Project Report
On

Compiler for Layman Friendly Calculator

Prepared by:

Heli Vachhani - IT049 - 18ITUON113

Dhruvi Jivani - IT051 - 18ITUOS046

Mitsu Kansagara - IT054 - 18ITUON118

Guided by:

Prof. Nikita P. Desai
Department of Information Technology,
Faculty of Technology, DD University

Department of Information Technology,
Faculty of Technology, Dharmsinh Desai University
College Road, Nadiad-387001.

April-2021

CERTIFICATE

This is to certify that the project entitled “**Layman friendly calculator”** is a bonafide report of the work carried out by

1) Ms. Heli Vachhani , Student ID No: 18ITUON113
2) Ms. Dhruvi Jivani , Student ID No: 18ITUOS046
3) Ms. Mitsu Kansagara , Student ID No: 18ITUON118

of Department of Information Technology, semester VI, under the guidance and supervision for the award of the degree of Bachelor of Technology at Dharmsinh Desai University, Nadiad (Gujarat). They were involved in Project in subject of “**Language Translator**” during the academic year 2020-2021.

Prof. N.P. Desai
(Lab Incharge)
Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University, Nadiad
Date:

Prof. (Dr.)V K Dabhi,
Head , Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University,
Nadiad
Date:

Index

1.0 INTRODUCTION

1.0.1 Project Details

Project definition:
Following is a valid sentence in some layman friendly calculator program. Generate its appropriate language description and compiler.

ADD 23 WITH 444.45 . ALSO ADD 4543.78 TO 123.
NOW DO ADDITION OF BOTH ANSWERS.WHAT IS
RESULT?

A. In above language:

1) Operators : add , addition,sub,subtraction,div,division,mul,multiplication
2) Punctuations : “.” , “?”
3) Keywords : with , also , to, of, now , do , both , answers , what , is , result
4) Constant : Int constant , Float constant

B. The Regular expression is as follows:

Language L1(Operators) : “add” | ”addition” |”sub” | “subtraction” | “div” | “division” | “mul” | “multiplication”
Language L2(Keywords) : “with” | “also” | “to” |”of” | “now” | “do” | “both” | “answers” | “what” | “is” | “result”
Language L3(Constant) :
Int constant : [0-9]+
float constant : ([0-9]+[.])[0-9]+
Language L4(Punctuations) : “.” | “?”

1.0.2 Project Planning List of Students with their Roles/Responsibilities:

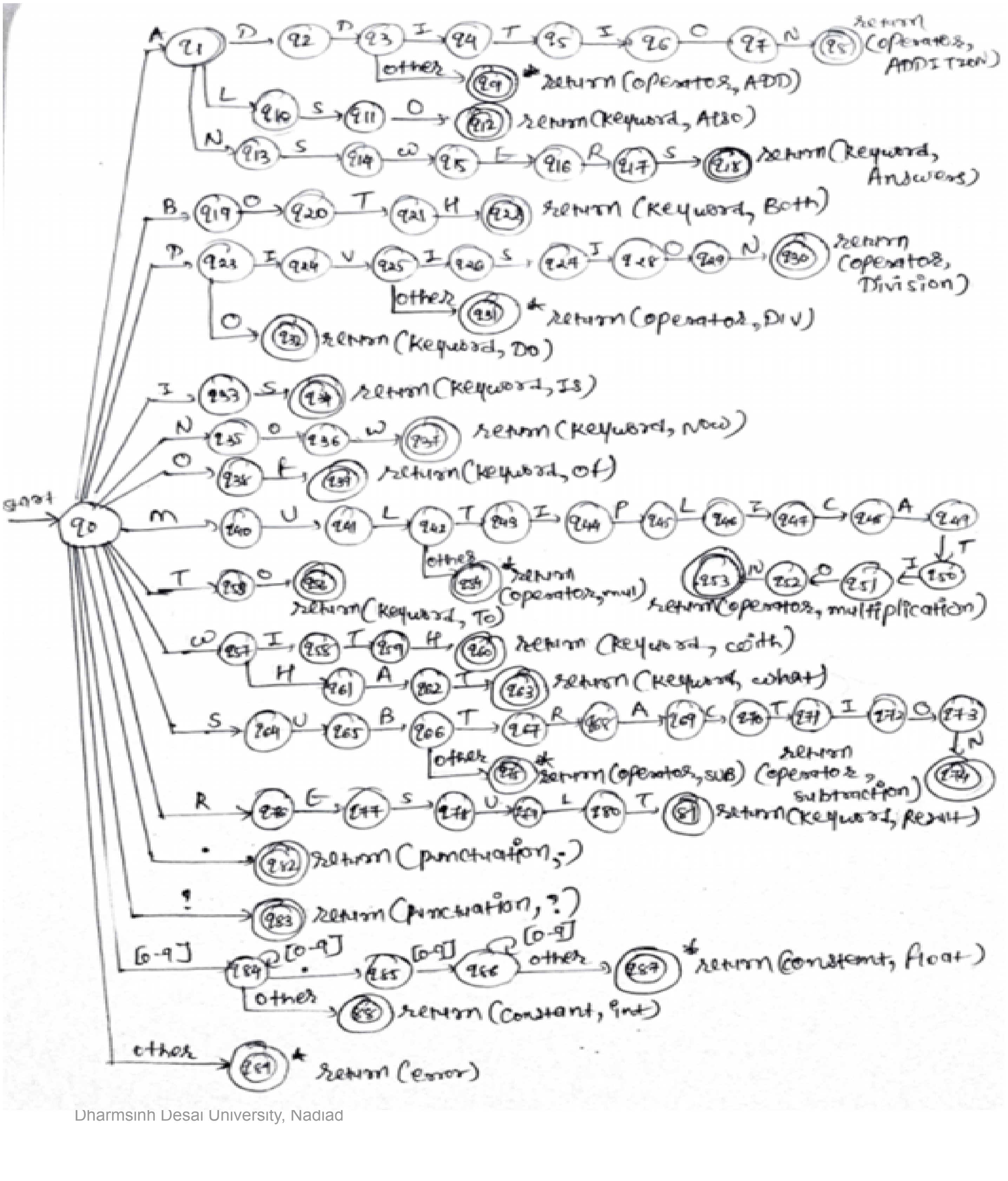
IT049 Heli Vachhani– Grammar Rules, YACC Implementation, Scanner Implementation, Final Code Verification, Documentation

IT051 Dhruvi Jivani– Algorithm Design, DFA Design, Scanner Implementation, YACC Implementation, Final Code Verification, Documentation

IT054 Mitsu Kansagara– DFA Design, Regular Expression, Documentation

2.0 LEXICAL PHASE DESIGN

2.0.1 Deterministic Finite Automaton design for lexer



2.0.2 Algorithm of lexer

```
Void lexer(){
        char ch;
        state:=0;
        ch=getchar();
        //eof symbol #
        While not # do
        case ‘0’:
                if(ch==’A’)
                    state:=1;
                    ch=getchar();
                else if(ch==’B’)
                    state:=19;
                    ch=getchar();
                else if(ch==’D’)
                    state:=23;
                    ch=getchar();
                else if(ch==’I’)
                    state:=33;
                    ch=getchar();
                else if(ch==’N’)
                    state:=35;
                    ch=getchar();
```

```
else if(ch==’O’)
        state:=38;
        ch=getchar();
else if(ch==’M’)
        state:=40;
        ch=getchar();
else if(ch==’T’)
        state:=55;
        ch=getchar();
else if(ch==’W’)
        state:=57;
        ch=getchar();
else if(ch==’S’)
        state:=64;
        ch=getchar();
else if(ch==’R’)
        state:=76;
        ch=getchar();
else if(ch==’.’)
        state:=82;
else if(ch==’?’)
        state:=83;
else if(ch==[0-9])
```

```
                    state:=84;
                    ch=getchar();
                else
                    state:=89;
                    exit while;
                end case;
        case ‘1’:
                if(ch==’D’)
                    state:=2;
                    ch=getchar();
                else if(ch==’L’)
                    state:=10;
                    ch=getchar();
                else if(ch==’N’)
                    state:=13;
                    ch=getchar();
                else
                    exit while;
                end case;

        case ‘2’:
                if(ch==’D’)
                    state:=3;
                    ch=getchar();
                end case;

        case ‘3’:
                if(ch==’I’)
                    state:=4;
                    ch=getchar();
                else
                    state:=9;
                end case;

        case ‘4’:
                if(ch==’T’)
                    state:=5;
                    ch=getchar();
                end case;

        case ‘5’:
                if(ch==’I’)
                    state:=6;
                    ch=getchar();
                end case;
```

```
case ‘6’:
        if(ch==’O’)
            state:=7;
            ch=getchar();
        end case;

case ‘7’:
        if(ch==’N’)
            state:=8;
            ch=getchar();
        end case;

case ‘8’:
        return(operator,ADDITION)
        exit while;
        end case;

case ‘9’:
        unput(ch);
        return(operator,ADD)
        exit while;
        end case;
case ‘10’:
```

```
                if(ch==’S’)
                        state:=11;
                        ch=getchar();
                end case;
        case ‘11’:
                if(ch==’O’)
                        state:=12;
                        ch=getchar();
                end case;
        case ‘12’:
                return(keyword,ALSO);
                exit while;
                end case;
        case ‘13’:
                if(ch==’S’)
                        state:=14;
                        ch=getchar();
                end case;
        case ‘14’:
                if(ch==’W’)
                        state:=15;
                        ch=getchar();
                end case;
```

```
case ‘15’:
            if(ch==’E’)
                state:=16;
                ch=getchar();
            end case;
case ‘16’:
            if(ch==’R’)
                state:=17;
                ch=getchar();
            end case;
case ‘17’:
            if(ch==’S’)
                state:=18;
                ch=getchar();
            end case;
case ‘18’:
            return(keyword,ANSWERS);
            exit while;
case ‘19’:
            if(ch==’O’)
                state:=20;
                ch=getchar();
            end case;
```

```
case ‘20’:
            if(ch==’T’)
                state:=21;
                ch=getchar();
            end case;
case ‘21’:
            if(ch==’H’)
                state:=22;
                ch=getchar();
            end case;
case ‘22’:
            return(keyword,BOTH);
            exit while;
case ‘23’:
            if(ch==’I’)
                state:=24;
                ch=getchar();
            else if(ch==’O’)
                state:=32;
                ch=getchar();
            end case;
```

```
case ‘24’:
            if(ch==’V’)
                state:=25;
                ch=getchar();
            end case;


case ‘25’:
            if(ch==’I’)
                state:=26;
                ch=getchar();
            else
                state:=31;
            end case;
case ‘26’:
            if(ch==’S’)
                state:=27;
                ch=getchar();
            end case;
case ’27’:
            if(ch==’I’)
                state:=28;
                ch=getchar();
            end case;
case ‘28’:
            if(ch==’O’)
                state:=29;
                ch=getchar();
            end case;
case ’29’:
            if(ch==’N’)
                state:=30;
                ch=getchar();
            end case;
case ‘30’:
            return(operator,DIVISION);
            exit while;
            end case;
case ‘31’:
            unput(ch);
            return(operator,DIV);
            exit while;
            end case;
case ‘32’:
            return(keyword,DO);
            exit while;
```

```
                end case;
        case ‘33’:
                if(ch==’S’)
                        state:=34;
                        ch=getchar();
                end case;
        case ‘34’:
                return(keyword,IS);
                exit while;
                end case;
        case ’35’:
                if(ch==’O’)
                        state:=36;
                        ch=getchar();
                end case;
        case ‘36’:
                if(ch==’W’)
                        state:=37;
                        ch=getchar();
                end case;
        case ‘37’:
                return(keyword,NOW);
                exit while;
```

```
                        end case;
        case ‘38’:
                        if(ch==’F’)
                                state:=39;
                                ch=getchar();
                        end case;



        case ‘39’:
                        return(keyword,OF);
                        exit while;
                        end case;
        case ‘40’:
                        if(ch==’U’)
                                state:=41;
                                ch=getchar();
                        end case;

        case ‘41’:
                        if(ch==’L’)
                                state:=42;
                                ch=getchar();
                        end case;
```

```
case ‘42’:
            if(ch==’T’)
                state:=43;
                ch=getchar();
            else
                state:=54;
            end case;
case ‘43’:
        if(ch==’I’)
                state:=44;
                ch=getchar();
            end case;
case ‘44’:
            if(ch==’P’)
                state:=45;
                ch=getchar();
            end case;
case ‘45’:
        if(ch==’L’)
                state:=46;
                ch==getchar();
            end case;
case ‘46’:
```

```
                    if(ch==’I’)
                        state:=47;
                        ch=getchar();
                    end case;
        case ‘47’:
                    if(ch==’C’)
                        state:=48;
                        ch=getchar();
                    end case;
        case ‘48’:
                if(ch==’A’)
                        state:=49;
                        ch=getchar();
                    end case;
        case ‘49’:
                    if(ch==’T’)
                        state:=50;
                        ch=getchar();
                    end case;
        case ‘50’:
                    if(ch==’I’)
                        state:=51;
                        ch=getchar();
                    end case;
        case ‘51’:
                    if(ch==’O’)
                        state:=52;
                        ch=getchar();
                    end case;
        case ‘52’:
                    if(ch==’N’)
                        state:=53;
                        ch=getchar();
                    end case;
        case ‘53’:
                    return(operator,MULTIPLICATION);
                    exit while;
                    end case;

        case ‘54’:
                    unput(ch);
                    return(operator,MUL);
                    exit while;
                    end case;
        case ‘55’:
                    if(ch==’O’)
                        state:=56;
                        ch=getchar();
                    end case;
        case ‘56’:
                    return(keyword,TO);
                    exit while;
                    end case;
        case ‘57’:
                    if(ch==’I’)
                        state:=58;
                        ch=getchar();
                    else if(ch==’H’)
                        state:=61;
                        ch=getchar();
                    end case;
        case ‘58’:
                    if(ch==’T’)
                        state:=59;
                        ch=getchar();
                    end case;
        case ‘59’:
                    if(ch==’H’)
                        state:=60;
                        ch=getchar();
                    end case;
        case ‘60’:
                    return(keyword,WITH);
                    exit while;
        case ‘61’:
                    if(ch==’A’)
                        state:=62;
                        ch=getchar();
                    end case;
        case ‘62’:
                    if(ch==’T’)
                        state:=63;
                        ch=getchar();
                    end case;
        case ‘63’:
                    return(keyword,WHAT);
                    exit while;
                    end case;
        case ‘64’:
                    if(ch==’U’)
                        state:=65;
                        ch=getchar();
```

```
            end case;
    case ‘65’:
            if(ch==’B’)
                state:=66;
                ch=getchar();
            end case;
    case ‘66’:
            if(ch==’T’)
                state:=67;
                ch=getchar();
            else
                state:=75;
            end case;
    case ‘67’:
            if(ch==’R’)
                state:=68;
                ch=getchar();
            end case;
    case ‘68’:
            if(ch==’A’)
                state:=69;
                ch=getchar();
            end case;
```

```
case ‘69’:
            if(ch==’C’)
                state:=70;
                ch=getchar();
            end case;
case ‘70’:
            if(ch==’T’)
                state:=71;
                ch=getchar();
            end case;
case ‘71’:
            if(ch==’I’)
                state:=72;
                ch=getchar();
            end case;
case ‘72’:
            if(ch==’O’)
                state:=73;
                ch=getchar();
            end case;
case ‘73’:
            if(ch==’N’)
                state:=74;
                ch=getchar();
            end case;
    case ‘74’:
            return(operator,SUBTRACTION);
            exit while;
            end case;
    case ‘75’:
            unput(ch);
            return(operator,SUB);
            exit while;
    case ‘76’:
            if(ch==’E’)
                state:=77;
                ch=getchar();
            end case;
    case ‘77’:
            if(ch==’S’)
                state:=78;
                ch=getchar();
            end case;
    case ‘78’:
            if(ch==’U’)
                state:=79;
                ch=getchar();
            end case;
case ‘79’:
            if(ch==’L’)
                state:=80;
                ch=getchar();
            end case;
case ‘80’:
            if(ch==’T’)
                state:=81;
                ch=getchar();
            end case;
case ‘81’:
            return(keyword,RESULT);
            exit while;
            end case;
case ‘82’:
            return(punctuation, . );
            exit while;
            end case;
case ‘83’:
            return(punctuation,?);
            exit while;
```

```
                        end case;
            case ‘84’:
                        if(ch==[0-9])
                                ch=getchar();
                        else if(ch==’.’)
                                state:=85;
                                ch=getchar();
                        else
                                state=88;
                        end case;
            case ‘85’:
                        if(ch==[0-9])
                                state:=86;
                                ch=getchar();
                        end case;
            case ‘86’:
                        if(ch==[0-9])
                                ch=getchar();
                        else
                                state:=87;
                        end case;
            case ‘87’:
                        unput(ch);
```

```
                return(constant,FLOAT);
                exit while;
                end case;
        case ‘88’:
                unput(ch);
                return(constant,INT);
                exit while;
                end case;
        case ‘89’:
                return(error);
                exit while;
                end case;
        default:
                exit while;
                end case;
        }
```

2.0.3 Implementation of lexer

Flex Program:

```
%{
        #include<stdio.h>
%}


%%
[0-9]+  {printf("valid integer constant:%s\n" , yytext);}
([0-9]*[.])[0-9]+ {printf("valid float constant:%s\n" , yytext); }


("addition")|("Addition")|("ADDITION")|("add")|("ADD")|("Add")|("subtrac
tion")|("Subtraction")|("SUBTRACTION")|("sub")|("Sub")|("SUB")
{printf("valid operator:%s\n" , yytext); }
("multiplication")|("Multiplication")|("MULTIPLICATION")|("mul")|("MUL
")|("Mul")|("DIVISION")|("division")|("Division")|("div")|("DIV")|("Div")
{printf("valid operator:%s\n" , yytext); }


("what")|("What")|("WHAT")|("is")|("IS")|("Is")|("result")|("Result")|("RESU
LT")|("do")|("Do")|("DO")|("BOTH")|("Both")|("both") {printf("valid
keyword:%s\n" , yytext); }


("now")|("NOW")|("Now")|("answers")|("ANSWERS")|("Answers")|("with"
)|("With")|("WITH") {printf("valid keyword:%s\n" , yytext); }
("To")|("to")|("TO")|("of")|("Of")|("OF")|("also")|("Also")|("ALSO")
{printf("valid keyword:%s\n" , yytext); }


(\?) {printf("valid punctuation: %s\n" , yytext); }
```

```
(\.) {printf("valid punctuation: %s\n" , yytext); }

(" ") {printf(""); }

. { printf("Invalid token:%s\n" , yytext);}
%%

main()
{ yylex();
}
yywrap()
{}
```

2.0.4 Execution environment setup

Step by Step Guide to Install FLEX and Run FLEX Program using Command Prompt(cmd)

Operating system: Windows 10

Step 1 /*For downloading CODEBLOCKS */

- Open your Browser and type in "codeblocks"

- Goto to Code Blocks and go to downloads section - Click on "Download the binary release"

- Download codeblocks-20.03mingw-setup.exe

- Install the software keep clicking on next /*For downloading FLEX GnuWin32 */

- Open your Browser and type in "download flex gnuwin32"
- Goto to "Download GnuWin from SourceForge.net"

- Downloading will start automatically

- Install the software keep clicking on next

/*SAVE IT INSIDE C FOLDER*

Step 2 /*PATH SETUP FOR CODEBLOCKS*/

- After successful installation Goto program files->CodeBlocks-->MinGW-->Bin

- Copy the address of bin :- it should somewhat look like this C:\Program Files (x86)\CodeBlocks\MinGW\bin

- Open Control Panel-->Goto System-->Advance System Settings-->Environment Variables

- Environment Variables--> Click on Path which is inside System variables

- Click on edit

- Click on New and paste the copied path to it:-

- C:\Program Files (x86)\CodeBlocks\MinGW\bin

- Press Ok!

Step 3 /*PATH SETUP FOR GnuWin32*/

- After successful installation Goto C folder

- Goto GnuWin32-->Bin

- Copy the address of bin it should somewhat look like this C:\GnuWin32\bin - Open Control Panel-->Goto System-->Advance System Settings-->Environment Variables

- Environment Variables--> Click on Path which is inside System variables - Click on edit

- Click on New and paste the copied path to it:-

- C:\GnuWin32\bin

- Press Ok

Step 4

- Create a folder on Desktop flex_programs or whichever name you like

- Open notepad type in a flex program

- Save it inside the folder like filename.l

-Note :- also include “”” void yywrap(){} “””””” in the .l file

/*Make sure while saving save it as all files rather than as a text document*/

Step 5

/*To RUN FLEX PROGRAM*/

- Goto to Command Prompt(cmd)

- Goto the directory where you have saved the program

- Type in command :- flex filename.l

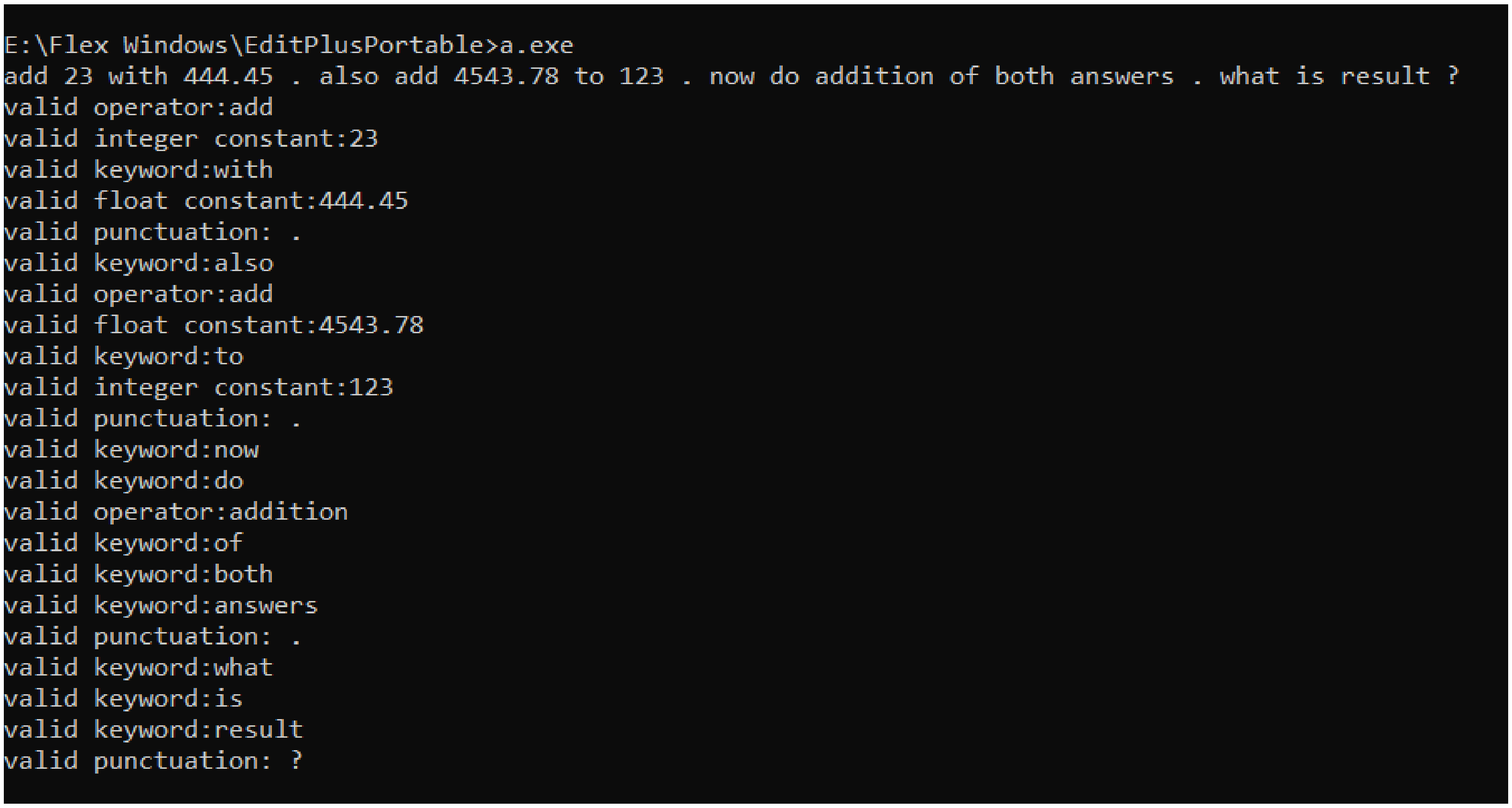
- Type in command :- gcc lex.yy.c

- Execute/Run for windows command prompt :- **a.exe**

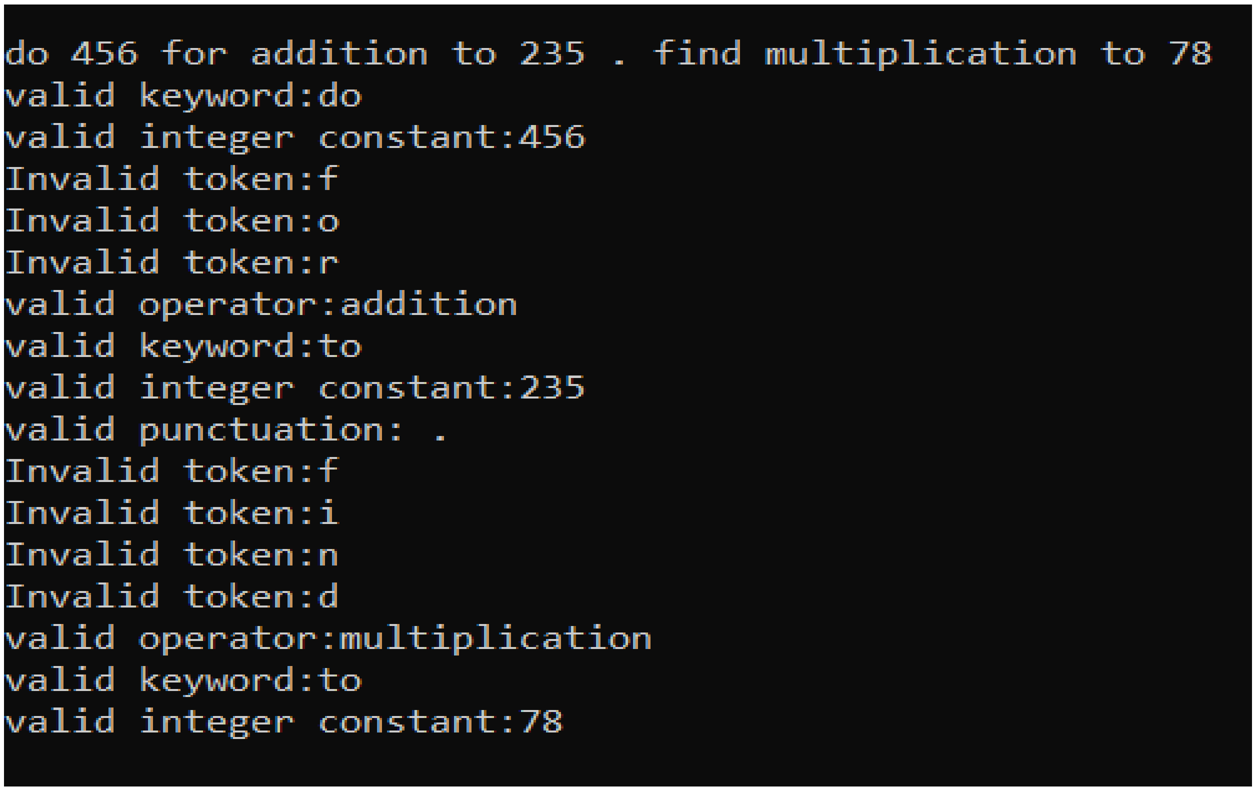
Step 6

- Finished

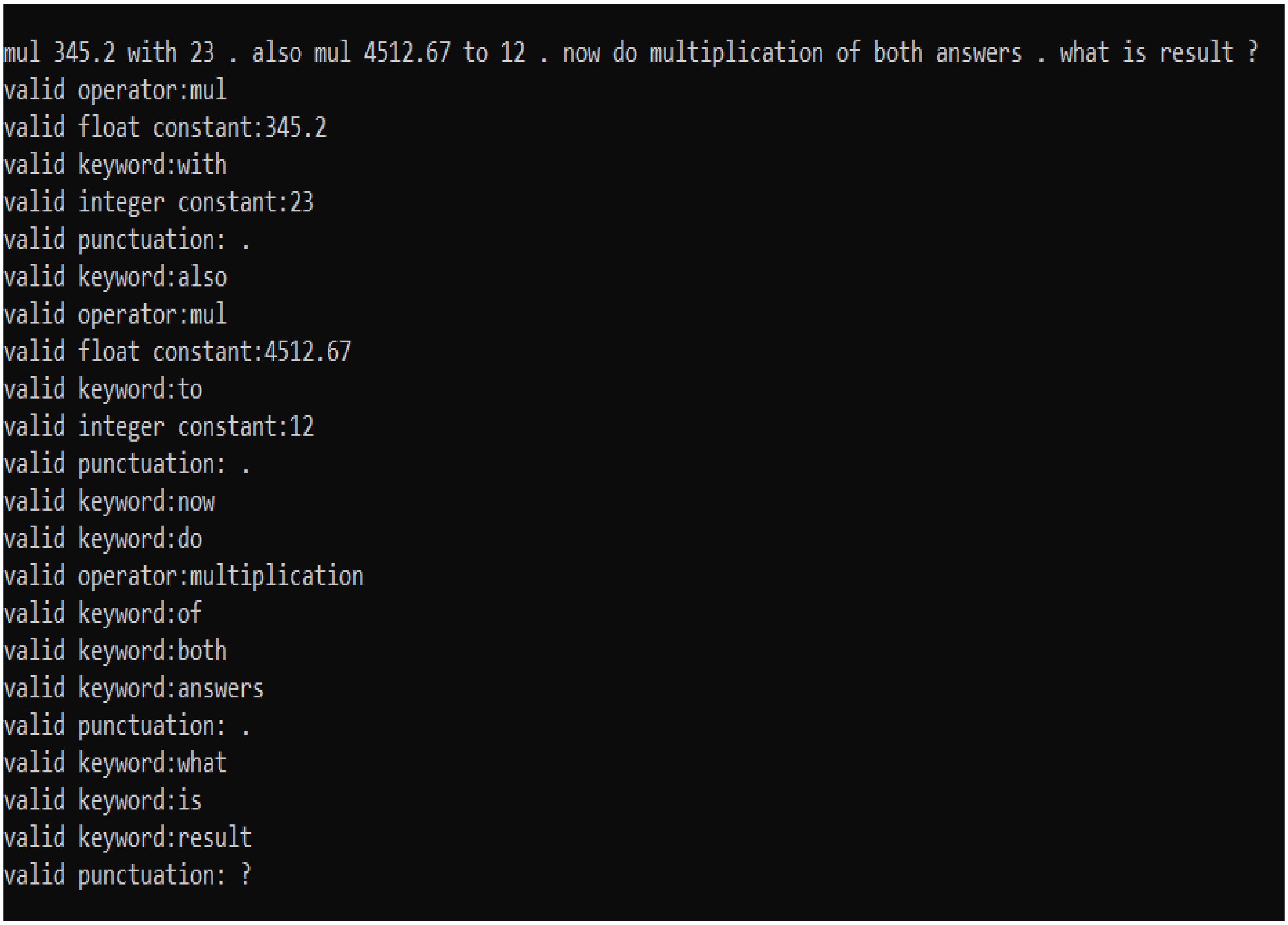
2.0.5 Output screenshots of Lexer



```
E:\Flex Windows\EditPlusPortable>a.exe
add 23 with 444.45 . also add 4543.78 to 123 . now do addition of both answers . what is result ?
valid operator:add
valid integer constant:23
valid keyword:with
valid float constant:444.45
valid punctuation: .
valid keyword:also
valid operator:add
valid float constant:4543.78
valid keyword:to
valid integer constant:123
valid punctuation: .
valid keyword:now
valid keyword:do
valid operator:addition
valid keyword:of
valid keyword:both
valid keyword:answers
valid punctuation: .
valid keyword:what
valid keyword:is
valid keyword:result
valid punctuation: ?
```



```
do 456 for addition to 235 . find multiplication to 78
valid keyword:do
valid integer constant:456
Invalid token:f
Invalid token:o
Invalid token:r
valid operator:addition
valid keyword:to
valid integer constant:235
valid punctuation: .
Invalid token:f
Invalid token:i
Invalid token:n
Invalid token:d
valid operator:multiplication
valid keyword:to
valid integer constant:78
```



```
mul 345.2 with 23 . also mul 4512.67 to 12 . now do multiplication of both answers . what is result ?
valid operator:mul
valid float constant:345.2
valid keyword:with
valid integer constant:23
valid punctuation: .
valid keyword:also
valid operator:mul
valid float constant:4512.67
valid keyword:to
valid integer constant:12
valid punctuation: .
valid keyword:now
valid keyword:do
valid operator:multiplication
valid keyword:of
valid keyword:both
valid keyword:answers
valid punctuation: .
valid keyword:what
valid keyword:is
valid keyword:result
valid punctuation: ?
```

3.0 SYNTAX ANALYZER DESIGN

3.0.1 Grammar rules

S-> FIRST F

F -> SECOND | FOURTH

FIRST-> O N X N .

SECOND-> also FIRST THIRD FOURTH.

THIRD-> now do O of both answers.

FOURTH-> what is result ?

O(operators)-> addition | add | subtraction | sub | multiplication | mul | division | div

N(number)-> float | int

X-> with | to

3.0.2 Yacc Implementation

Flex Code :

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

%}
SIMPLE [0-9]+
FLOAT [0-9]*[.][0-9]+
%%

{SIMPLE} {printf("valid integer constant:%s\n" , yytext); return NUMBER;}
{FLOAT} {printf("valid float constant:%s\n" , yytext); return NUMBER; }

("addition")|("Addition")|("ADDITION")|("add")|("ADD")|("Add")|("
subtraction")|("Subtracti
on")|("SUBTRACTION")|("sub")|("Sub")|("SUB") {printf("valid
operator:%s\n" , yytext); return OP; }
```

```
("multiplication")|("Multiplication")|("MULTIPLICATION")|("mul"
)|("MUL")|("Mul")|("DIVI
SION")|("division")|("Division")|("div")|("DIV")|("Div")
{printf("valid operator:%s\n" , yytext); return OP ;}


("what is result ?")|("WHAT IS RESULT ?")|("What is
result ?") {printf("valid sentence:%s\n",yytext); return
W;}

("also")|("Also")|("ALSO") {printf("valid Keyword: also \n");

return A;} ("now do")|("Now do")|("NOW DO") {printf("valid

Keywords : now , do \n"); return N;}

("OF BOTH ANSWERS")|("of both answers") {printf("valid
Keywords : of ,both , answers \n"); return B;}

(\.) {printf("valid punctuation: %s\n" , yytext); return C; }

("with")|("WITH") { printf("valid Keyword: with \n"); return P;}

("TO")|("to") { printf("valid Keyword: to \n");return T;}

(" ") {printf(""); }

\n {return NL;}
. {printf("Error at %s\n",yytext);}

%%

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

YACC Code:

```
%{
        #include<stdio.h>
        #include<stdlib.h>
        int yyerror(char *s);
        int yyparse(void);
%}
%token NL A B N OP W C P T NUMBER

%%
S : T1 F NL {printf("Given statement is
valid\n"); return 0;} F : E
  | R
T1 : O N1 X N1 C
E : A T1 U R
U : N O B C
R : W
O : OP
N1 : NUMBER
X : P
  | T
%%

int main()
{
 while(1)
  {
  printf("Enter:");
 yyparse(); }
}
int yyerror(char *s)
{
 fprintf(stderr,"%s\n",s);
 exit(0);
}
```

3.0.3 Execution Environment setup

Step 1: Installing YACC/Bison

Command to install yacc/bison: sudo apt-install bison

Step 2: Saving Flex file

- Writing the source code in a file and saving it with “ .y ” extension at any desired directory / location in the system.

Step 3: Running the Flex code

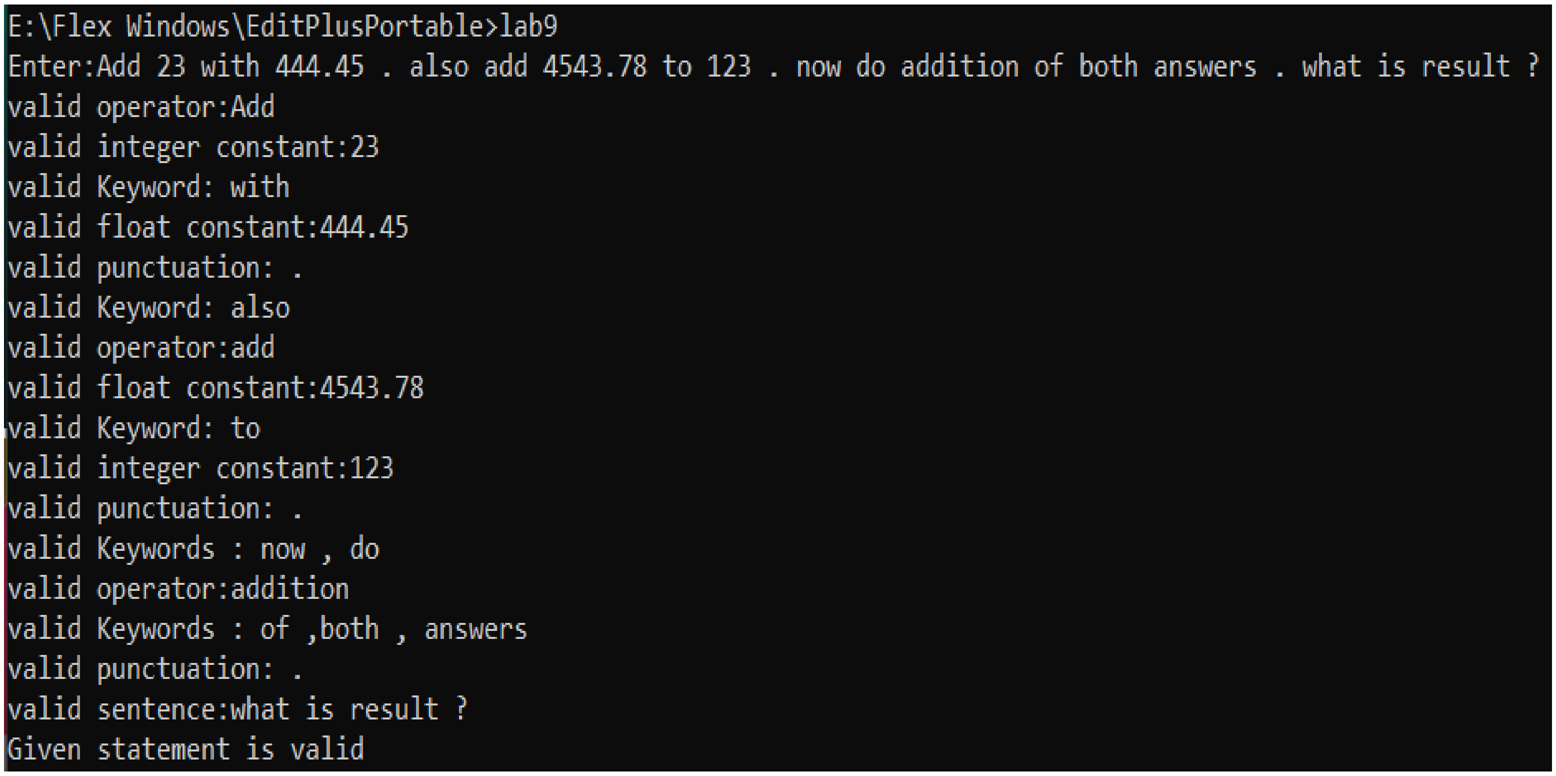
- Run following commands:

```
1. flex <fileName>.l
2. bison -dy <fileName>.y
3. gcc lex.yy.c  y.tab.c -o  <exeName>.exe
4. <exeName>
```

3.0.4 Output screenshots of YACC based implementation

Valid Input :
Add 23 with 444.45 . also add 4543.78 to 123 . now do addition of both answers . what is result ?

Output:

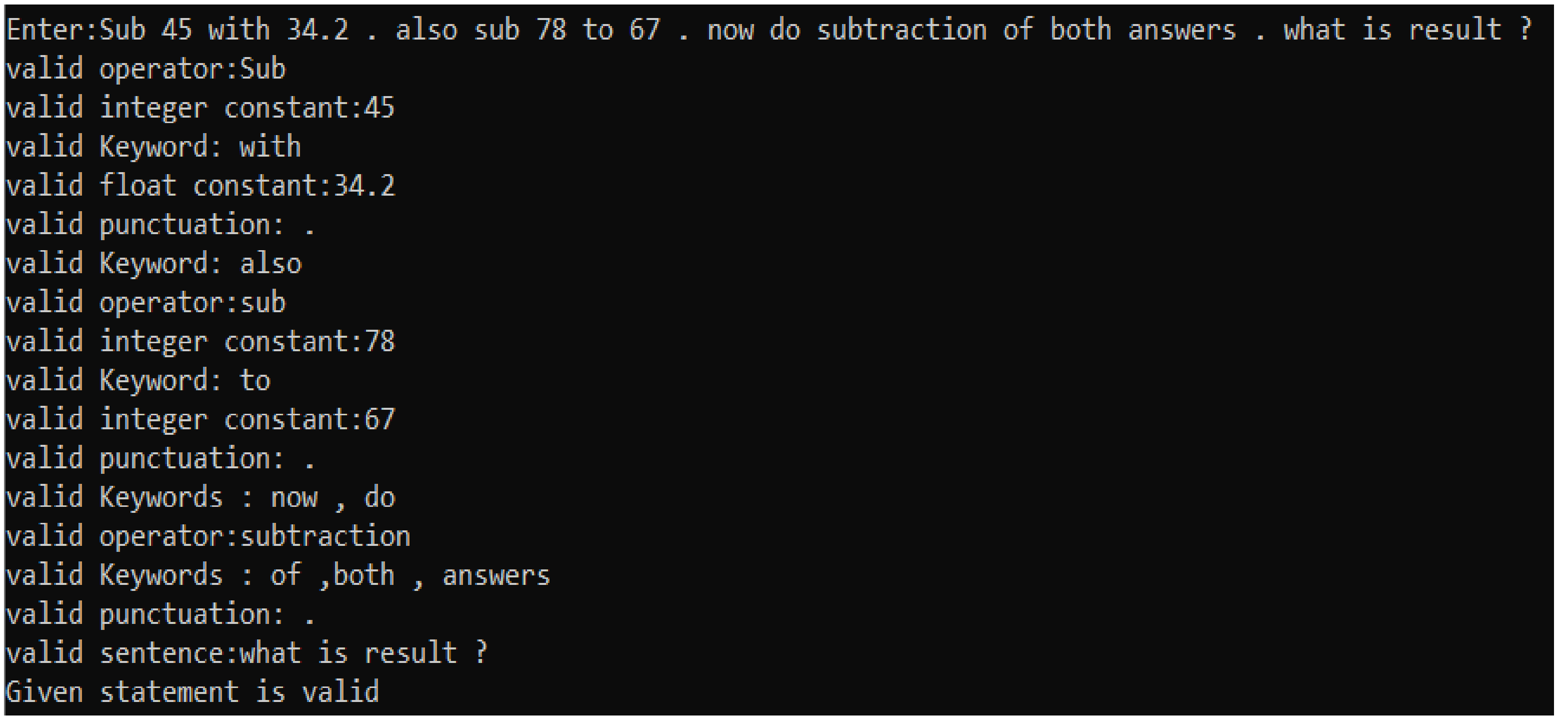


```
E:\Flex Windows\EditPlusPortable>lab9
Enter:Add 23 with 444.45 . also add 4543.78 to 123 . now do addition of both answers . what is result ?
valid operator:Add
valid integer constant:23
valid Keyword: with
valid float constant:444.45
valid punctuation: .
valid Keyword: also
valid operator:add
valid float constant:4543.78
valid Keyword: to
valid integer constant:123
valid punctuation: .
valid Keywords : now , do
valid operator:addition
valid Keywords : of ,both , answers
valid punctuation: .
valid sentence:what is result ?
Given statement is valid
```

Valid Input:

Sub 45 with 34.2 . also sub 78 to 67 . now do subtraction of both answers . what is result ?

Output:

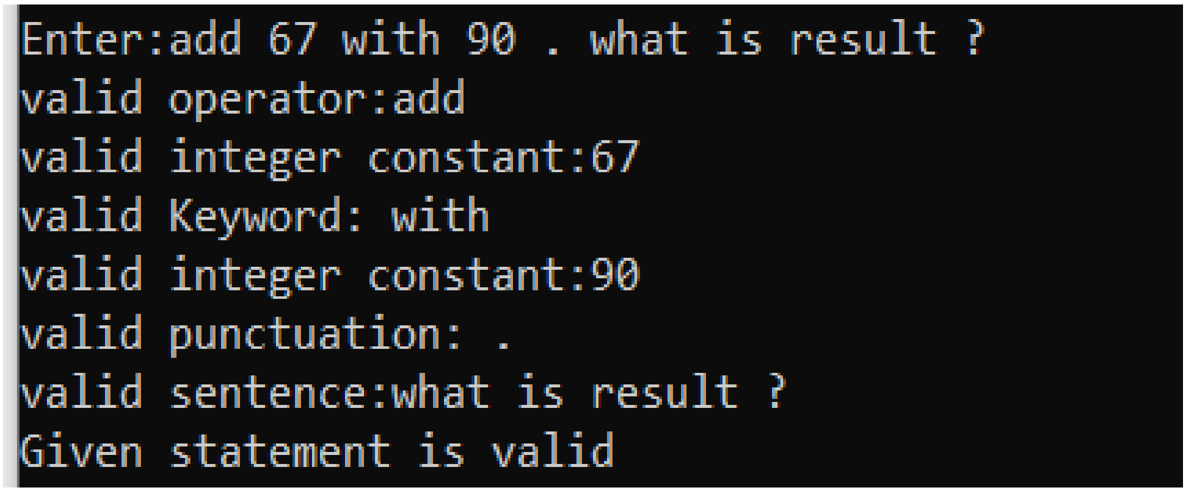


```
Enter:Sub 45 with 34.2 . also sub 78 to 67 . now do subtraction of both answers . what is result ?
valid operator:Sub
valid integer constant:45
valid Keyword: with
valid float constant:34.2
valid punctuation: .
valid Keyword: also
valid operator:sub
valid integer constant:78
valid Keyword: to
valid integer constant:67
valid punctuation: .
valid Keywords : now , do
valid operator:subtraction
valid Keywords : of ,both , answers
valid punctuation: .
valid sentence:what is result ?
Given statement is valid
```

Valid Input:

add 67 with 90 . what is result ?

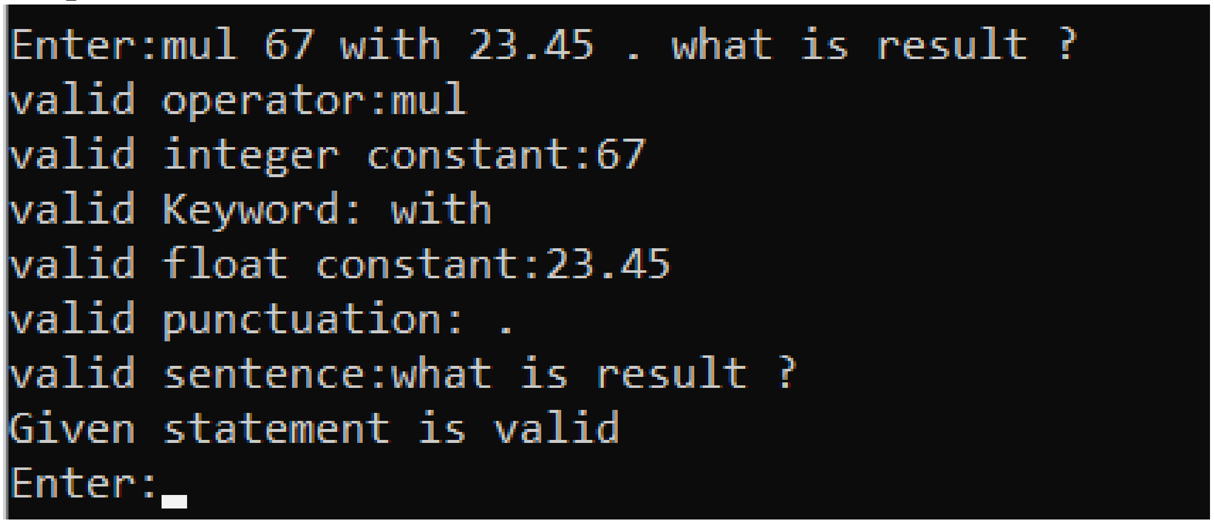
Output:



```
Enter:add 67 with 90 . what is result ?
valid operator:add
valid integer constant:67
valid Keyword: with
valid integer constant:90
valid punctuation: .
valid sentence:what is result ?
Given statement is valid
```

Valid Input:
mul 67 with 23.45 . what is result ?

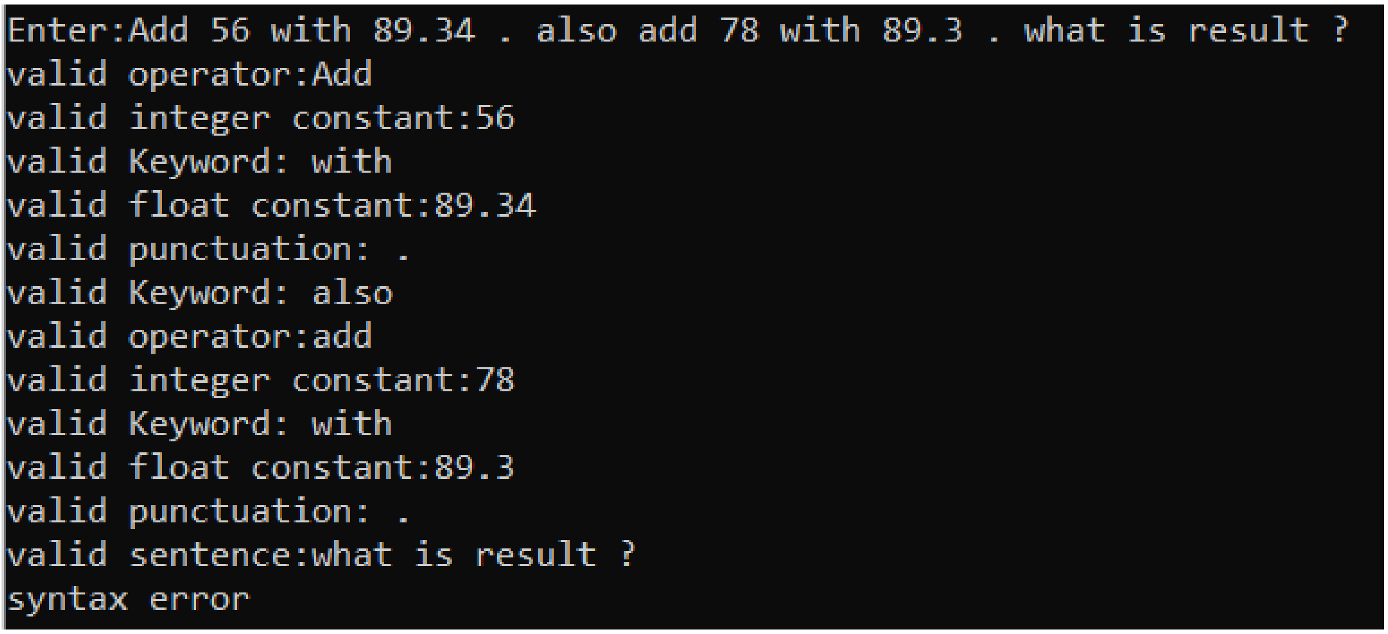
Output:



```
Enter:mul 67 with 23.45 . what is result ?
valid operator:mul
valid integer constant:67
valid Keyword: with
valid float constant:23.45
valid punctuation: .
valid sentence:what is result ?
Given statement is valid
Enter:_
```

Invalid Input :
Add 56 with 89.34 . also add 78 with 89.3 . What is the result ?

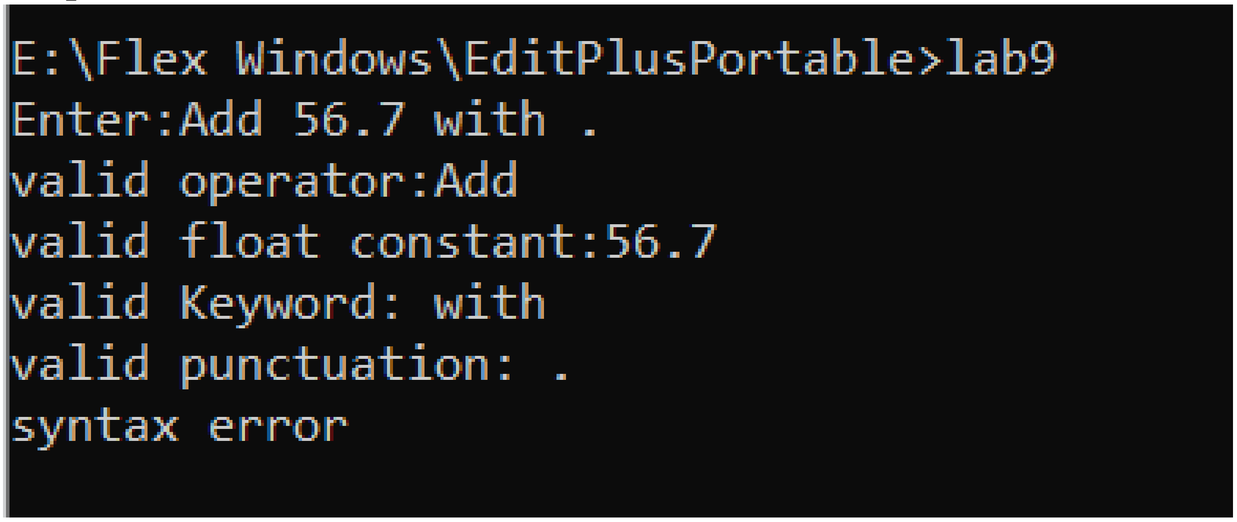
Output:



```
Enter:Add 56 with 89.34 . also add 78 with 89.3 . what is result ?
valid operator:Add
valid integer constant:56
valid Keyword: with
valid float constant:89.34
valid punctuation: .
valid Keyword: also
valid operator:add
valid integer constant:78
valid Keyword: with
valid float constant:89.3
valid punctuation: .
valid sentence:what is result ?
syntax error
```

Invalid Input:
Add 56.7 with .

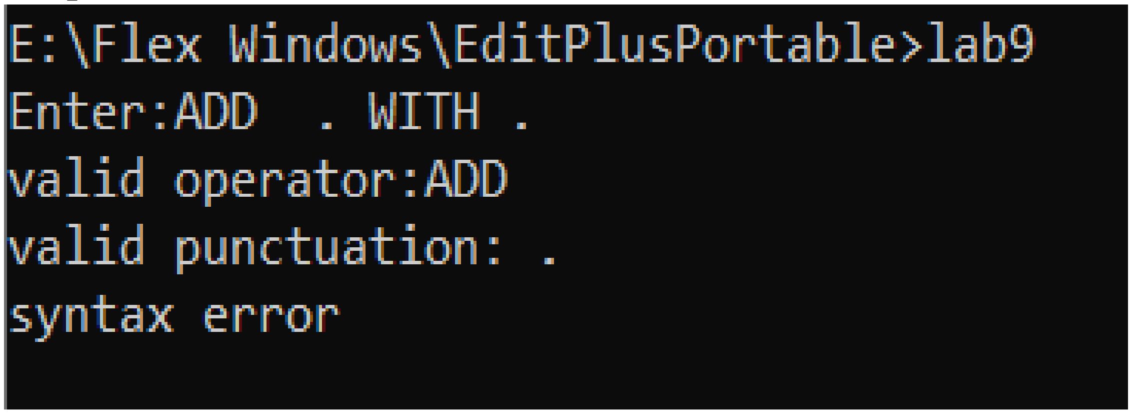
Output:



```
E:\Flex Windows\EditPlusPortable>lab9
Enter:Add 56.7 with .
valid operator:Add
valid float constant:56.7
valid Keyword: with
valid punctuation: .
syntax error
```

Invalid Input:
ADD . WITH .

Output:



```
E:\Flex Windows\EditPlusPortable>lab9
Enter:ADD  . WITH .
valid operator:ADD
valid punctuation: .
syntax error
```

4.0 CONCLUSION

Compiler for our layman friendly calculator have been made to the best of our knowledge by applying the concepts learnt so far from our academic curriculum and from the well of resources - internet.

This project gave us an opportunity to have a better understanding of the compiler concepts that we learnt by implementing them in great detail. We would like to extend thanks to Prof. Nikita P. Desai for encouraging us to implement this project. This project has helped us build our problem solving and team work skills and has given us a better insight of how actually the compiler works in real terms.