);
}
```

Output:

```
C:\Flex Windows\EditPlusPortable>9_2.exe

Enter the Expression : *5/4

Invalid Expression
C:\Flex Windows\EditPlusPortable>9_2.exe

Enter the Expression : */23-20
THIS IS A VALID PREFIX EXPRESSION

C:\Flex Windows\EditPlusPortable>
```

3 Postfix expression

Yacc

```
%{
        #include<stdio.h>
        int yylex();
        int yyerror(char *str);
        int c=0;
%}
%token NUMBER
%left '+' '-'
%left '*' '/'
%nonassoc UMINUS
%%
state : exp { c++; }
;
exp : NUMBER
| exp exp '+' {}
| exp exp '-' {}
| exp exp '*' {}
| exp exp '/' {}
;
%%
int yyerror(char * str){
        printf("\nInvalid Expression");
        c=0;
        return 1;
}

int main(){
        printf("\nEnter the Expression : ");
        yyparse();
        if(c>0)
        printf("THIS IS A VALID POSTFIX EXPRESSION\n");
        return(0);
}
```

Output:

```
C:\Flex Windows\EditPlusPortable>9_3.exe

Enter the Expression : 5+6-

Invalid Expression
C:\Flex Windows\EditPlusPortable>9_3.exe

Enter the Expression : 21-45/+
THIS IS A VALID POSTFIX EXPRESSION
```

4 Validate For Loop

Lex

```
%option noyywrap
%{
        #include<stdio.h>
        #include "y.tab.h"
%}
        alpha [A-Za-z]
        digit [0-9]
%%
[\t \n]
printf[^;]* return PRINTF;
"int"|"float"|"char" return DTYPE;
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];
%%
```

Yacc

```
%{
        #include <stdio.h>
        #include <stdlib.h>
        int yylex();
        int yyerror();
%}

%token ID NUM FOR LE GE EQ NE OR AND DTYPE PRINTF
%right "="
%left OR AND
%left '>' '<' LE GE EQ NE DTYPE PRINTF
%left '+' '-'
%left '*' '/'
%right UMINUS
%right '!'
%%
S : ST {printf("For construct is valid\n"); return 0;}
ST : FOR '('DTYPE E ';' E2 ';' E ')' DEF
;
DEF : '{' BODY '}'
| E';'
| ;
BODY : BODY BODY
| E1 ';'
|
;
E : ID '=' E
| E '+' E
| E '-' E
| E '*' E
| E '/' E
| E '+' '+'
| E '-' '-'
| ID
| NUM
;
```

```
E1 : ID '=' E3
| PRINTF
;
E3 : E3 '+' E3
| E3 '-' E3
| E3 '*' E3
| E3 '/' E3
| E3 '+' '+'
| E3 '-' '-'
| E3 LE E3
| E3 GE E3
| E3 EQ E3
| E3 NE E3
| E3 OR E3
| E3 AND E3
| 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(){
        printf("For construct is invalid\n");
}
```

Output:

```
C:\Flex Windows\EditPlusPortable>9_4.exe
Enter the expression:
for(int i=0; i<2; i++)
{
       printf("%d\n",i);
}
For construct is valid
```