Project Report
on

Compiler for

<< String Operations Using Gujarati Language>>

Developed by

Nakrani Dhrumil-IT082-19ITUES054

Nikhil Nasit-IT083-19ITUOS065

Gurvinder Singh-IT084-19ECUOG038

Department of Information Technology
Faculty of Technology, Dharmsinh Desai University
College Road, Nadiad-387001 2020-2021

DHARMSINH DESAI UNIVERSITY
NADIAD-387001, GUJARAT

CERTIFICATE

This is to certify that the project entitled " **Compiler for String Operations using Gujarati language**" is a bonafied report of the work carried out by

1) Mr. Nakrani Dhrumil, Student ID No: 19ITUES054

2)Mr. Nikhil Nasit, Student ID No: 19ITUOS065

3) Mr. Gurvinder Singh, Student ID No: 19ECUOG038

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 academic year 2021-2022.

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

Chatper1- Introduction

1.1 Project Details:

Language Name: String Operations using Gujarati language
Language description:

Write an appropriate language description for a layman language which can do string operations using Gujarati sentences ,written in roman script .
Example of valid program in this language are:-

1) shu 129 ane 129 bane sarkha number che ?

2) 32 , 34 mathi kayo number moto che ?

3) 32 , 34 mathi kayo number nano che ?

4) nano number kayo che 7 ke 2 ?

5) moto number kayo che 7 ke 2 ?

1.2 Project Planning

List of students with their Responsibilities:

IT082 NAKRANI DHRUMIL: Regular Expression, DFA Design, Algorithm Design and implementation, Scanner phase Implementation, Grammar rules, YACC implementation, Final Report.

IT083 NIKHIL NASIT: Regular Expression, DFA Design, Algorithm Design and implementation, Scanner phase Implementation, Grammar rules, YACC implementation, Final Report.

IT084 GURVINDER SINGH: Regular Expression, DFA Design, Algorithm Design and implementation, Scanner phase Implementation, Grammar rules, YACC implementation, Final Report.

Chatper2- Lexical Phase Design

2.1 Regular Expression:

Keywords :

RE-Token

ane - ane

mathi - mathi

kayo - kayo

nano - nano

moto - moto

number - number

bane - bane

sarkha - sarkha

shu – shu

ke – ke

che - che

Operations :

Values type : int and float

RE Token

[0-9]+ int

[0-9]+(.[0-9]+) float

Delimiters : {. , ? \t}

RE Token

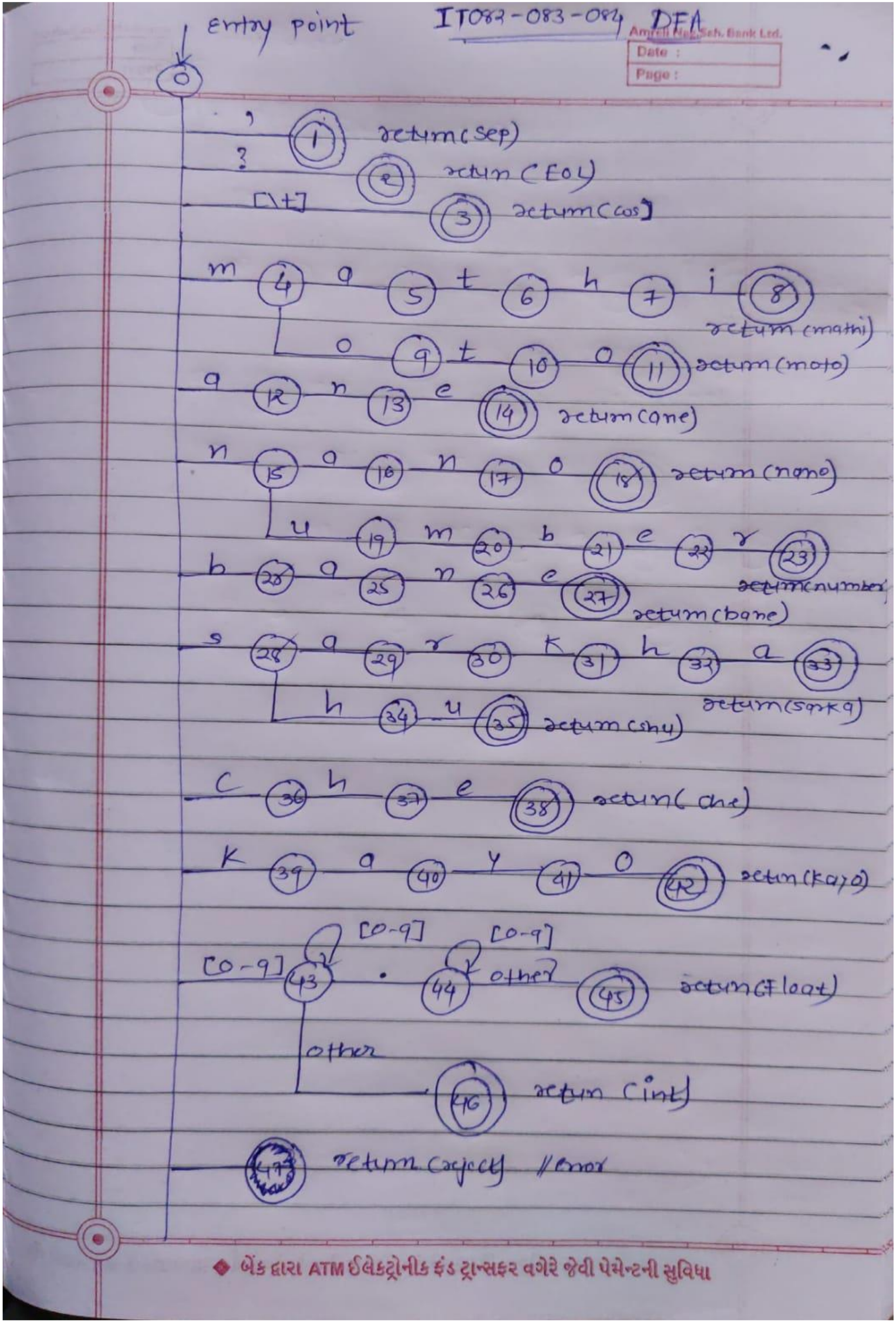
. eos

, sep

? qm

[\t] ws

2.2 DFA Design for LEXER:



2.3 Algorithm of LEXER:

```
switch(state)
        {
        case 0:
        {
            if(inputchar[i]==',')
            {
                i++;
                state=0;
                error=false;
            }
            else if(inputchar[i]=='?')
            {
                i++;
                state=0;
                error=false;
            }
            else if(inputchar[i]=='\t' || inputchar[i]==' ')
            {
                i++;
                state=0;
                error=false;
            }
            else if(inputchar[i]=='m')
            {
                i++;
                state=4;
                error=false;
            }
            else if(inputchar[i]=='a')
            {
                i++;
                state=8;
                error=false;
            }
            else if(inputchar[i]=='n')
            {
                i++;
                state=11;
                error=false;
            }
        else if(inputchar[i]=='b')
        {
          i++;
          state=20;
          error=false;
        }
        else if(inputchar[i]=='s')
        {
          i++;
          state=24;
          error=false;
        }
        else if(inputchar[i]=='c')
        {
          i++;
          state=30;
          error=false;
        }
        else if(inputchar[i]=='k')
        {
          i++;
          state=33;
          error=false;
        }
        else if(inputchar[i]>='0' && inputchar[i]<='9')
        {
          i++;
          state=37;
          error=false;
        }
        else
        {
          error=true;
        }
        break;

      } // end of case 0
      case 1:
      {
        cout << inputchar[i-1] << endl; // return seperator
        error=false;
        break;
```

```
}
case 2:
{
    cout << inputchar[i-1] << endl; // return EOI
    error=false;
    break;
}
case 3:
{
    cout << inputchar[i-1] << endl; // return whitespace
    error=false;
    break;
}
case 4:
{
    if(inputchar[i]=='o')
    {
        i++;
        state=5;
        error=false;
    }
    else
    {
        error=true;
    }
    break;
}
case 5:
{
    if(inputchar[i]=='t')
    {
        i++;
        state=6;
        error=false;
    }
    else
    {
        error=true;
    }
    break;
}
case 6:
```

```
{
   if(inputchar[i]=='o')
   {
      i++;
      state=7;
      error=false;
   }
   else
   {
      error=true;
   }
   break;
}
case 7:
{
   cout << inputchar[i-1] << endl; // return moto
   error=false;
   break;
}
case 8:
{
   if(inputchar[i]=='n')
   {
      i++;
      state=9;
      error=false;
   }
   else
   {
      error=true;
   }
   break;
}
case 9:
{
   if(inputchar[i]=='e')
   {
      i++;
      state=10;
      error=false;
   }
   else
```

```
        {
            error=true;
        }
        break;
    }
    case 10:
    {
        cout << inputchar[i-1] << endl; // return ane
        error=false;
        break;
    }
    case 11:
    {
        if(inputchar[i]=='a')
        {
            i++;
            state=12;
            error=false;
        }
        else if(inputchar[i]=='u')
        {
            i++;
            state=15;
            error=false;
        }
        else
        {
            error=true;
        }
        break;
    }
    case 12:
    {
        if(inputchar[i]=='n')
        {
            i++;
            state=13;
            error=false;
        }
        else
        {
            error=true;
        }
        break;
    }
    case 13:
    {
        if(inputchar[i]=='0')
        {
            i++;
            state=14;
            error=false;
        }
        else
        {
            error=true;
        }
        break;
    }
    case 14:
    {
        cout << inputchar[i-1] << endl; // return nano
        error=false;
        break;
    }
    case 15:
    {
        if(inputchar[i]=='m')
        {
            i++;
            state=16;
            error=false;
        }
        else
        {
            error=true;
        }
        break;
    }
    case 16:
    {
        if(inputchar[i]=='b')
        {
            i++;
            state=17;
            error=false;
        }
        else
        {
            error=true;
        }
        break;
    }
    case 17:
    {
        if(inputchar[i]=='e')
        {
            i++;
            state=18;
            error=false;
        }
        else
        {
            error=true;
        }
        break;
    }
    case 18:
    {
        if(inputchar[i]=='r')
        {
            i++;
            state=19;
            error=false;
        }
        else
        {
            error=true;
        }
        break;
    }
    case 19:
    {
        cout << inputchar[i-1] << endl; // return number
        error=false;
        break;
```

```
}
case 20:
{
   if(inputchar[i]=='a')
   {
      i++;
      state=21;
      error=false;
   }
   else
   {
      error=true;
   }
   break;
}
case 21:
{
   if(inputchar[i]=='n')
   {
      i++;
      state=22;
      error=false;
   }
   else
   {
      error=true;
   }
   break;
}
case 22:
{
   if(inputchar[i]=='e')
   {
      i++;
      state=23;
      error=false;
   }
   else
   {
      error=true;
   }
   break;
```

```
}
case 23:
{
    cout << inputchar[i-1] << endl; // return bane
    error=false;
    break;
}
case 24:
{
    if(inputchar[i]=='a')
    {
        i++;
        state=25;
        error=false;
    }
    else if(inputchar[i]=='h')
    {
        i++;
        state=30;
        error=false;
    }
    else
    {
        error=true;
    }
    break;
}
case 25:
{
    if(inputchar[i]=='r')
    {
        i++;
        state=26;
        error=false;
    }
    else
    {
        error=true;
    }
    break;
}
case 26:
```

```
{
   if(inputchar[i]=='k')
   {
      i++;
      state=27;
      error=false;
   }
   else
   {
      error=true;
   }
   break;
}
case 27:
{
   if(inputchar[i]=='h')
   {
      i++;
      state=28;
      error=false;
   }
   else
   {
      error=true;
   }
   break;
}
case 28:
{
   if(inputchar[i]=='o')
   {
      i++;
      state=29;
      error=false;
   }
   else
   {
      error=true;
   }
   break;
}
case 29:
```

```
{
    cout << inputchar[i-1] << endl; // return sarkho
    error=false;
    break;
}
case 30:
{
    if(inputchar[i]=='u')
    {
        i++;
        state=31;
        error=false;
    }
    else
    {
        error=true;
    }
    break;
}
case 31:
{
    cout << inputchar[i-1] << endl; //return shu
    error=false;
    break;
}
case 32:
{
    if(inputchar[i]=='h')
    {
        i++;
        state=33;
        error=false;
    }
    else
    {
        error=true;
    }
    break;
}
case 33:
{
    if(inputchar[i]=='e')
```

```
        {
            i++;
            state=34;
            error=false;
        }
        else
        {
            error=true;
        }
        break;
    }
    case 34:
    {
        cout << inputchar[i=1] << endl;//return che
        error=false;
        break;
    }
    case 35:
    {
        if(inputchar[i]=='a')
        {
            i++;
            state=36;
            error=false;
        }
        else
        {
            error=true;
        }
        break;
    }
    case 36:
    {
        if(inputchar[i]=='y')
        {
            i++;
            state=37;
            error=false;
        }
        else
        {
            error=true;
        }
        break;
    }
    case 37:
    {
        if(inputchar[i]=='o')
        {
            i++;
            state=38;
            error=false;
        }
        else
        {
            error=true;
        }
        break;
    }
    case 38:
    {
        cout << inputchar[i-1]<< endl; //return kayo
        error=false;
        break;
    }
    case 39:
    {
        if(inputchar[i]>='0' && inputchar[i]<='9')
        {
            i++;
            state=39;
            error=false;
        }
        else if(inputchar[i]==' ')
        {
            i++;
            error=false;
            state=42;
        }
        else if(inputchar[i]=='.')
        {
            i++;
            error=false;
            state=40;
        }
        else
        {
            error=true;
        }
        break;
    }
    case 40:
    {
        if(inputchar[i]>='0' && inputchar[i]<='9')
        {
            i++;
            state=40;
            error=false;
        }
        else if(inputchar[i]==' ')
        {
            i++;
            state=41;
            error=false;
        }
        else
        {
            error=true;
        }
        break;
    }
    case 41:
    {
        cout << "FLOAT" << endl; //return float
        error=false;
        break;
    }
    case 42:
    {
        cout << "INT" << endl; //return INT
        error=false;
        break;
    }
    default:
        break;
    }
```

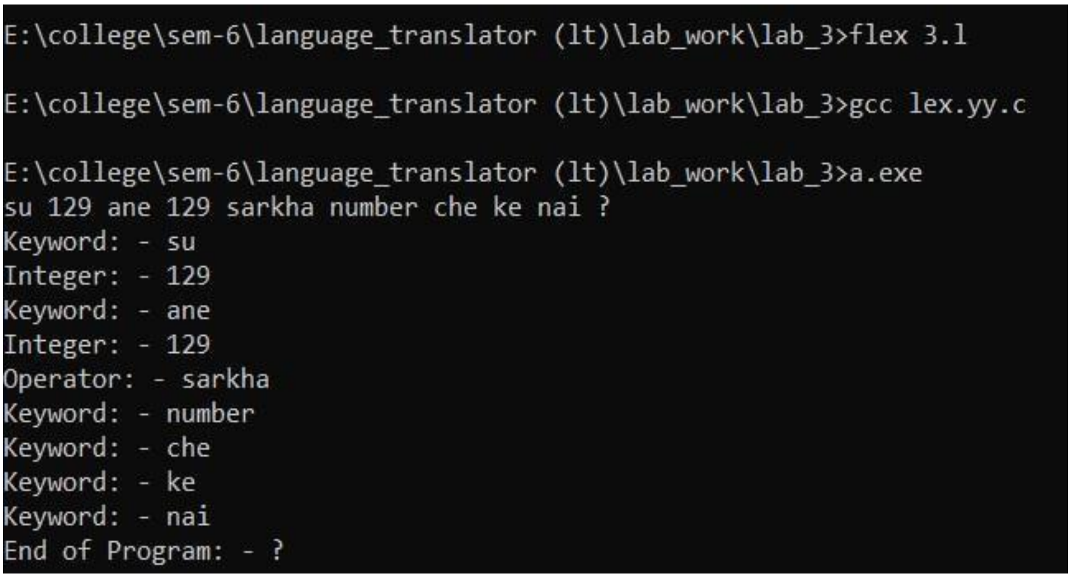
2.4 Implementation of LEXER:

Flex Program:

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

Keyword
"aa"|"bane"|"be"|"number"|"aakdo"|"kayo"|"che"|"ane"|"thi"|"ke"|"mathi"|
"nai"|" su"
Op          "moto"|"nano"|"sarkha"
Digit       [0-9]
Int         {Digit}+
Float       {Digit}+"."({Digit}+)
qm          "?"
ws1         [\n]
ws2         [\t]
sep         ","

" "         {printf("");}
{ws1}       {return 0;}
{ws2}       {printf("Give string without TAB");return 0;}
.           {printf("Invalid Token : %s\n",yytext);return 0;return *yytext;}
%%

int
yywrap(
){} int
main(){
yylex();
return 0;
}
```

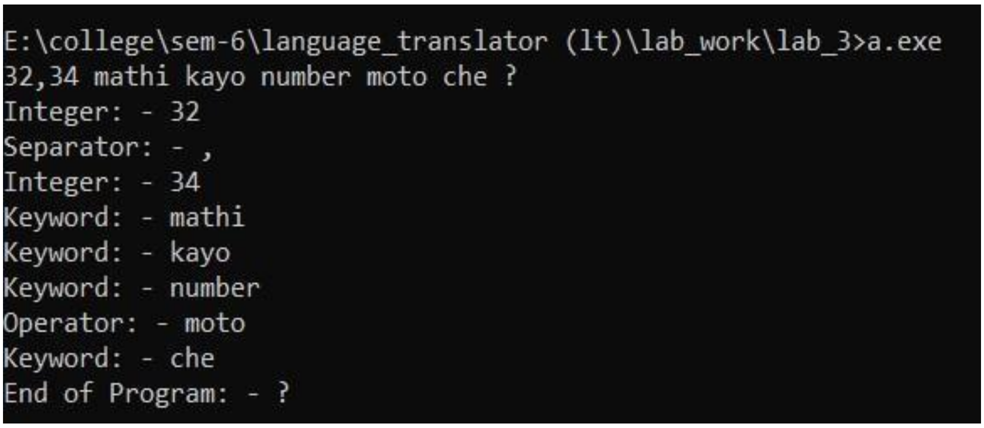
Output screenshots of lexer :

1.) su 129 ane 129 sarkha number che ke nai ?



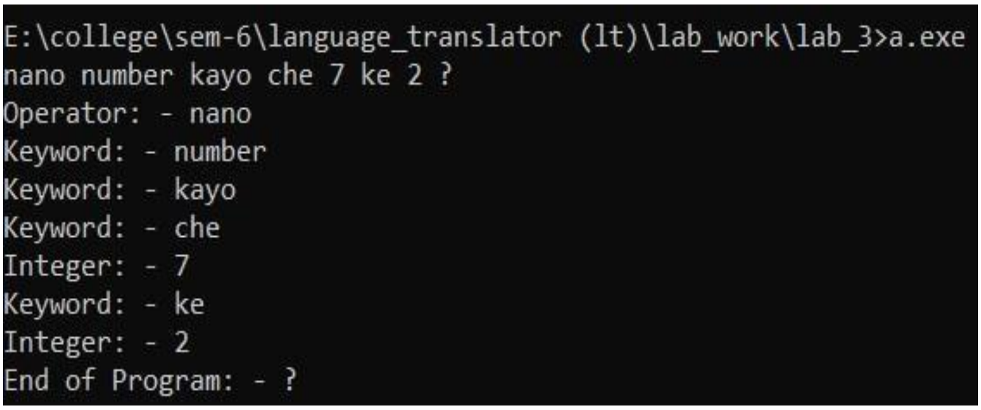
```
E:\college\sem-6\language_translator (lt)\lab_work\lab_3>flex 3.l

E:\college\sem-6\language_translator (lt)\lab_work\lab_3>gcc lex.yy.c

E:\college\sem-6\language_translator (lt)\lab_work\lab_3>a.exe
su 129 ane 129 sarkha number che ke nai ?
Keyword: - su
Integer: - 129
Keyword: - ane
Integer: - 129
Operator: - sarkha
Keyword: - number
Keyword: - che
Keyword: - ke
Keyword: - nai
End of Program: - ?
```

2.) 32,34 mathi kayo number moto che ?



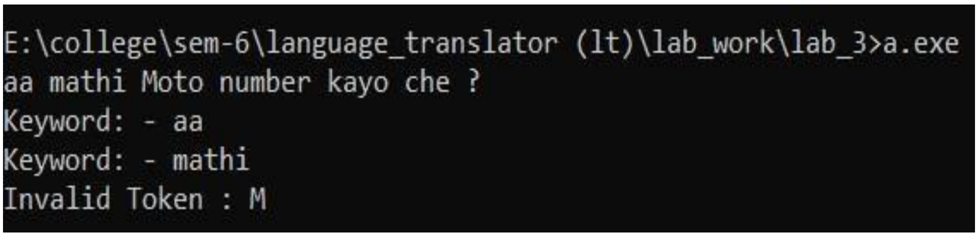
```
E:\college\sem-6\language_translator (lt)\lab_work\lab_3>a.exe
32,34 mathi kayo number moto che ?
Integer: - 32
Separator: - ,
Integer: - 34
Keyword: - mathi
Keyword: - kayo
Keyword: - number
Operator: - moto
Keyword: - che
End of Program: - ?
```

3.) nano number kayo che 7 ke 2 ?



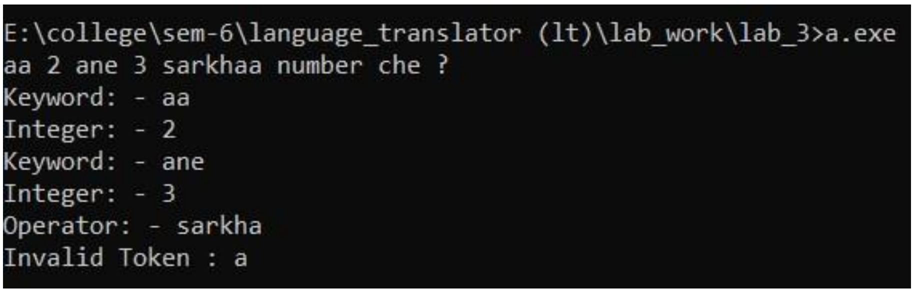
```
E:\college\sem-6\language_translator (lt)\lab_work\lab_3>a.exe
nano number kayo che 7 ke 2 ?
Operator: - nano
Keyword: - number
Keyword: - kayo
Keyword: - che
Integer: - 7
Keyword: - ke
Integer: - 2
End of Program: - ?
```

4.) Operation starting with capital Letter:



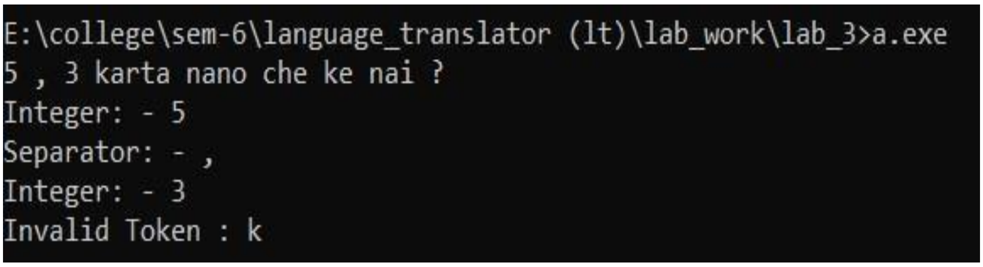
```
E:\college\sem-6\language_translator (lt)\lab_work\lab_3>a.exe
aa mathi Moto number kayo che ?
Keyword: - aa
Keyword: - mathi
Invalid Token : M
```

5.) Operation is invalid:



```
E:\college\sem-6\language_translator (lt)\lab_work\lab_3>a.exe
aa 2 ane 3 sarkhaa number che ?
Keyword: - aa
Integer: - 2
Keyword: - ane
Integer: - 3
Operator: - sarkha
Invalid Token : a
```

6.) Keyword is invalid:



```
E:\college\sem-6\language_translator (lt)\lab_work\lab_3>a.exe
5 , 3 karta nano che ke nai ?
Integer: - 5
Separator: - ,
Integer: - 3
Invalid Token : k
```

2.5 Execution Environment Setup:

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

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!

/*WARNING!!! PLEASE MAKE SURE THAT PATH OF CODEBLOCKS
IS BEFORE GNUWIN32---THE ORDER MATTERS*/

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 promt :- **a.exe**

Step 6:

- Finished

Chatper3- Syntax Analyzer Design

3.1 Grammar Rules:

A-> K1 I K2 I K3 ?

A-> I S I K4 O K5 ?

A-> O K6 I K7 I ?

O-> “nano”| “moto”

S-> ,

K1-> “shu”

K2-> “ane”

K3-> “bane sarkha number che”

K4-> “mathi kayo number”

K5-> “che”

K6-> “number kayo che”

K7-> “ke”

I-> int|float

3.2 YACC based implementation of Syntax Analyzer:

Project.l :

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

keyword "shu"
keyword1 "ane"
keyword2 "bane sarkha number che"
keyword3 "mathi kayo number"
keyword4 "che"
keyword5 "number kayo che"
keyword6 "ke"

op "moto"|"nano"|"sarkha"
eol "?"
sep ","
digit [0-9]
ws " "

%%
{keyword} {
        printf("%10s : keyword\n",yytext);
        return K;
        }
{keyword1} {
        printf("%10s : keyword1\n",yytext);
        return K1;
        }
{keyword2} {
        printf("%10s : keyword2\n",yytext);
        return K2;
        }
{keyword3} {
        printf("%10s : keyword3\n",yytext);
        return K3;
        }
{keyword4} {
        printf("%10s : keyword4\n",yytext);
```

```
            return K4;
            }
{keyword5} {
            printf("%10s : keyword5\n",yytext);
            return K5;
            }
{keyword6} {
            printf("%10s : keyword6\n",yytext);
            return K6;
            }
{eol} {
            printf("%10s : End of line\n",yytext);
            return E;
            }
{sep} {
            printf("%10s : Seperation\n",yytext);
            return S;
            }
{op} {
            printf("%10s : Operator\n",yytext);
            return O;
            }
{digit}+ {
            printf("%10s : digit\n",yytext);
            return I;
            }
{digit}+"."{digit}* {
            printf("%10s : digit\n",yytext);
            return I;
            }
{ws} {
            return W;
            }

. {
            printf("\nString is Invalid"); exit(1);
            }

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

```
}
```

Project.y:

```
%{
	#include<stdio.h>
	#include<stdlib.h>
	#define YYERROR_VERBOSE 1
	void yyerror(char *err);
%}

%token K K1 K2 K3 K4 K5 K6 O I E S W

%%
G: A {printf("\nThis sentence is valid.\n"); return 0;};
A: K W I W K1 W I W K2 W E {}|
I W S W I W K3 W O W K4 W E{}|
O W K5 W I W K6 W I W E {};
%%


void yyerror(char *err) {
	printf("Error: ");
	fprintf(stderr, "%s\n", err);
	exit(1);
}

void main(){
	printf("Enter String: ");
	yyparse();
	printf("\n Valid Expression\n");
}
```

3.3 Execution Environment Setup:

Download flex and bison from the given links.

http://gnuwin32.sourceforge.net/packages/flex.htm
http://gnuwin32.sourceforge.net/packages/bison.htm
when installing on windows you store this in c:/gnuwin32 folder and not in c:/program files(X86)/gnuwin32

Download IDE

https://sourceforge.net/projects/orwelldevcpp/ set environment variable for flex and bison.

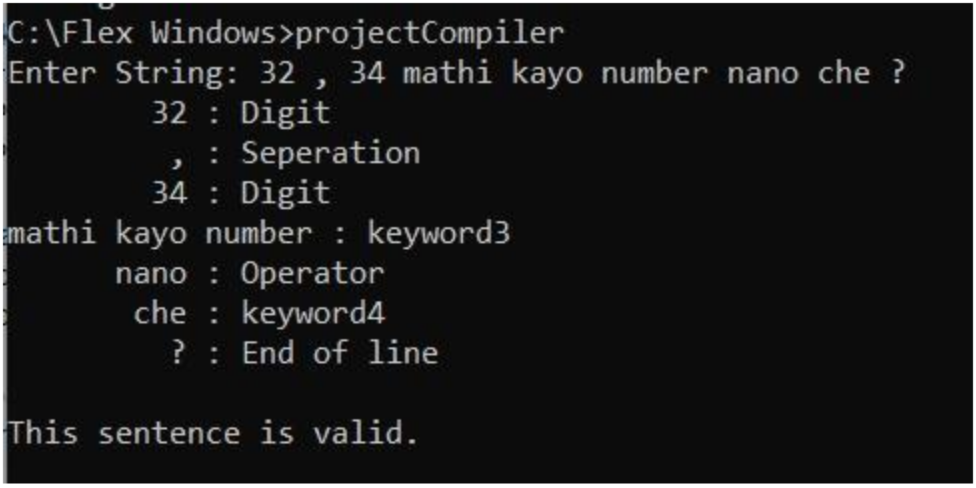
To run the program:

Open a prompt, cd to the directory where your ".l" and ".y" are, and compile them with:

```
flex yacc.l
bison -dy yacc.y
 gcc lex.yy.c y.tab.c -o yacc.exe
```

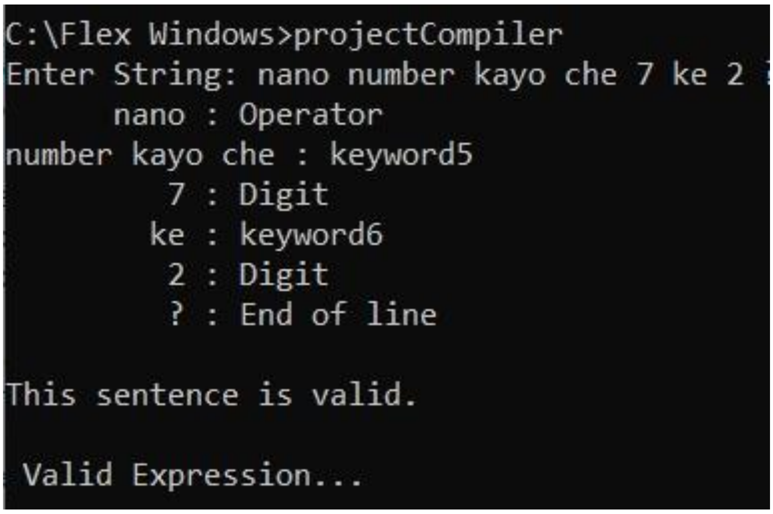
3.4 Output Screenshots of YACC based Implementation:

1)32 , 34 mathi kayo number nano che ?



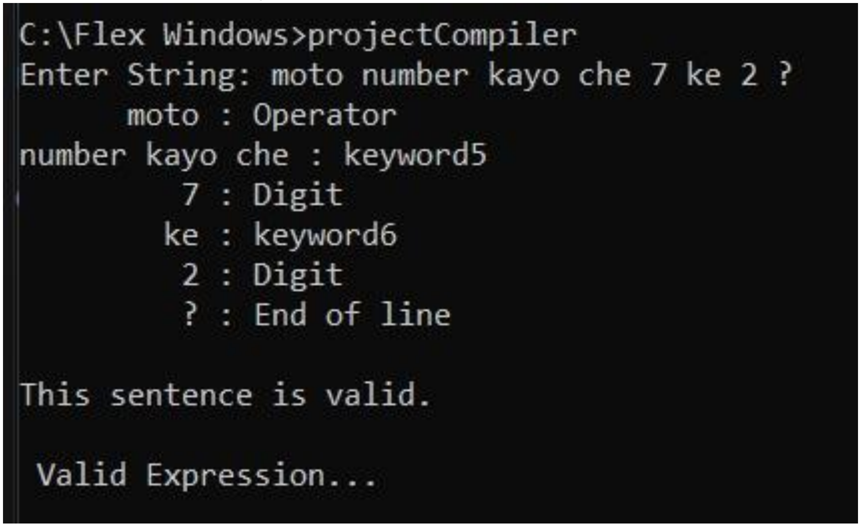
```
C:\Flex Windows>projectCompiler
Enter String: 32 , 34 mathi kayo number nano che ?
        32 : Digit
         , : Seperation
        34 : Digit
mathi kayo number : keyword3
      nano : Operator
       che : keyword4
         ? : End of line

This sentence is valid.
```

2)nano number kayo che 7 ke 2



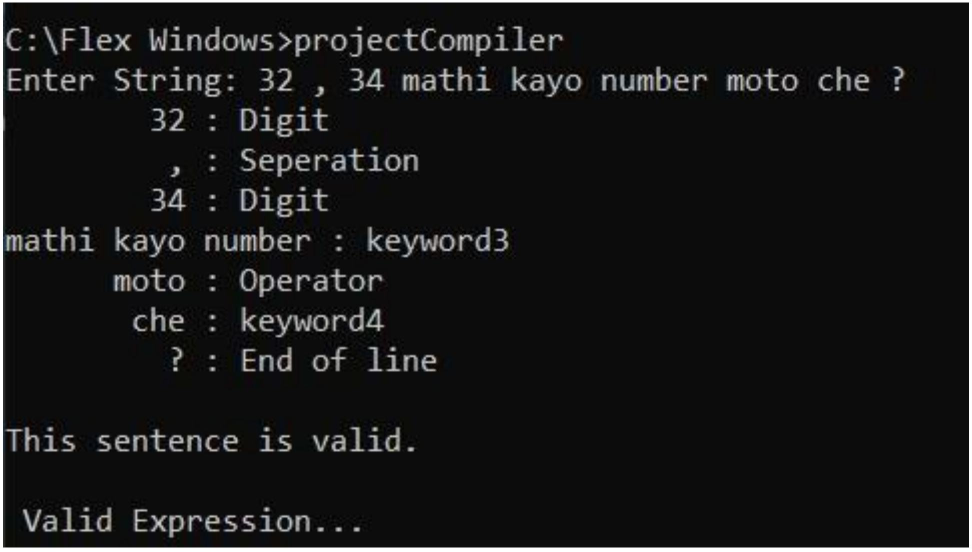
```
C:\Flex Windows>projectCompiler
Enter String: nano number kayo che 7 ke 2
      nano : Operator
number kayo che : keyword5
         7 : Digit
        ke : keyword6
         2 : Digit
         ? : End of line

This sentence is valid.

 Valid Expression...
```

3)moto number kayo che 7 ke 2 ?



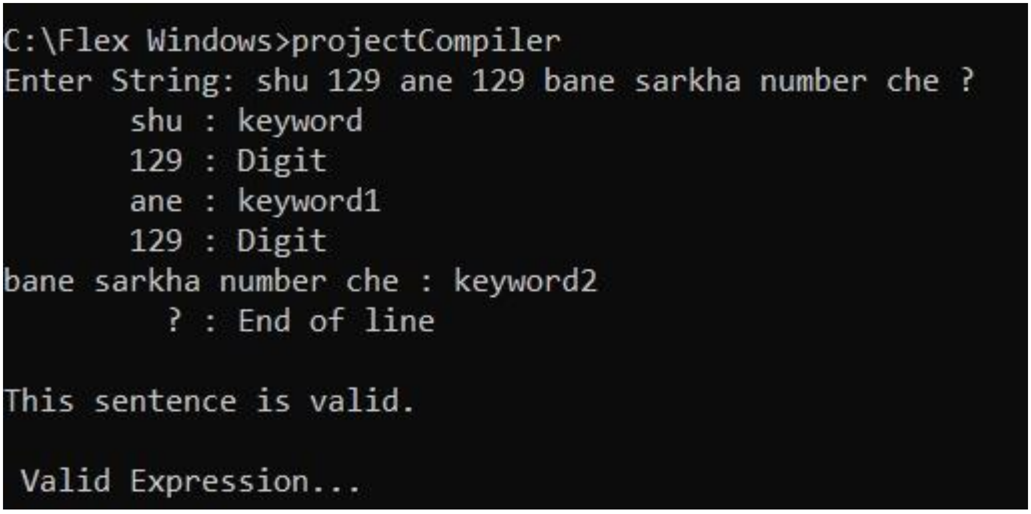
```
C:\Flex Windows>projectCompiler
Enter String: moto number kayo che 7 ke 2 ?
      moto : Operator
number kayo che : keyword5
         7 : Digit
        ke : keyword6
         2 : Digit
         ? : End of line

This sentence is valid.

 Valid Expression...
```

4) 32 , 34 mathi kayo number moto che ?



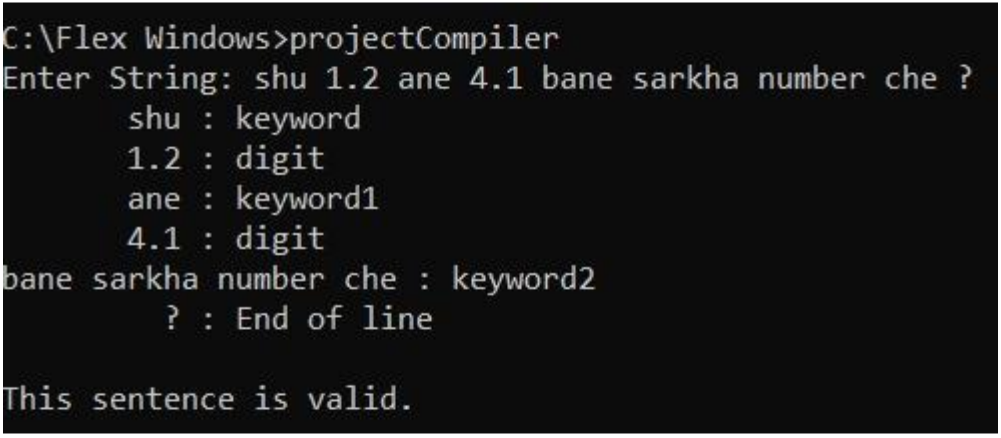
```
C:\Flex Windows>projectCompiler
Enter String: 32 , 34 mathi kayo number moto che ?
        32 : Digit
         , : Seperation
        34 : Digit
mathi kayo number : keyword3
      moto : Operator
       che : keyword4
         ? : End of line

This sentence is valid.

 Valid Expression...
```

5)shu 129 ane 129 bane sarkha number che?



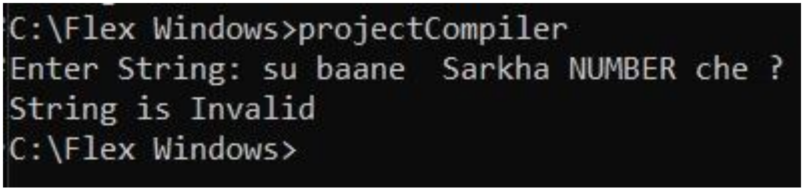
```
C:\Flex Windows>projectCompiler
Enter String: shu 129 ane 129 bane sarkha number che ?
       shu : keyword
       129 : Digit
       ane : keyword1
       129 : Digit
bane sarkha number che : keyword2
         ? : End of line

This sentence is valid.

 Valid Expression...
```

6)shu 1.2 ane 4.1 sarkha number che ?



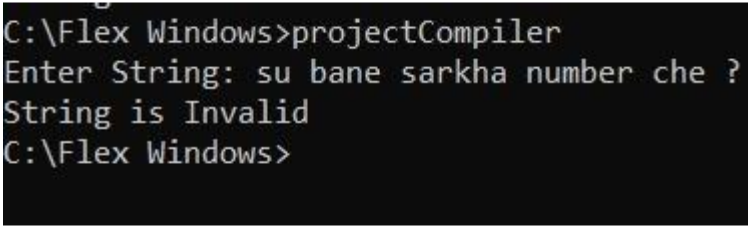
```
C:\Flex Windows>projectCompiler
Enter String: shu 1.2 ane 4.1 bane sarkha number che ?
       shu : keyword
       1.2 : digit
       ane : keyword1
       4.1 : digit
bane sarkha number che : keyword2
         ? : End of line

This sentence is valid.
```

7)Invalid input Check



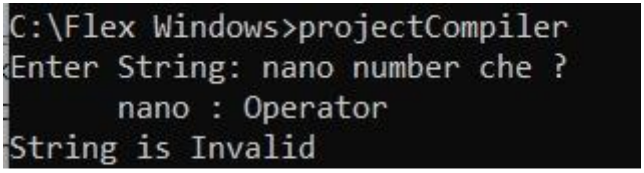
```
C:\Flex Windows>projectCompiler
Enter String: su baane  Sarkha NUMBER che ?
String is Invalid
C:\Flex Windows>
```

8)Invalid input Check



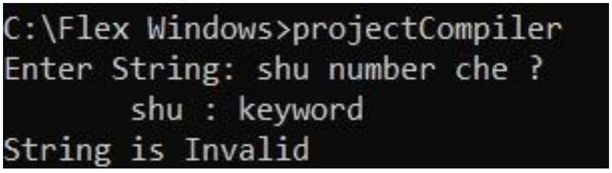
```
C:\Flex Windows>projectCompiler
Enter String: su bane sarkha number che ?
String is Invalid
C:\Flex Windows>
```

9)Invalid input Check



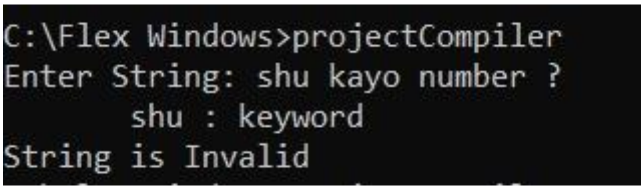
```
C:\Flex Windows>projectCompiler
Enter String: nano number che ?
       nano : Operator
String is Invalid
```

10)Invalid input Check



```
C:\Flex Windows>projectCompiler
Enter String: shu number che ?
        shu : keyword
String is Invalid
```

11)Invalid input Check



```
C:\Flex Windows>projectCompiler
Enter String: shu kayo number ?
        shu : keyword
String is Invalid
```

Chatper4- Conclusion

This project has been implemented from what we have learned in our college curriculum and many rich resources from the web. After doing this project we conclude that we have got more knowledge about how different compilers are working in practical world and also how various types of errors are handled.