Project Report
on

Compiler for
<< Time Converter>>

Developed by

Chaudhary Hetavi - IT013 – 19ITUOS002

Chauhan Preet - IT014 – 19ITUBS145

Chauhan Priya - IT015 – 19ITUBF019

Guided By:

Prof. Nikita P. Desai
Dept. of Information Technology

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

DHARMSINH DESAI UNIVERSITY

NADIAD-387001, GUJARAT

CERTIFICATE

This is to certify that the project entitled “**Compiler for Time Converter”** is a bonafied report of the work carried out by

1) Miss. Chaudhary Hetavi, Student ID No: 19ITUOS002
2) Mr. Chauhan Preet, Student ID No: 19ITUBS145
3) Miss. Chauhan Priya, Student ID No: 19ITUBF019

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

1.0 Introduction

2.0 Lexical phase design

3.0 Syntax analyzer design

1.0 INTRODUCTION

1.0.1 Project Details

Language Name: Time Coverter **Language description:**

Write an appropriate language description for a layman language which can do Time Conversation.

Example of valid program in this language:

- 5 hour aetle ketla minute?
- 10 hour na ketla second?

1.0.2 Project Planning

List of Students with their Roles/Responsibilities:

Chaudhary Hetavi: DFA Design, Algorithm Design, Final Report.

Chauhan Preet: Regular Expression, Grammar rules, Final Report.

Chauhan Priya: Scanner phase implementation**,** YACC implementation Final Report.

2.0 LEXICAL PHASE DESIGN

2.0.1 Regular Expression:

Keywords:

RE	**Token**
atle	atle
ketla	ketla
na	na

Operations:

RE	**Token**
hour	hour
minute	minute
second	second
millisecond	milisecond

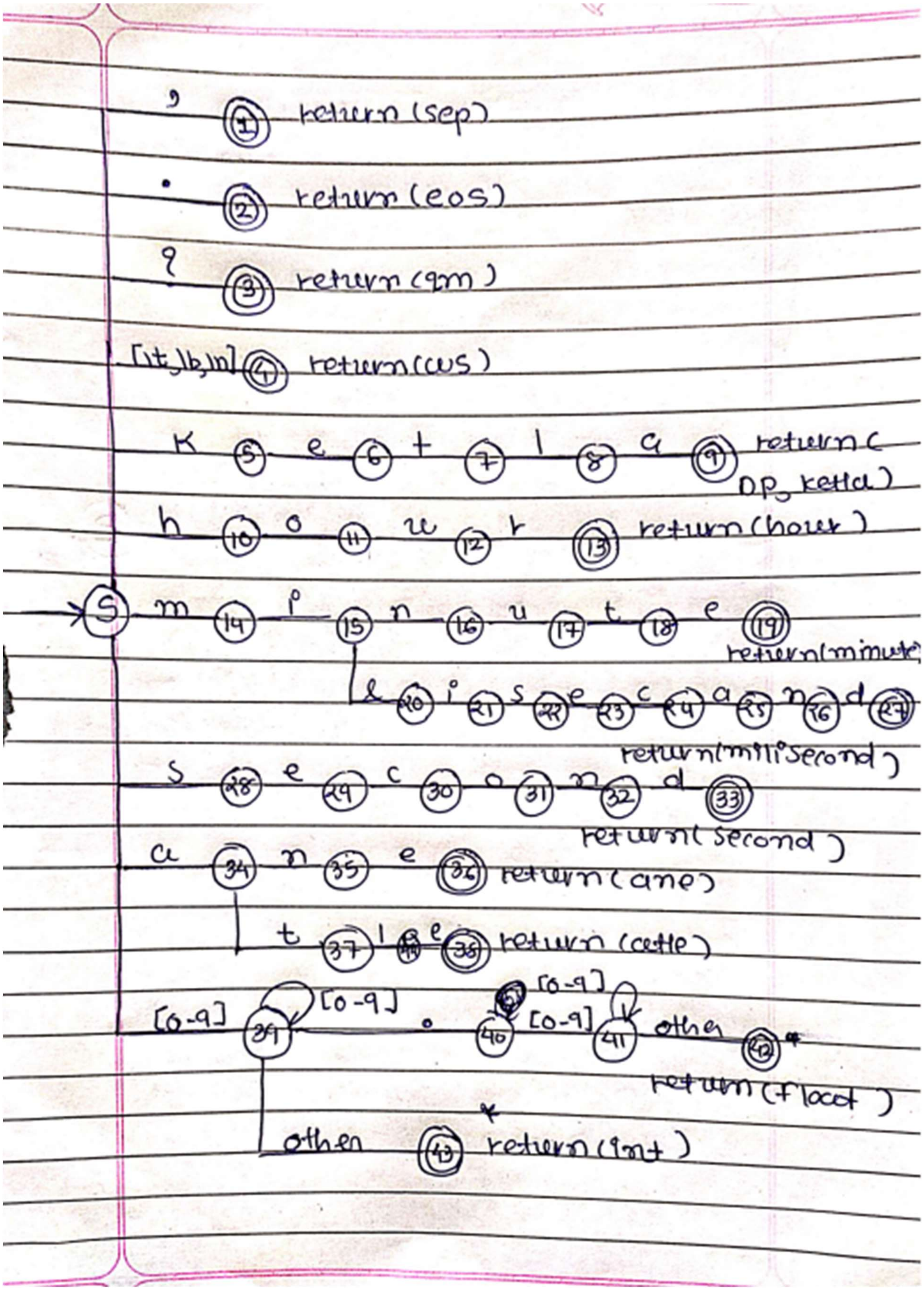
Values type: int and float

RE	**Token**
[0-9]+	int
[0-9]+(.[0-9]+)	float

Delimiters : {. , ? blanks}

RE	**Token**
?	qm
blanks	ws

2.0.2 Deterministic Finite Automata design for lexer



2.0.3 Algorithm of lexer

```
lexer
{

    int c=0;
    bool f=false;
    int len=string.length();
    while not eof do
    {

        state="S";
        while not eof do(c<len)
        {

            if(f)
            {

                f=false;

            }
            char ch=nextchar();
            switch(state)
            {

            case state of "S":

                case state of ",":
                    state="1";
                    ch=nextchar();
                    f-true;
                    break;

                case state of ".":
                    state="2";
                    ch=nextchar();
                    f-true;
                    break;
```

```
case state of "?":
   state="3";
   ch=nextchar();
   f-true;
   break;

case state of "[\t,\b,\n]":
   state="4";
   ch=nextchar();
   f-true;
   break;

case state of 'k':
   state="5";
   ch=nextchar();
   break;

case state of 'h':
   state="10";
   ch=nextchar();
   break;

case state of "m":
   state="14";
   ch=nextchar();
   break;

case state of 's':
   state="28";
   ch=nextchar();
   break;

case state of 'a':
   state="35";
   ch=nextchar();
   break;

case state of '[0-9]':
   state="39";
   ch=nextchar();
   break;
default:
   f=true

case state of "5":
   case state of 'e':
       state="6";
       ch=nextchar();

case state of "6":
   case state of 't':
       state="7";
       ch=nextchar();

case state of "7":
   case state of 'l':
       state="8";
       ch=nextchar();

case state of "8":
   case state of 'a':
       state="9";
       ch=nextchar();

       f=true;
case state of "10":
   case state of 'o':
       state="11";
       ch=nextchar();

case state of "11":
   case state of 'u':
       state="12";
       ch=nextchar();

case state of "12":
   case state of 'r':
       state="13";

       ch=nextchar();
       f=true;

case state of "14":
   case state of 'i':
       state="15";
       ch=nextchar();

case state of "15":
   case state of 'n':
       state="16";
       ch=nextchar();
       break;

   case state of 'l':
       state="20";
       ch=nextchar();
       break;

case state of "16":
   case state of 'u':
       state="17";
       ch=nextchar();

case state of "17":
   case state of 't':
       state="18";
       ch=nextchar();

case state of "18":
   case state of 'e':
       state="19";
       ch=nextchar();
       f=true;

case state of "20":
   case state of 'i':
       state="21";
       ch=nextchar();

case state of "21":
   case state of 's':
       state="22";
       ch=nextchar();

case state of "22":
   case state of 'e':
       state="23";
       ch=nextchar();

case state of "23":
   case state of 'c':
       state="24";
       ch=nextchar();

case state of "24":
   case state of 'a':
       state="25";
       ch=nextchar();

case state of "25":
   case state of 'n':
       state="26";
       ch=nextchar();

case state of "26":
   case state of 'd':
       state="27";
       ch=nextchar();

       f=true;
case state of "28":
   case state of 'e':
       state="29";
       ch=nextchar();
```

```
case state of "29":
    case state of 'c':
        state="30";

        ch=nextchar();

case state of "30":
    case state of 'a':
        state="31";
        ch=nextchar();

case state of "31":
    case state of 'n':
        state="32";
        ch=nextchar();

case state of "32":
    case state of 'd':
        state="33";
        ch=nextchar();
        f=true;

case state of "34":
    case state of 'n':
        state="35";
        ch=nextchar();
        break;

    case state of 't':
        state="37";
        ch=nextchar();
        break;

case state of "35":
    case state of 'e':
        state="36";
        ch=nextchar();
        f=true;
```

```
case state of "37":
    case state of 'l':
        state="44";
        ch=nextchar();

case state of "44":
    case state of 'e':
        state="38";
        ch=nextchar();
        f=true;

case state og "39":
    case state of '[0-9]':
        ch=nextchar();
        break;

    case state of '.':
        state="40";
        ch=nextchar();
        break;

    default:
        state="43";
        f=true;

case state og "40":

    case state of '[0-9]':
        state="41":
        ch=nextchar();
        break;

    default:
        f=true;

case state og "41":

    case state of '[0-9]':
        ch=nextchar();
```

```
                  break;

               default:
                  state="42":
                  f=true;

            }
         }

         case state of "13"|"19"|"27"|"33"|"36"|"38":
            print("keyword");

         case state of "42":
            print("float");

         case state of "43":
            print("int");

         case state of "9":
            print("operator");

         case state of "1":
            print("sep");

         case state of "2":
            print("eos");

         case state of "3":
            print("question tag");

         case state of "4":
            print("ws");

         default:
            print("Invalid Input");
            ch=nextchar();
            end case;

   }
}
```

2.0.4 Implementation of lexer

Flex Program:

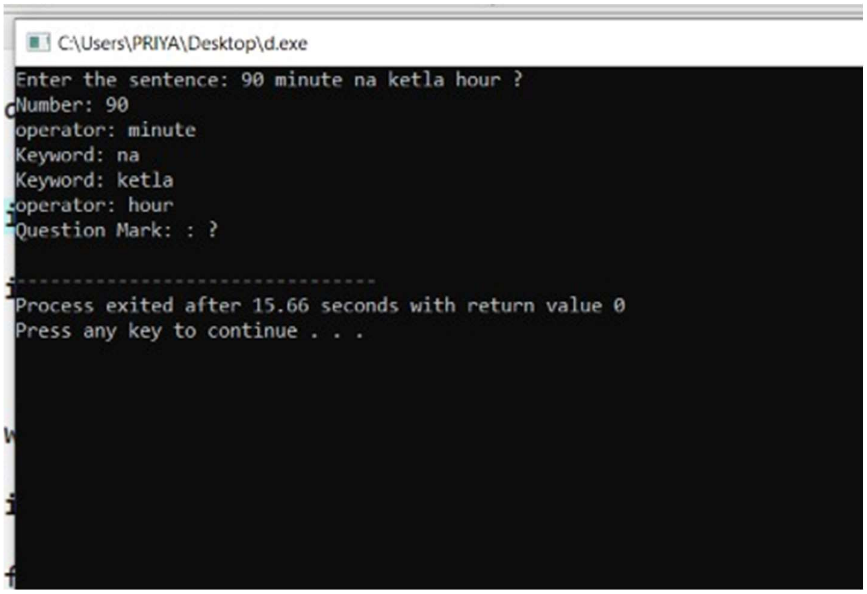
```
%{
#include<stdio.h>
%}
Keyword "ketla"|"atle"|"na"
Op "second"|"minute"|"hour"|"milisecond"
Digit [0-9]
Int {Digit}+
qm "?"
ws " "

%%
{Keyword} {printf("Keyword - %s\n",yytext);}
{Op} {printf("Operator - %s\n",yytext);}
{Int} {printf("Integer - %s\n",yytext);}

{qm} {printf("que tag - %s\n",yytext);}
{ws} {printf("ws \n",yytext);}
. {printf("%s is not a valid token\n",yytext);}
%%
int yywrap(){return 1;}
int main()
{
yylex();
return 0;
}
```

2.0.5 Scanner phase implementation in "C++" language

```
#include<bits/stdc++.h>
#include<iostream>
using namespace std;
bool isNumber(string s)
{
    if (s.size() == 0)
        return false;
    for (int i = 0; i < s.size(); i++)
    {
        if ((s[i] >= '0' && s[i] <= '9') == false)
        {
            return false;
        }

    }
    return true;
}

int main()
{
    string keywords[3] = {"ketla","atle","na" };
    string operators[4] = {"hour", "minute", "second", "milisecond"};
    string qm="?" ;
    string str;
    bool found;
    cout<<"Enter the sentence: ";
    getline(cin, str);
    istringstream iss(str);
    string word;
    while (iss >> word)
    {
        found=false;
        for (int j = 0; j < 3; j++)
        {
            if (word.compare(keywords[j]) == 0)
            {
                cout << "Keyword: " << keywords[j] << " \n";
                found=true;
            }
        }
        for (int k = 0; k < 4; k++)
        {
            if (word.compare(operators[k]) == 0)
            {
                cout << "operator: " << operators[k] << " \n";
                found=true;
            }
        }
        if (isNumber(word))
        {
            cout << "Number: " << word << " \n";
```

```
                continue;
            }


            if (word == qm)
            {
                cout << "Question Mark: : " << qm << " \n";
                continue;
            }
            if(!found)
            {
                cout<<"Error: "<<word<<"\n";
                break;
            }
        }
    }
```

Output:



```
Enter the sentence: 90 minute na ketla hour ?
Number: 90
operator: minute
Keyword: na
Keyword: ketla
operator: hour
Question Mark: : ?

--------------------------------
Process exited after 15.66 seconds with return value 0
Press any key to continue . . .
```

2.0.6 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 FORGnuWin32*/

- After successful installation Goto Cfolder
- 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**

-

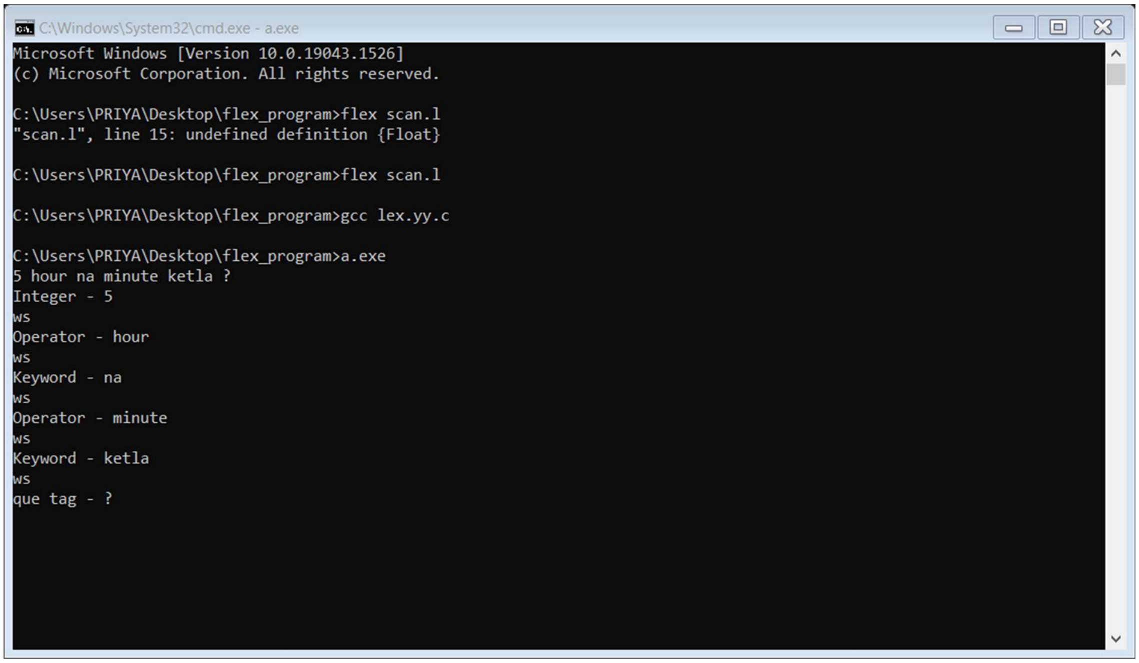
Step 6

- Finished

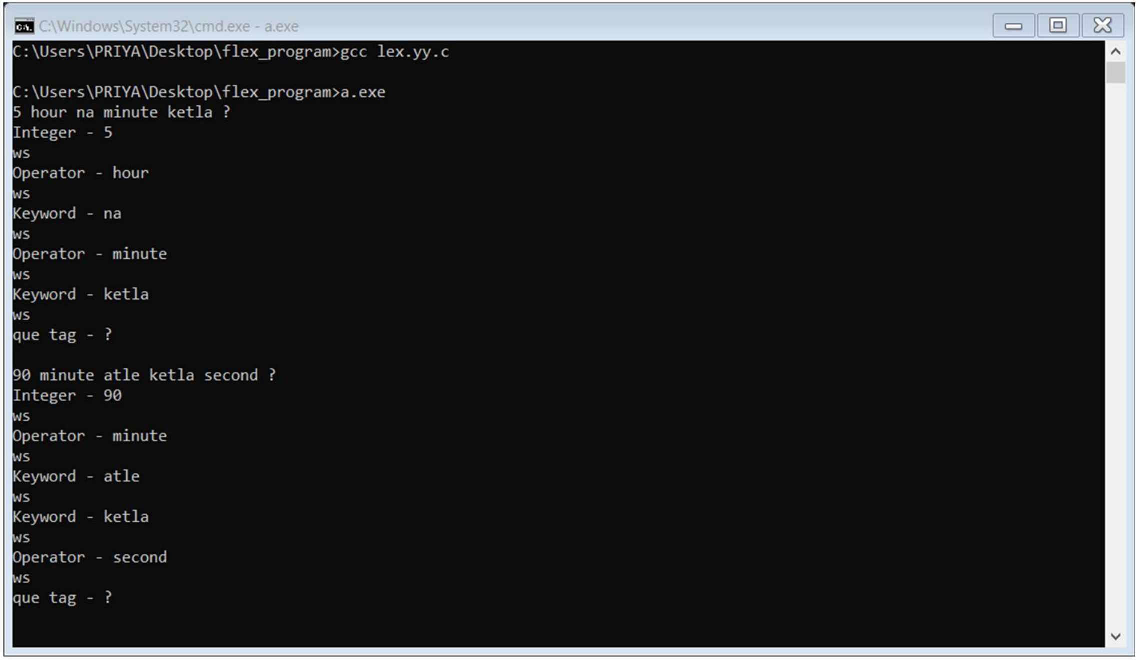
-

2.0.7 Output screenshots of lexer:

Input/output for valid Inputs:



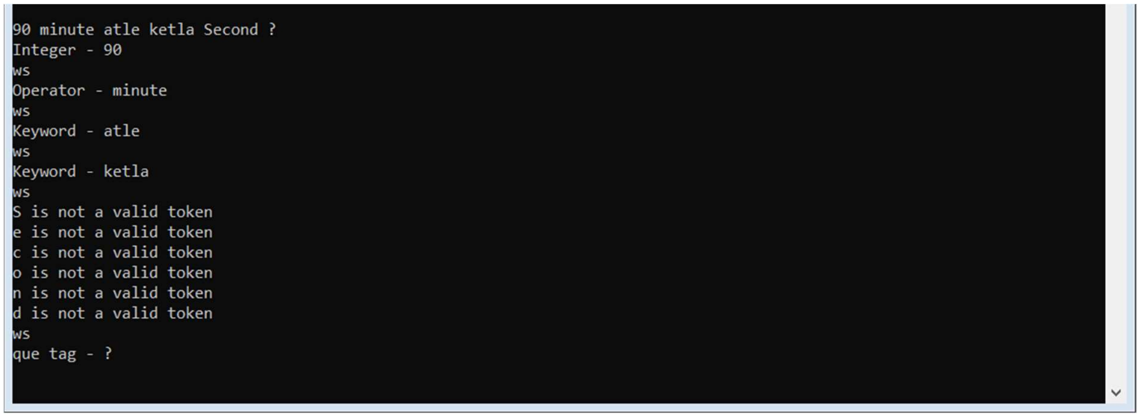
```
Microsoft Windows [Version 10.0.19043.1526]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PRIYA\Desktop\flex_program>flex scan.l
"scan.l", line 15: undefined definition {Float}

C:\Users\PRIYA\Desktop\flex_program>flex scan.l

C:\Users\PRIYA\Desktop\flex_program>gcc lex.yy.c

C:\Users\PRIYA\Desktop\flex_program>a.exe
5 hour na minute ketla ?
Integer - 5
ws
Operator - hour
ws
Keyword - na
ws
Operator - minute
ws
Keyword - ketla
ws
que tag - ?
```



```
C:\Users\PRIYA\Desktop\flex_program>gcc lex.yy.c

C:\Users\PRIYA\Desktop\flex_program>a.exe
5 hour na minute ketla ?
Integer - 5
ws
Operator - hour
ws
Keyword - na
ws
Operator - minute
ws
Keyword - ketla
ws
que tag - ?

90 minute atle ketla second ?
Integer - 90
ws
Operator - minute
ws
Keyword - atle
ws
Keyword - ketla
ws
Operator - second
ws
que tag - ?
```

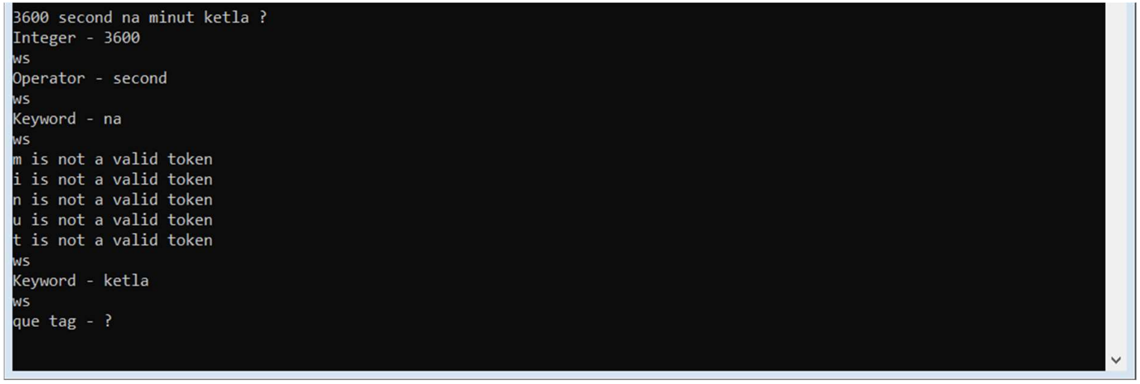
Input/output for invalid tokens:

1.Operation starting with capital letter:



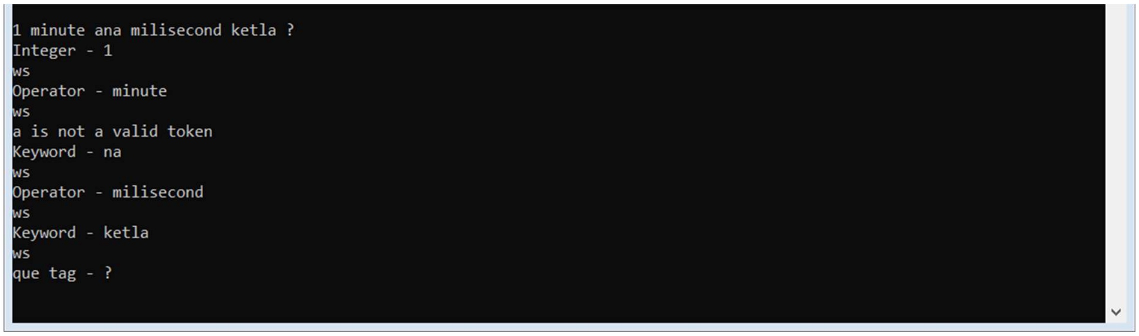
```
90 minute atle ketla Second ?
Integer - 90
ws
Operator - minute
ws
Keyword - atle
ws
Keyword - ketla
ws
S is not a valid token
e is not a valid token
c is not a valid token
o is not a valid token
n is not a valid token
d is not a valid token
ws
que tag - ?
```

2.Operation is invalid:



```
3600 second na minut ketla ?
Integer - 3600
ws
Operator - second
ws
Keyword - na
ws
m is not a valid token
i is not a valid token
n is not a valid token
u is not a valid token
t is not a valid token
ws
Keyword - ketla
ws
que tag - ?
```

3.Keyword is invalid:



```
1 minute ana milisecond ketla ?
Integer - 1
ws
Operator - minute
ws
a is not a valid token
Keyword - na
ws
Operator - milisecond
ws
Keyword - ketla
ws
que tag - ?
```

3.0 SYNTAX ANALYZER DESIGN

3.0.1 Grammar rules

S->DPDXQ | DXQ

X->AKO | NKO

D->digit

P->.

A->atle

K->ketla

O->hour | minute | second | millisecond

N->na

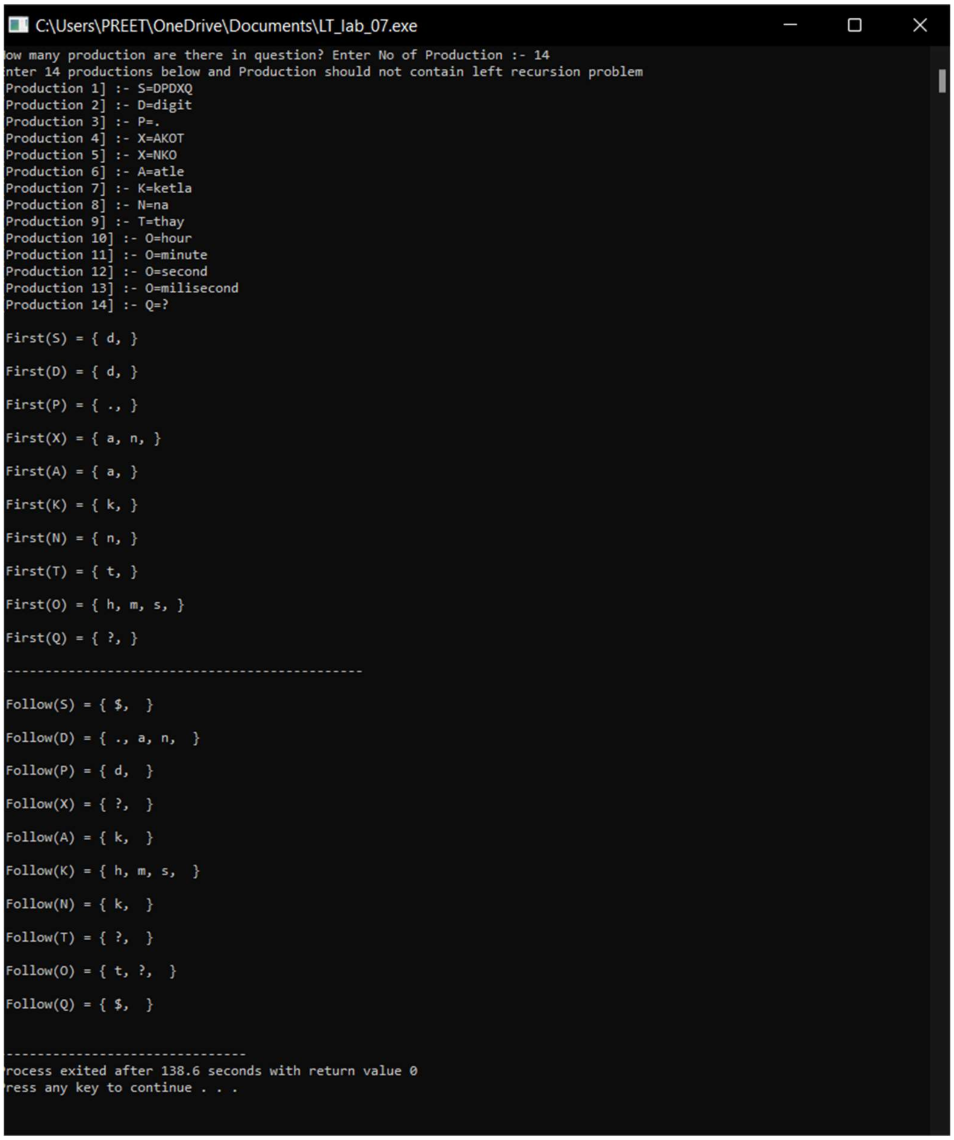
Q->?

First and follow of grammar:

```
#include<stdio.h>
#include<ctype.h>
#include<string.h>
void followfirst(char, int, int);
void follow(char c);
void findfirst(char, int, int);
int count, n = 0;
char calc_first[100][100];
char calc_follow[100][100];
int m = 0;
char production[100][100];
char f[100], first[100];
int k;
char ck;
int e;

int main(int argc, char ** argv)
{
    int jm = 0;
    int km = 0;
    int i, choice;
    char c, ch;
    printf("How many production are
there in question? Enter No of
Production :- ");
    scanf("%d", & count);

    printf("Enter %d productions below
and Production should not contain left
recursion problem\n", count);
    for ( i = 0; i < count; i++)
    {
        printf("[Production %d] :- ", (i +
1));
        scanf("%s", & production[i]);
    }
    int kay;
    char done[count];
    int ptr = -1;
    for (k = 0; k < count; k++)
    {
        for (kay = 0; kay < 100; kay++)
        {
            calc_first[k][kay] = '!';
        }
    }
    int point1 = 0, point2, xxx;
    for (k = 0; k < count; k++)
    {
        c = production[k][0];
        point2 = 0;
        xxx = 0;
        for (kay = 0; kay <= ptr; kay++)
            if (c == done[kay])
                xxx = 1;
        if (xxx == 1)
            continue;

        findfirst(c, 0, 0);
        ptr += 1;
        done[ptr] = c;
        printf("\n First(%c) = { ", c);
        calc_first[point1][point2++] = c;
        for (i = 0 + jm; i < n; i++)
        {
            int lark = 0, chk = 0;
            for (lark = 0; lark < point2;
lark++)
            {
                if (first[i] ==
calc_first[point1][lark])
                {
                    chk = 1;
                    break;
                }
            }
            if (chk == 0)
            {
                printf("%c, ", first[i]);
                calc_first[point1][point2++]
= first[i];
            }
        }
        printf("}\n");
        jm = n;
        point1++;
    }
    printf("\n");
    printf("-----------------------------------
------------\n\n");

    char donee[count];
    ptr = -1;
    for (k = 0; k < count; k++)
    {
        for (kay = 0; kay < 100; kay++)
        {
            calc_follow[k][kay] = '!';
        }
    }
    point1 = 0;
    int land = 0;
    for (e = 0; e < count; e++)
    {
        ck = production[e][0];
        point2 = 0;
        xxx = 0;
        for (kay = 0; kay <= ptr; kay++)
            if (ck == donee[kay])
                xxx = 1;
        if (xxx == 1)
            continue;
        land += 1;
        follow(ck);
        ptr += 1;
        donee[ptr] = ck;
        printf(" Follow(%c) = { ", ck);
        calc_follow[point1][point2++] =
ck;

        for (i = 0 + km; i < m; i++)
        {
            int lark = 0, chk = 0;
            for (lark = 0; lark < point2;
lark++)
            {
                if (f[i] ==
calc_follow[point1][lark])
                {
                    chk = 1;
                    break;
                }
            }
            if (chk == 0)
            {
                printf("%c, ", f[i]);

calc_follow[point1][point2++] = f[i];
            }
        }
        printf(" }\n\n");
        km = m;
        point1++;
    }
}
void follow(char c)
{
    int i, j;
    if (production[0][0] == c)
    {
        f[m++] = '$';
    }
    for (i = 0; i < 10; i++)
    {

        for (j = 2; j < 10; j++)
        {
            if (production[i][j] == c)
            {
                if (production[i][j + 1] !=
'\0')
                {

followfirst(production[i][j + 1], i, (j +
2));
                }
                if (production[i][j + 1] ==
'\0' && c != production[i][0])
                {
                    follow(production[i][0]);
                }
            }
        }
    }
}
void findfirst(char c, int q1, int q2)
{
    int j;
    if (!(isupper(c)))
    {
        first[n++] = c;
    }
    for (j = 0; j < count; j++)
    {
        if (production[j][0] == c)
        {
            if (production[j][2] == '#')
            {
                if (production[q1][q2] ==
'\0')
                    first[n++] = '#';
                else if (production[q1][q2]
!= '\0' &&
                         (q1 != 0 || q2 != 0))
                {


findfirst(production[q1][q2], q1, (q2 +
1));
                }
                else
                    first[n++] = '#';
            }
            else if
(!isupper(production[j][2]))
            {
                first[n++] =
production[j][2];
            }
            else
            {
                findfirst(production[j][2], j,
3);
            }
        }
    }
}
void followfirst(char c, int c1, int c2)
{
    int k;
    if (!(isupper(c)))
        f[m++] = c;
    else
    {
        int i = 0, j = 1;
        for (i = 0; i < count; i++)
        {
            if (calc_first[i][0] == c)
                break;
        }
        while (calc_first[i][j] != '!')
        {
            if (calc_first[i][j] != '#')
            {
                f[m++] = calc_first[i][j];

            }
            else
            {
                if (production[c1][c2] ==
'\0')
                {

follow(production[c1][0]);
                }
                else
                {

followfirst(production[c1][c2], c1, c2
+ 1);
                }
            }
            j++;
        }
```

```
                }
            }
```

Output:



```
ow many production are there in question? Enter No of Production :- 14
nter 14 productions below and Production should not contain left recursion problem
Production 1] :- S=DPDXQ
Production 2] :- D=digit
Production 3] :- P=.
Production 4] :- X=AKOT
Production 5] :- X=NKO
Production 6] :- A=atle
Production 7] :- K=ketla
Production 8] :- N=na
Production 9] :- T=thay
Production 10] :- O=hour
Production 11] :- O=minute
Production 12] :- O=second
Production 13] :- O=milisecond
Production 14] :- Q=?

First(S) = { d, }

First(D) = { d, }

First(P) = { ., }

First(X) = { a, n, }

First(A) = { a, }

First(K) = { k, }

First(N) = { n, }

First(T) = { t, }

First(O) = { h, m, s, }

First(Q) = { ?, }

-----------------------------------------------

Follow(S) = { $,  }

Follow(D) = { ., a, n,  }

Follow(P) = { d,  }

Follow(X) = { ?,  }

Follow(A) = { k,  }

Follow(K) = { h, m, s,  }

Follow(N) = { k,  }

Follow(T) = { ?,  }

Follow(O) = { t, ?,  }

Follow(Q) = { $,  }


--------------------------------
rocess exited after 138.6 seconds with return value 0
ress any key to continue . . .
```

3.0.2 Yacc based imlementation of syntax analyzer

- **project.l (Lex file)**

```
%{
        #include<stdio.h>
        #include "project.tab.h"
        extern int yylval;
%}

%%
[0-9]+ {yylval=atoi(yytext);  printf("%s : number\n",yytext);
return NUM; }
"na"|"atle"|"ketla"   printf("%s : keyword\n",yytext);   return
KEYQUE;
"minute" printf("%s : opratore\n",yytext);  return UNITM;
"second" printf("%s : opratore\n",yytext);  return UNITS;
"hour" printf("%s : opratore\n",yytext);  return UNITH;
"milisecond" printf("%s : opratore\n",yytext);  return UNITMS;
"?" printf("%s : question mark\n",yytext); return Q;
[\t] ;
[\n] return 0;
. return yytext[0];
%%
int yywrap()
{
return 1;
}
```

- **project.y (yacc code)**

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

%token NUM
%token UNITM
%token UNITS
%token UNITH
%token UNITMS
%token KEYQUE
%token Q

%%
ADI: A{
printf("Answer is: %d \n",$$);
return 0; };
|B{
printf("Answer is: %f",(float)$$/60);
return 0;
};
|C{
printf("Answer is: %f",(float)$$/1000);
return 0;
};


A: NUM' 'UNITH' 'KEYQUE' 'KEYQUE' 'UNITM' 'Q {$$=($1*60);}
  |NUM' 'UNITH' 'KEYQUE' 'UNITM' 'KEYQUE' 'Q {$$=($1*60);}
  |NUM' 'UNITH' 'KEYQUE' 'KEYQUE' 'UNITS' 'Q {$$=($1*60*60);}
  |NUM' 'UNITH' 'KEYQUE' 'UNITS' 'KEYQUE' 'Q {$$=($1*60*60);}
  |NUM' 'UNITH' 'KEYQUE' 'KEYQUE' 'UNITMS' 'Q
{$$=($1*60*60*1000);}
  |NUM' 'UNITH' 'KEYQUE' 'UNITMS' 'KEYQUE' 'Q
{$$=($1*60*60*1000);}
  |NUM' 'UNITM' 'KEYQUE' 'KEYQUE' 'UNITS' 'Q {$$=($1*60);}
  |NUM' 'UNITM' 'KEYQUE' 'UNITS' 'KEYQUE' 'Q {$$=($1*60);}
```

```
          |NUM' 'UNITM' 'KEYQUE' 'KEYQUE' 'UNITMS' 'Q
{$$=($1*60*1000);}
          |NUM' 'UNITM' 'KEYQUE' 'UNITMS' 'KEYQUE' 'Q
{$$=($1*60*1000);}
          |NUM' 'UNITS' 'KEYQUE' 'KEYQUE' 'UNITMS' 'Q {$$=($1*1000);}
          |NUM' 'UNITS' 'KEYQUE' 'UNITMS' 'KEYQUE' 'Q {$$=($1*1000);}
        ;
        B: NUM' 'UNITM' 'KEYQUE' 'KEYQUE' 'UNITH' 'Q {$$=$1;}
          |NUM' 'UNITM' 'KEYQUE' 'UNITH' 'KEYQUE' 'Q {$$=$1;}
          |NUM' 'UNITS' 'KEYQUE' 'KEYQUE' 'UNITH' 'Q {$$=(float)$1/60;}
          |NUM' 'UNITS' 'KEYQUE' 'UNITH' 'KEYQUE' 'Q {$$=(float)$1/60;}
          |NUM' 'UNITS' 'KEYQUE' 'KEYQUE' 'UNITM' 'Q {$$=$1;}
          |NUM' 'UNITS' 'KEYQUE' 'UNITM' 'KEYQUE' 'Q {$$=$1;}


          ;

        C: NUM' 'UNITMS' 'KEYQUE' 'KEYQUE' 'UNITS' 'Q {$$=$1;}
          |NUM' 'UNITMS' 'KEYQUE' 'UNITS' 'KEYQUE' 'Q {$$=$1;}
          |NUM' 'UNITMS' 'KEYQUE' 'KEYQUE' 'UNITH' 'Q
{$$=(float)$1/3600;}
          |NUM' 'UNITMS' 'KEYQUE' 'UNITH' 'KEYQUE' 'Q
{$$=(float)$1/3600;}
          |NUM' 'UNITMS' 'KEYQUE' 'KEYQUE' 'UNITM' 'Q {$$=(float)$1/60;}
          |NUM' 'UNITMS' 'KEYQUE' 'UNITM' 'KEYQUE' 'Q {$$=(float)$1/60;}
        ;
        %%
        void main(){
                printf("valid string\n");
                yyparse();
        }
        void yyerror(){printf("Please enter valid value. \n");}
```

3.0.3 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.

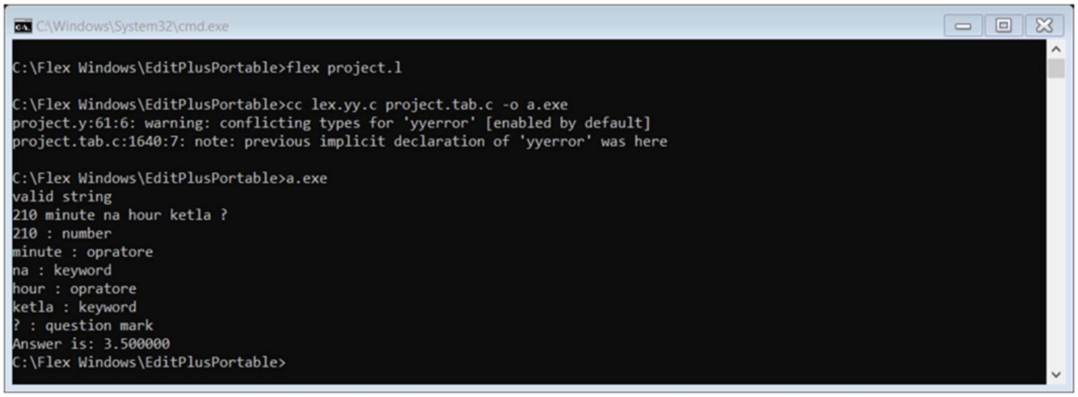
To run the program:

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

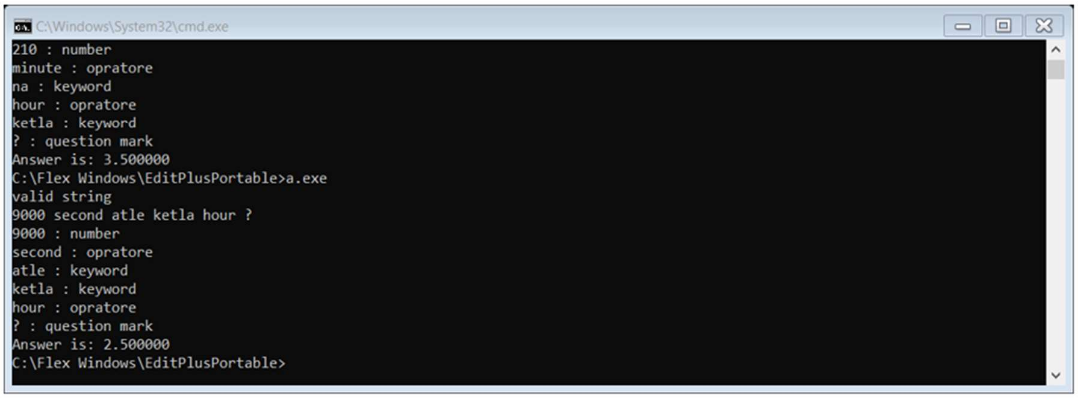
```
flex yacc.l bison -dy yacc.y gcc
lex.yy.c y.tab.c -o yacc.exe
```

3.0.4 Output screenshots of yacc based implementation

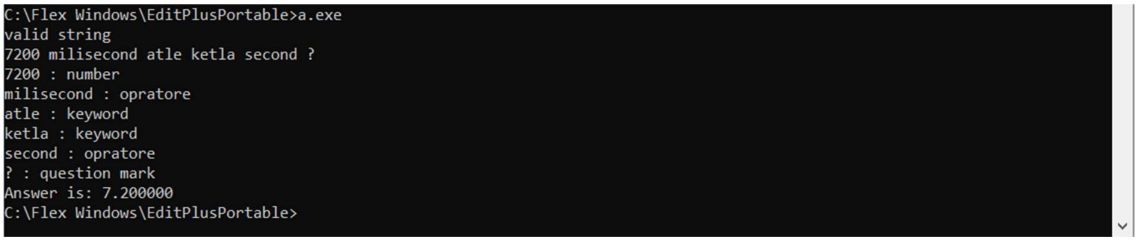
- **Valid Input with all the possible combinations:**



```
C:\Flex Windows\EditPlusPortable>flex project.l

C:\Flex Windows\EditPlusPortable>cc lex.yy.c project.tab.c -o a.exe
project.y:61:6: warning: conflicting types for 'yyerror' [enabled by default]
project.tab.c:1640:7: note: previous implicit declaration of 'yyerror' was here

C:\Flex Windows\EditPlusPortable>a.exe
valid string
210 minute na hour ketla ?
210 : number
minute : opratore
na : keyword
hour : opratore
ketla : keyword
? : question mark
Answer is: 3.500000
C:\Flex Windows\EditPlusPortable>
```



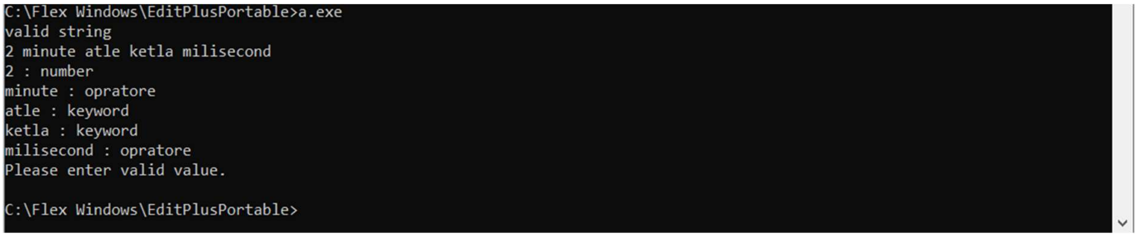
```
210 : number
minute : opratore
na : keyword
hour : opratore
ketla : keyword
? : question mark
Answer is: 3.500000
C:\Flex Windows\EditPlusPortable>a.exe
valid string
9000 second atle ketla hour ?
9000 : number
second : opratore
atle : keyword
ketla : keyword
hour : opratore
? : question mark
Answer is: 2.500000
C:\Flex Windows\EditPlusPortable>
```



```
C:\Flex Windows\EditPlusPortable>a.exe
valid string
7200 milisecond atle ketla second ?
7200 : number
milisecond : opratore
atle : keyword
ketla : keyword
second : opratore
? : question mark
Answer is: 7.200000
C:\Flex Windows\EditPlusPortable>
```

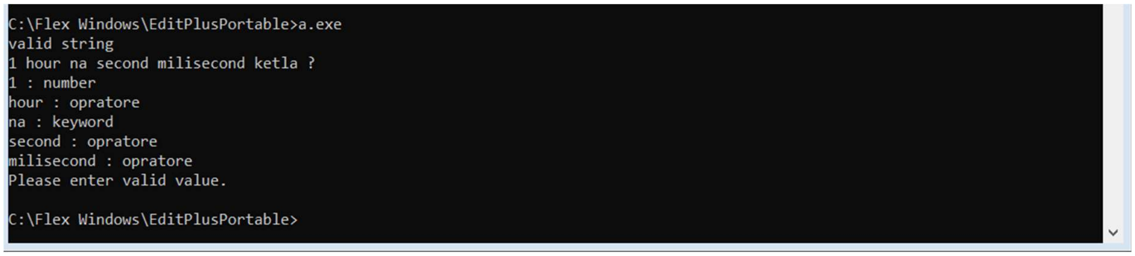
- **Invalid Syntax:**

1. **Program is not complete yet (expecting input after dot)**



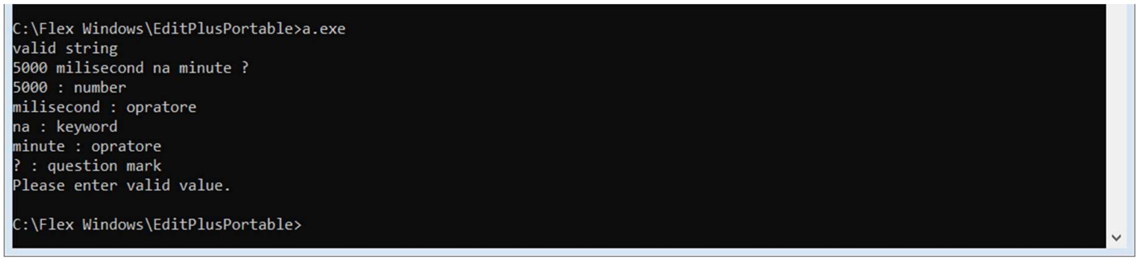
```
C:\Flex Windows\EditPlusPortable>a.exe
valid string
2 minute atle ketla milisecond
2 : number
minute : opratore
atle : keyword
ketla : keyword
milisecond : opratore
Please enter valid value.

C:\Flex Windows\EditPlusPortable>
```

2. **Invalid input(use two operators consecutively)**



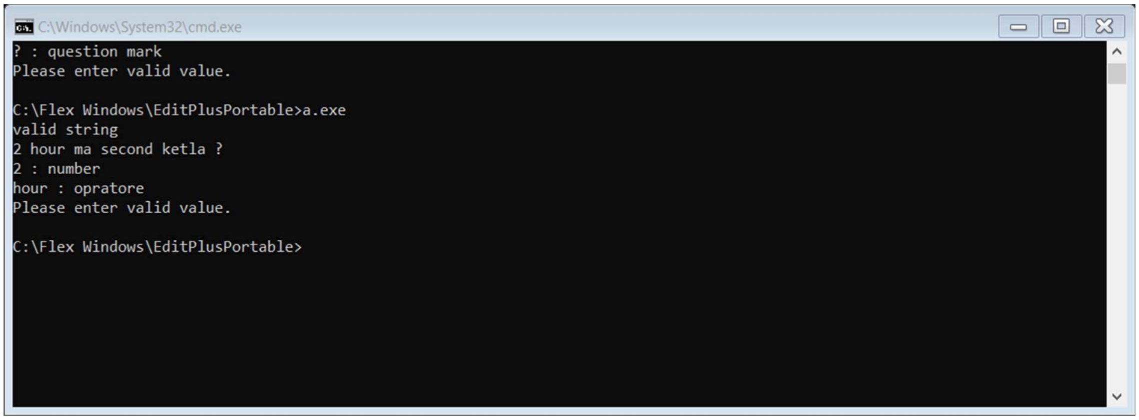
```
C:\Flex Windows\EditPlusPortable>a.exe
valid string
1 hour na second milisecond ketla ?
1 : number
hour : opratore
na : keyword
second : opratore
milisecond : opratore
Please enter valid value.

C:\Flex Windows\EditPlusPortable>
```

3. **Missing another number or misplacement of a keyword**



```
C:\Flex Windows\EditPlusPortable>a.exe
valid string
5000 milisecond na minute ?
5000 : number
milisecond : opratore
na : keyword
minute : opratore
? : question mark
Please enter valid value.

C:\Flex Windows\EditPlusPortable>
```

4. Invalid token



```
? : question mark
Please enter valid value.

C:\Flex Windows\EditPlusPortable>a.exe
valid string
2 hour ma second ketla ?
2 : number
hour : opratore
Please enter valid value.

C:\Flex Windows\EditPlusPortable>
```

4.0 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.