Lab Assignment 7

U21CS089 | Garvit Shah

1. Write a YACC and LEX program to implement a Calculator and recognize a valid arithmetic expression that uses operator +, -, *, /.

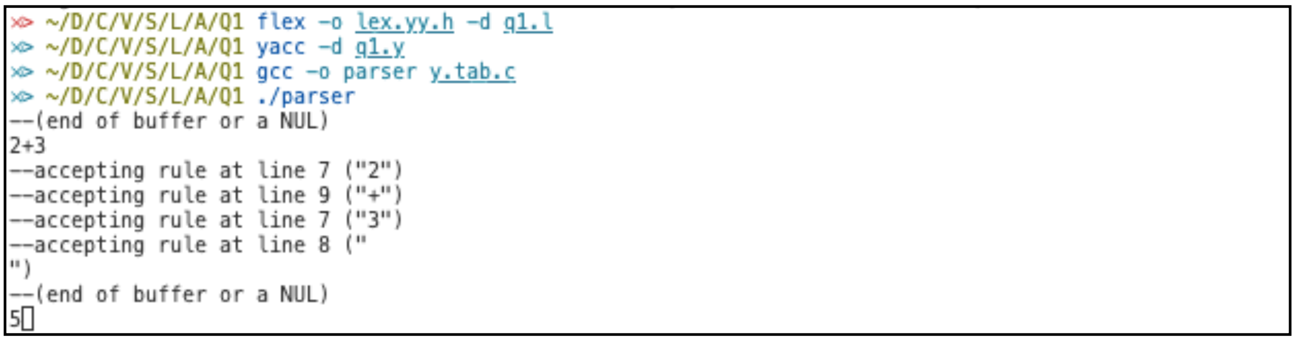
LEX File

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

%%

[0-9]+       {yylval.num = atoi(yytext); return NUM;}
[ \t\n]      ;
[-+*/;]      {return yytext[0];}
.            ;

%%

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

YACC File

```
%{
#include <stdlib.h>
#include <stdio.h>
void yyerror(char *s);
#include "lex.yy.h"

%}

%union {int num;}
%start S
%token NUM
%type <num> NUM

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


%%
S   :   NUM             {printf("%d", $1);}
    |   NUM '+' NUM     {printf("%d", $1 + $3);}
    |   NUM '-' NUM     {printf("%d", $1 - $3);}
    |   NUM '*' NUM     {printf("%d", $1 * $3);}
    |   NUM '/' NUM     {printf("%d", $1 / $3);}
    ;

%%

int main(){
    return yyparse();
}

void yyerror(char *s) {
    printf("\nExpression is invalid\n");
}

}
```

On Terminal:



```
⋟ ~/D/C/V/S/L/A/Q1 flex -o lex.yy.h -d q1.l
⋟ ~/D/C/V/S/L/A/Q1 yacc -d q1.y
⋟ ~/D/C/V/S/L/A/Q1 gcc -o parser y.tab.c
⋟ ~/D/C/V/S/L/A/Q1 ./parser
--(end of buffer or a NUL)
2+3
--accepting rule at line 7 ("2")
--accepting rule at line 9 ("+")
--accepting rule at line 7 ("3")
--accepting rule at line 8 ("
")
--(end of buffer or a NUL)
5
```

2. Write a YACC and LEX program to check whether given string is palindrome or not.

LEX File

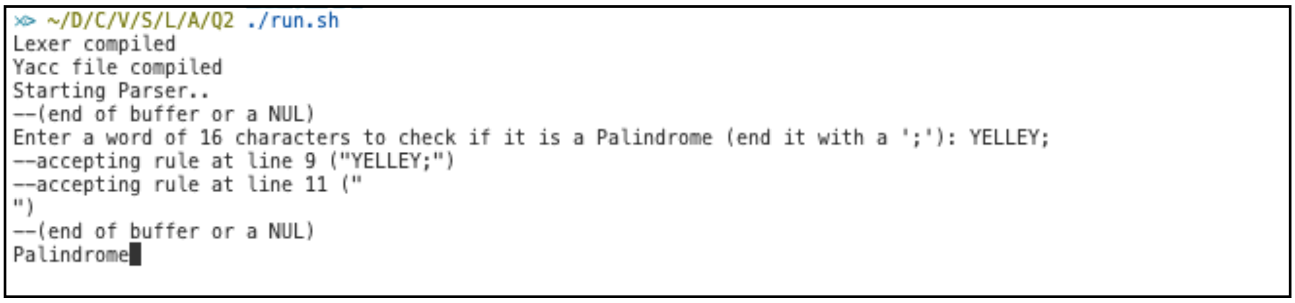
```
%{
#include "y.tab.h"
%}

LETTER [a-zA-Z]
SEMI [;]
%%

{LETTER}+{SEMI}     {yylval.str = yytext; ;return L;}

[ \t\n]     ;
.           ;

%%

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

YACC File

```
%{
#include <stdlib.h>
#include <stdio.h>
#include "lex.yy.h"

void yyerror(char *s);
int symbtab[26];
char word[16];
int x = 0;
%}

%union {char* str;}
%start S
%token L
%type <str> L S

%%
S   :   L       {
                    $$ = $1;
                    int i =0, flag = 0;
                    for(i =0; $$[i]!=';'; i++){
                        word[i] = $$[i];
                    }
                    for(int j = 0; j < i/2; j++){
                        if(word[j]!=word[i-1-j]){
                            printf("Not Palindrome!");
                            flag = 1;
                        }
                    }
                    if(flag == 0){
                        printf("Palindrome");
                    }
                }
    ;

%%

int main(){
    printf("Enter a word of 16 characters to check if it is a Palindrome (end it
with a ';'): ");
    return yyparse();
}

void yyerror(char *s) {
    printf("\nExpression is invalid\n");
}
```

SHELL FIle

```
flex q1.l
flex -o lex.yy.h -d q1.l
echo "Lexer compiled"
yacc -d q1.y
echo "Yacc file compiled"
gcc -o parser y.tab.c
echo "Starting Parser.."
./parser
```



```
≫ ~/D/C/V/S/L/A/Q2 ./run.sh
Lexer compiled
Yacc file compiled
Starting Parser..
--(end of buffer or a NUL)
Enter a word of 16 characters to check if it is a Palindrome (end it with a ';'): YELLEY;
--accepting rule at line 9 ("YELLEY;")
--accepting rule at line 11 ("
")
--(end of buffer or a NUL)
Palindrome
```

3. Write a program for implementing given grammar for computing the expression using semantic rules of the YACC tool and LEX. Grammar: S-> SS* | SS+ | a

LEX File

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


SYMB [+*]

%%

^a$                                    {return STRING;}
^a[a\+\*]*                   {return STRING;}
[ \t\n]                                             ;
.                            {return 0;}

%%

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

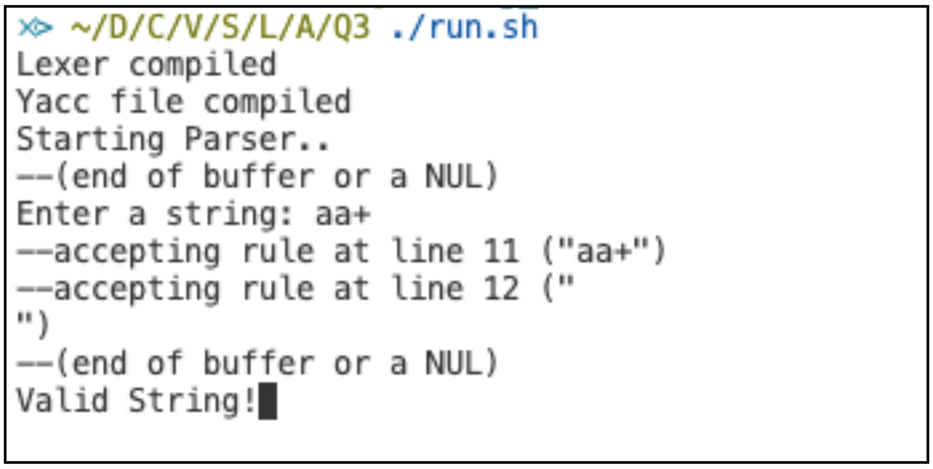
YACC File

```
%{
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "lex.yy.h"

void yyerror(char *s);
%}


%start S
%token STRING

%%
S   :   STRING      {printf("Valid String!");}
    ;

%%

int main(){
    printf("Enter a string: ");
    return yyparse();
}

void yyerror(char *s) {
    printf("\nExpression is invalid\n");
}
```

SHELL FIle

```
flex q3.l
flex -o lex.yy.h -d q3.l
echo "Lexer compiled"
yacc -d q3.y
echo "Yacc file compiled"
gcc -o parser y.tab.c
echo "Starting Parser.."
```

```
./parser
```



```
≫ ~/D/C/V/S/L/A/Q3 ./run.sh
Lexer compiled
Yacc file compiled
Starting Parser..
--(end of buffer or a NUL)
Enter a string: aa+
--accepting rule at line 11 ("aa+")
--accepting rule at line 12 ("
")
--(end of buffer or a NUL)
Valid String!
```

4. Write a YACC and LEX program to accept strings that starts and ends with 0 or 1.

LEX File

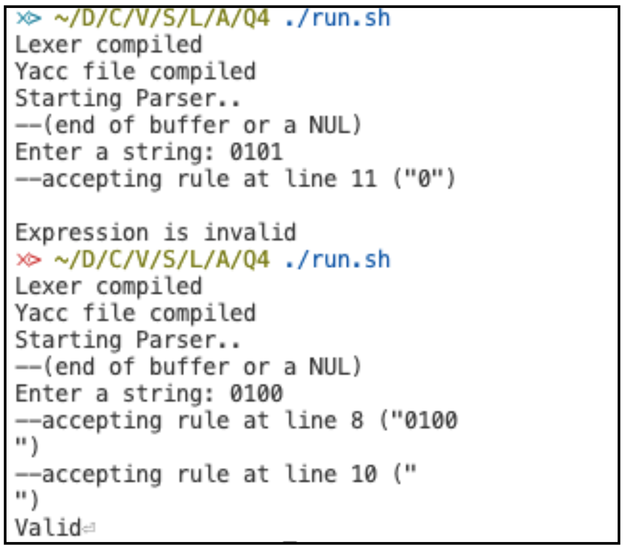
```
%{
#include "y.tab.h"
%}

TERM [01]
%%

^0[01]*0$        { return STRING; }
^1[01]*1$           { return STRING; }
[ \t\n]            {return 0;}
.                  {return 0;}

%%

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

YACC File

```
%{
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "lex.yy.h"

void yyerror(char *s);
%}

%start S
%token STRING

%%
S   :   STRING      {printf("Valid");}
    ;

%%

int main(){
    printf("Enter a string: ");
    return yyparse();
}

void yyerror(char *s) {
    printf("\nExpression is invalid\n");
}
```

SHELL File

```
flex q4.l
flex -o lex.yy.h -d q4.l
echo "Lexer compiled"
yacc -d q4.y
echo "Yacc file compiled"
gcc -o parser y.tab.c
echo "Starting Parser.."
./parser
```



```
⋙ ~/D/C/V/S/L/A/Q4 ./run.sh
Lexer compiled
Yacc file compiled
Starting Parser..
--(end of buffer or a NUL)
Enter a string: 0101
--accepting rule at line 11 ("0")

Expression is invalid
⋙ ~/D/C/V/S/L/A/Q4 ./run.sh
Lexer compiled
Yacc file compiled
Starting Parser..
--(end of buffer or a NUL)
Enter a string: 0100
--accepting rule at line 8 ("0100
")
--accepting rule at line 10 ("
")
Valid
```