Objectives:

1. To gather knowledge about Lexical Analysis
2. To get hands off practice of Flex Software
3. To Identify and categorize various language constructs
4. To learn to design our own language syntax of a programming language

Introduction:

Lexical analysis is also known as scanning, is the initial part of a compiler. Its primary purpose is to transform a high-level program into a sequence of tokens. These tokens are the smallest unit of a meaningful information in a programming language and serve as the building blocks for the subsequent phases of the compilation process. The main objectives of lexical analysis is to identify tokens, associate them with their corresponding lexemes (the original source code representations) and generate a token stream that can be used by compiler's parser and other components for further processing.

Basic Structure:

Data Types:

a) **Integer:**

Syntax: "cholok" {variable-name} = {format-specifier} "<<" "value"
Here,
cholok--> cholok is a keyword which used to declare a variable
variable-name--> Here any combination of digits or upper or lower case alphabets or underscore can be used to initialize a variable
= --> Here equal sign is a part of the syntax
format-specifier--> format specifier is the type of the variable which is declared for the integer case "intnum" is used.
<< --> It is also a part of the syntax which is used to set a value to the variable

value --> Any valid value assigned to this variable

b) **Float:**
Syntax: cholok a = floatnum << 10.2

c) **Long:**
Syntax: cholok b = longnum << 5

d) **Double:**
Syntax: cholok c = doublenum << 50.33

e) **Boolean:**
Syntax: cholok d = boolnum << true

f) **Character:**
Syntax: cholok e= charnum << 'h'

g) **String:**
Syntax: cholok f = stringnum << "development"

h) **Static Integer:**
Syntax: "Sthir " "cholok" {variable-name} = {format-specifier} << "value"
Here,
Sthir --> Keyword is used to declare a static variable and rest of the syntax is same as before
format-specifier--> format specifier is the type of the variable which is declared for the integer case "intnum" is used.

i) **Static Float:**
Syntax: Sthir cholok a= floatnum << 10.2

j) **Static Double:**
Syntax: Sthir cholok b = doublebnum<<10.8

k) **Static Long:**
Syntax: Sthir cholok e = longnum << 90

l) **Static Boolean:**
Syntax: Sthir cholok f = boolnum << false

m) **Static Character:**
Syntax: Sthir cholok g = charnum << ‘k’

n) **Static String:**
Syntax: Sthir cholok b = stringnum << “Turbo”

Operators:

Operators	Data Type	Type	Description	Syntax
+	Integer Float	Arithmetic	Adds two operands.	+(a,b)
-	Integer Float	Arithmetic	Subtracts second operand from the first.	-(a,b)
*	Integer Float	Arithmetic	Multiplies both operands.	*(a,b)
/	Integer Float	Arithmetic	Divides numerator by denumerator.	/(a,b)
%	Integer	Arithmetic	Modulus Operator and remainder of after an integer division.	%(a,b)

++	Integer	Arithmetic	Increment operator increases the integer value by one.	+(a,+)
--	Integer	Arithmetic	Decrement operator decreases the integer value by one.	-(a,-)
<=	Integer Float	Relational	True if the value of left operand is greater than or equal to the value of right operand.	<=(a,b)
>=	Integer Float	Relational	True if the value of left operand is less than or equal to the value of right operand.	>=(a,b)
>	Integer Float	Relational	True if the value of left operand is greater than the value of right operand.	>(a,b)
<	Integer Float	Relational	True if the value of left operand is less than the value	<(a,b)

			of right operand.	
==	Integer Float	Relational	True if the values of two operands are equal.	==(a,b)
!=	Integer Float	Relational	True if the values of two operands are not equal	!=(a,b)

Loop Statements:

1. For Loop

Syntax: forchokro [expression1, expression2, expression3] {

Statements

}

For example,

Suppose i is a loop control variable.
expression1: initial value of loop control variable (i << expression1)
expression2: upper bound of the loop control variable (<(i,expression2))
expression3: the value by which loop control variable will increment (i<< +(i ,expression3))

2.While Loop:

Syntax: whilechokro[condition/expression] {

@@ Statements

}

Conditional Statements:

IF-ELSE IF-ELSE

IF: Here in this language IF is used by “jodi” keyword
ELSE IF: Here in this language ELSE IF is replaced by “abarjodi” keyword
ELSE: Here in this language ELSE is replaced by “nahole” keyword

Some Keywords:

“*?” →	This symbol is used for Void keyword
“!!” →	This symbol is used for null keyword
“motamot” →	This symbol is used for return keyword
“thamo” →	This symbol is used for break keyword
“choltethako” →	This symbol is used for continue keyword
“somosto” →	This symbol is used for Public keyword
“goponio” →	This keyword is used for Private Keyword
“rokkhito” →	This keyword is used for Protected keyword

Comments:

Single Line Comment:

@@ for single line comment

Multi Line Comment:

@! This

Is a multiline

Comment.!@

Header File Inclusion:

“jog koro” [Header-file name.h]

Project Code:

```
%{
    #include<bits/stdc++.h>
    using namespace std;
    map<string,int>m2;//map to count variables;
    map<string,int>m1;//to count keywords and
    map<string,int>::iterator it;//to iterate through map container
  //To Check variable validity and variable count
  bool checkvarvalidity( string s)
  {
   string st; int i;
   //extracting the variable from string
   for(i=6;s[i]==' ';i++);
      for(;;i++){
         if(s[i]=='='||s[i]==' ')break;
         st+=s[i];
      }
      bool flag=0;
      m2[st]++;
      if(m2[st]>1)
      {
         cout<<"Problem is At line "<<s<<endl;
         cout<<"This Variable is Declared Before Please Choose another Variable"<<endl;
         m2[st]--;
         return flag;
      }
      else
      {
         cout<<"Variable Declared Successfully"<<endl;
         flag=1;
         return flag;

      }

  }


  //To Check operator related error like uninitialized value
  bool checkoperator(string t)
```

```
{
 bool flag=1;
 char c;
 string st;
 string s=t;
 for(int i=0;i<s.size();i++)
 {
    if((s[i]>='a'&& s[i]<='z')||(s[i]>='A'&&s[i]<='Z'))
    st+=s[i];
 }


 for(int i=0;i<st.size();i++)
 {
    char d=st[i];
    string ab(1,d);

    if(m2.find(ab)==m2.end())
    {
       flag=0;
       c=s[i];
       break;
    }
 }
 if(flag==0)
 {
    cout<<"There is a value "<<c<<" undeclared! Please declare this"<<endl;
    return flag;
 }
 return flag;

}
//to count the number of lines in the file
int num_lines=1;
%}
variable [A-Za-z_]+[A-Za-z_0-9]*
Assignment {variable}[ ]*"<<"[ ]*({variable}|[0-9]+|[0-9]+[.][0-9]+|['].[']|["].*["])
Condition (=|!|<|>)"("({variable}|([0-9]+[.]*[0-9]*)),({variable}|([0-9]+[.]*[0-9]*))")"
post ("+"|-)"("({variable}|([0-9]+[.]*[0-9]*)),("+"|-)")"
pre ("+"|-)"("("+"|-),({variable}|([0-9]+[.]*[0-9]*))")"
%%
"cholok"[ ]+{variable}[=]"intnum"[ ]*"<<"[ ]*[-]?[0-9]+([.][0-9]+)?          {    cout<<endl;
                                                printf("Integer ");
                                                //check whether the variable is valid or not
                                                bool res=checkvarvalidity(yytext);
                                                //if valid the increase the count
```

```
                                                    if(res==1)
                                                    {
                                                    m1["cholok"]++;
                                                    m1["intnum"]++;
                                                    }
                                                    //checking if any wrong data is sent to this datatype
                                                    string t;
                                                    t=yytext;
                                                    for(int i=0;i<t.size();i++)
                                                    {
                                                    if(t[i]=='.')
                                                    {
                                                    cout<<endl;
                                                    cout<<"Warning! You have assign improper data to
variable"<<endl;
                                                    cout<<endl;
                                                    cout<<yytext<<endl;
                                                    cout<<"Issue Arised!"<<endl;
                                                    cout<<endl;
                                                    break;
                                                    }
                                                    }
                                                }

"cholok"[ ]+{variable}[=]"longnum"[ ]*"<<"[ ]*[-]?[0-9]+([.][0-9]+)?            {
                                                    cout<<endl;
                                                    printf("Long ");
                                                    //check whether the variable the valid or not
                                                    bool res=checkvarvalidity(yytext);
                                                    if(res==1)
                                                    {
                                                    m1["cholok"]++;
                                                    m1["longnum"]++;
                                                    }
                                                    //if valid increase the count
                                                    string t;
                                                    t=yytext;
                                                    //to check whether the data contains improper value or
not
                                                    for(int i=0;i<t.size();i++)
                                                    {
                                                    if(t[i]=='.')
                                                    {
                                                    cout<<endl;
                                                    cout<<"Warning! You have assign improper data to this
variable"<<endl;
```

```
                cout<<yytext<<endl;
                cout<<"Issue Arised!"<<endl;
                cout<<endl;
                break;
                }
                }
            }

"cholok"[ ]+{variable}[=]"floatnum"[ ]*"<<"[ ]*[-]?[0-9]+([.][0-9]+)?          {
                cout<<endl;
                printf("Float ");
                //check whether the variable the valid or not
                bool res=checkvarvalidity(yytext);
                if(res==1)
                {
                m1["cholok"]++;
                m1["floatnum"]++;
                }

            }


"cholok"[ ]+{variable}[=]"doublenum"[ ]*"<<"[ ]*[-]?[0-9]+[.][0-9]+            {
                cout<<endl;
                printf("Double ");
                //check whether the variable the valid or not
                bool res=checkvarvalidity(yytext);
                if(res==1)
                {
                m1["cholok"]++;
                m1["doublenum"]++;
                }
            }

"cholok"[ ]+{variable}[=]"boolnum"[ ]*"<<"[ ]*(true|false|True|False)          {   cout<<endl;
                printf("Boolean ");

                //check whether the variable the valid or not
                bool res=checkvarvalidity(yytext);
                if(res==1)
                {
                m1["cholok"]++;
                m1["boolnum"]++;
                }

            }
```

```
"cholok"[ ]+{variable}[=]"charnum"[ ]*"<<"[ ]*['].[']                    {
                                                    printf("Character ");
                                                    //check whether the variable the valid or not
                                                    bool res=checkvarvalidity(yytext);
                                                    if(res==1)
                                                    {
                                                    m1["cholok"]++;
                                                    m1["charnum"]++;
                                                    }
                                                }
"cholok"[ ]+{variable}[=]"stringnum"[ ]*"<<"[ ]*["].*["]                    {    cout<<endl;
                                                    printf("String ");
                                                    bool res=checkvarvalidity(yytext);
                                                    if(res==1)
                                                    {
                                                    m1["cholok"]++;
                                                    m1["stringnum"]++;
                                                    }

                                                }

"sthir ""cholok"[ ]+{variable}[=]"intnum"[ ]*"<<"[ ]*[-]?[0-9]+                {   cout<<endl;
                                                    printf("Static Integer ");
                                                    //check whether the variable is valid or not
                                                    bool res=checkvarvalidity(yytext);
                                                    if(res==1)
                                                    {
                                                    m1["sthir cholok"]++;
                                                    m1["intnum"]++;
                                                    }
                                                    //if valid the increase the count
                                                    string t;
                                                    t=yytext;
                                                    for(int i=0;i<t.size();i++)
                                                    {
                                                    if(t[i]=='.')
                                                    {
                                                    cout<<endl;
                                                    cout<<"Warning! You have assign improper data to
variable"<<endl;
                                                    cout<<endl;
                                                    cout<<yytext<<endl;
                                                    cout<<"Issue Arised!"<<endl;
                                                    cout<<endl;
                                                    break;
                                                    }
```

```
                                                }
                                            }

"sthir ""cholok"[ ]+{variable}[=]"longnum"[ ]*"<<"[ ]*[-]?[0-9]+                {   cout<<endl;
                                                printf("Static Long ");
                                                //check whether the variable is valid or not
                                                bool res=checkvarvalidity(yytext);
                                                if(res==1)
                                                {
                                                m1["sthir cholok"]++;
                                                m1["longnum"]++;
                                                }
                                                //if valid the increase the count
                                                string t;
                                                t=yytext;
                                                for(int i=0;i<t.size();i++)
                                                {
                                                if(t[i]=='.')
                                                {
                                                cout<<endl;
                                                cout<<"Warning! You have assign improper data to
variable"<<endl;
                                                cout<<endl;
                                                cout<<yytext<<endl;
                                                cout<<"Issue Arised!"<<endl;
                                                cout<<endl;
                                                break;
                                                }
                                                }
                                            }


"sthir ""cholok"[ ]+{variable}[=]"floatnum"[ ]*"<<"[ ]*[-]?[0-9]+[.][0-9]+         {   cout<<endl;
                                                printf("Static Long ");
                                                //check whether the variable is valid or not
                                                bool res=checkvarvalidity(yytext);
                                                if(res==1)
                                                {
                                                m1["sthir cholok"]++;
                                                m1["floatnum"]++;
                                                }
                                                //if valid the increase the count

                                            }

"sthir ""cholok"[ ]+{variable}[=]"doublenum"[ ]*"<<"[ ]*[-]?[0-9]+[.][0-9]+       {
```

```
                                                cout<<endl;
                                                printf("Static Double ");
                                                //check whether the variable the valid or not
                                                bool res=checkvarvalidity(yytext);
                                                if(res==1)
                                                {
                                                m1["sthir cholok"]++;
                                                m1["doublenum"]++;
                                                }

                                              }

"sthir ""cholok"[ ]+{variable}[=]"boolnum"[ ]*"<<"[ ]*(true|false|True|False)    {    cout<<endl;
                                                printf("Static Boolean ");

                                                //check whether the variable the valid or not
                                                bool res=checkvarvalidity(yytext);
                                                if(res==1)
                                                {
                                                m1["sthir cholok"]++;
                                                m1["boolnum"]++;
                                                }

                                                }

"sthir ""cholok"[ ]+{variable}[=]"charnum"[ ]*"<<"[ ]*['].['] {
                                                printf("Static Character ");
                                                //check whether the variable the valid or not
                                                bool res=checkvarvalidity(yytext);
                                                if(res==1)
                                                {
                                                m1["sthir cholok"]++;
                                                m1["charnum"]++;
                                                }
                                        }

"sthir ""cholok"[ ]+{variable}[=]"stringnum"[ ]*"<<"[ ]*["].*["]    {    cout<<endl;
                                                printf("Static String ");
                                                bool res=checkvarvalidity(yytext);
                                                if(res==1)
                                                {
                                                m1["sthir cholok"]++;
                                                m1["stringnum"]++;
                                                }

                                                }
```

```
{variable}[ ]*"<<"[ ]*"+("({variable}|([0-9]+[.]*[0-9]*)),({variable}|([0-9]+[.]*[0-9]*))")"      {
                                                    if(checkoperator(yytext))
                                                    {
                                                        printf("Summation Command Executed\n");
                                                        m1["Summation Command"]++;

                                                    }
                                                   }


{variable}[ ]*"<<"[ ]*"-("({variable}|([0-9]+[.]*[0-9]*)),({variable}|([0-9]+[.]*[0-9]*))")"     {
                                                    if(checkoperator(yytext))
                                                    {
                                                        printf("Subtraction Command Executed\n");
                                                        m1["Subtaction Command"]++;

                                                    }
                                                   }

{variable}[ ]*"<<"[ ]*"*("({variable}|([0-9]+[.]*[0-9]*)),({variable}|([0-9]+[.]*[0-9]*))")"        {
                                                    if(checkoperator(yytext))
                                                    {
                                                        printf("Multiplication Command Executed\n");
                                                        m1["Multiplication Command"]++;

                                                    }
                                                   }

{variable}[ ]*"<<"[ ]*"/("({variable}|([0-9]+[.]*[0-9]*)),({variable}|([0-9]+[.]*[0-9]*))")"       {
                                                    if(checkoperator(yytext))
                                                    {
                                                        printf("Division Command Executed\n");
                                                        m1["Division Command"]++;

                                                    }
                                                   }

{variable}[ ]*"<<"[ ]*"%("({variable}|([0-9]+[.]*[0-9]*)),({variable}|([0-9]+[.]*[0-9]*))")"       {
                                                    if(checkoperator(yytext))
                                                    {
                                                        printf("Modulus Command Executed\n");
                                                        m1["Modulus Command"]++;

                                                    }
```

```
                                                    }

"=("({variable}|([0-9]+[.]*[0-9]*)),({variable}|([0-9]+[.]*[0-9]*))")"                  {
                                                     if(checkoperator(yytext))
                                                     {
                                                         printf("Equal Command Executed\n");
                                                         m1["Equal Command"]++;

                                                     }
                                                    }

"!("({variable}|([0-9]+[.]*[0-9]*)),({variable}|([0-9]+[.]*[0-9]*))")"                  {
                                                     if(checkoperator(yytext))
                                                     {
                                                         printf("Equal Command Executed\n");
                                                         m1["Equal Command"]++;

                                                     }
                                                    }

"<("({variable}|([0-9]+[.]*[0-9]*)),({variable}|([0-9]+[.]*[0-9]*))")"                  {
                                                     if(checkoperator(yytext))
                                                     {
                                                         printf("Smaller Than Command Executed\n");
                                                         m1["Smaller Than Command"]++;

                                                     }
                                                    }

">("({variable}|([0-9]+[.]*[0-9]*)),({variable}|([0-9]+[.]*[0-9]*))")"                   {
                                                     if(checkoperator(yytext))
                                                     {
                                                         printf("Greater Than Command Executed\n");
                                                         m1["Greater Than Command"]++;

                                                     }
                                                    }

"+("({variable}|([0-9]+[.]*[0-9]*)),"+)"                                    {
                                                     if(checkoperator(yytext))
                                                     {
                                                         printf("Post Increment Command Executed\n");
                                                         m1["Post Increment Command"]++;

                                                     }
                                                    }
```

```
"+(+",({variable}|([0-9]+[.]*[0-9]*))")"                    {
                                                             if(checkoperator(yytext))
                                                             {
                                                                 printf("Pre Increment Command Executed\n");
                                                                 m1["Pre Increment Command"]++;

                                                             }
                                                            }

"-("({variable}|([0-9]+[.]*[0-9]*)),"-)"                    {
                                                             if(checkoperator(yytext))
                                                             {
                                                                 printf("Post decrement Command Executed\n");
                                                                 m1["Post decrement Command"]++;

                                                             }
                                                            }

"-(-",({variable}|([0-9]+[.]*[0-9]*))")"                    {
                                                             if(checkoperator(yytext))
                                                             {
                                                                 printf("Pre decrement Command Executed\n");
                                                                 m1["Pre decrement Command"]++;

                                                             }
                                                            }

{variable}[ ]*"<<"[ ]*({variable}|[0-9]+|[0-9]+[.][0-9]+|['].[']|["].*["])"\n"          {
                                                             if(checkoperator(yytext))
                                                             {
                                                                 printf("Assignment Command Executed\n");
                                                                 m1["Assignment Command"]++;

                                                             }
                                                            }

"forchokro""["[ ]*{Assignment}[ ]*,[ ]*{Condition}[ ]*,[ ]*({post}|{pre})[ ]*"]"[ ]*[\n]*"{"[ ]*[\n]*.*[
]*[\n]*"}"      {printf("For Loop Declared\n");
                                                                 string s;
                                                                 s=yytext;
                                                                 for(int i=0;i<s.size();i++)
                                                                 {
                                                                     if(s[i]=='\n')num_lines++;
                                                                 }
                                                                 num_lines++;
```

```
                                                        m1["For Loop"]++;}

"whilechokro""["[ ]*{Condition}[ ]*"]"[ ]*[\n]*"{"[ ]*[\n]*.*[ ]*[\n]*"}"
{printf("While Loop Declared\n");
                                                            string s;
                                                        s=yytext;
                                                        for(int i=0;i<s.size();i++)
                                                        {
                                                            if(s[i]=='\n')num_lines++;
                                                        }
                                                        num_lines++;
                                                         m1["While Loop"]++;}

("jodi""["[ ]*{Condition}[ ]*"]"[ ]*[\n]*"{"[ ]*[\n]*.*[ ]*[\n]*"}""\n")("abarjodi""["[ ]*{Condition}[ ]*"]"[
]*[\n]*"{"[ ]*[\n]*.*[ ]*[\n]*"}""\n")*("nahole"[ ]*[\n]*"{"[ ]*[\n]*.*[ ]*[\n]*"}""\n")*          {printf("If Else
Declared\n");

string s;

s=yytext;

for(int i=0;i<s.size();i++)

{

if(s[i]=='\n')num_lines++;

}
num_lines++;

}

"*?" {printf("Void\n");m1["Void"]++;}

"!!" {printf("Null\n");m1["null"]++;}

"motamot" {printf("return\n");m1["return"]++;}

"thamo" {printf("break\n");m1["break"]++;}

"choltethako" {printf("continue\n");m1["continue"]++;}

"somosto" {printf("Public\n");m1["Public"]++;}

"goponio" {printf("Private\n");m1["Private"]++;}
```

```
"rokkhito" {printf("Protected\n");m1["Protected"]++;}

"jog koro"[ ]*["[a-zA-Z0-9]+[.][h]"]"  {printf("Header File Added\n");m1["header file"]++;}

"@@".* {printf("Single Line Comment Found!\n");m1["comment"]++;}

"@!"({variable}|(\n)*|[ ]*)*"!@" {printf("Multi Line Comment Found\n");m1["comment"]++;}
\n {num_lines++;
.+ {printf("SYNTAX ERROR\n");
   cout<<"Problem is at : "<<yytext<<" this line!"<<endl;}

%%
int yywrap(){
          return 1;
}
int main(){
   yyin = fopen("finaldemo.txt","r");
   yylex();
   cout<<endl;
   cout<<endl;
   cout<<endl;

   m2.erase("cholok");
   m1.erase("Operator Command");
   printf("To Check Keyword Count: \n\n");

   for(it=m1.begin();it!=m1.end();it++)
   {
       cout<<"Key-> "<<(*it).first<<" Count-> "<<(*it).second<<endl;
   }

   cout<<endl;
   cout<<"To check Variable Count: \n\n";

   for(it=m2.begin();it!=m2.end();it++)
   {
      cout<<"Variable-> "<<(*it).first<<" Count-> "<<(*it).second<<endl;
   }
   cout<<endl;

   cout<<"Total Number of lines: "<<num_lines<<endl;
   cout<<"All Status Checked!"<<endl;
          return 0;
}
```

Input:
Input taken from finaldemo.txt file:

```
cholok a=intnum<<-1

cholok b=longnum<<10

cholok c=longnum<<10.4

cholok d=floatnum<<5.8

cholok e=boolnum<<True

cholok f=charnum<<' '

cholok g=stringnum<<"Mahdi"

sthir cholok f=intnum<<9

i<<+(a,1)

a<<+(a,b)

forchokro[a<<3, <(a,20), +(a,+)]
{
  a<<+(a,b)
  b<<+(a,b)
}

whilechokro[ <(a,20) ]{

  a<<%(a,b)
  b<</(c,d)

}

jodi[<(a,20)]{

}
abarjodi[ <(b,10)]{

}
*?
!!
motamot
```

thamo
choltethako

@!
This is
a multiline Comment
!@

@@ This is a single line comment

Qwe

Output:

Taken From Terminal:
Integer Variable Declared Successfully

Long Variable Declared Successfully

Long Variable Declared Successfully

Warning! You have assign improper data to this variable
cholok c=longnum<<10.4
Issue Arised!

Float Variable Declared Successfully

Boolean Variable Declared Successfully
Character Variable Declared Successfully

String Variable Declared Successfully

Static Integer Variable Declared Successfully
There is a value i undeclared! Please declare this
Summation Command Executed
For Loop Declared
Void
Null
return
break
continue

Multi Line Comment Found
Single Line Comment Found!
SYNTAX ERROR
Problem is at : qwe this line!

To Check Keyword Count:

Key-> For Loop Count-> 1
Key-> Summation Command Count-> 1
Key-> Void Count-> 1
Key-> boolnum Count-> 1
Key-> break Count-> 1
Key-> charnum Count-> 1
Key-> cholok Count-> 7
Key-> comment Count-> 2
Key-> continue Count-> 1
Key-> floatnum Count-> 1
Key-> intnum Count-> 2
Key-> longnum Count-> 2
Key-> null Count-> 1
Key-> return Count-> 1
Key-> sthir cholok Count-> 1
Key-> stringnum Count-> 1

To check Variable Count:

Variable-> a Count-> 1
Variable-> b Count-> 1
Variable-> c Count-> 1
Variable-> d Count-> 1
Variable-> e Count-> 1
Variable-> f Count-> 1
Variable-> g Count-> 1

Total Number of lines: 36
All Status Checked!

Discussion:

After doing this assignment, knowledge has gained about how to design a custom defined language by defining new variables, new keywords and new tokens. Moreover, by completing this assignment we can now design any basic language

and implement its functionality using lexical analysis.

Conclusion:

So, The first step of building a compiler is lexical analysis and this assignment helps us to gather practical knowledge and sharpen our development skill.

References:

1. https://www.geeksforgeeks.org/flex-fast-lexical-analyzer-generator/
2. https://regex101.com/