Project Report
on
Compiler for English Sentence Analysis System

Developed by

Mehul Chovatiya– IT023 20ITUOS028

Meet Dadhania-IT024 20ITUOS079

Manushi Dagli-IT025 20ITUON046

Sanjana daki –IT026 20ITUBS051

Guided By:
Prof. Nilamba Vala
Dept. of Information Technology

Department of Information Technology
Faculty of Technology, Dharmsinh Desai University
College Road, Nadiad-387001
2022-2023

DHARMSINH DESAI UNIVERSITY
NADIAD-387001, GUJARAT

CERTIFICATE

This is to certify that the project entitled "**Compiler for English Sentence Analysis System ”** is a bonafied report of the work carried out by

1) Mehul Chovatiya, Student ID No: 20ITUOS028
2) Meet Dadhania, Student ID No: 20ITUOS079
3) Manushi Dagli , Student ID No: 20ITUON046
4) Sanjana daki , Student ID No: 20ITUBS051

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 2022-2023.

Prof. Nilamba Vala
(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

Table of Contents

1.0 INTRODUCTION

1.0.1 Project Details

Language Name: English Sentence Analysis System

Language description:

This language analyzes the grammar of English sentences and identifies the subject, verb, object, and other parts of speech. The input is a sentence in English and the output is the analysis of the sentence in terms of its grammatical structure, including the identification of the subject, verb, object, and other relevant parts of speech.

Input:
"He had been driving car"

Expected output:
subject = He
had = had
been = been
verb-with-ing = driving
object = car
eos = .

2.0 LEXICAL PHASE DESIGN

2.0.1 Regular Expression:

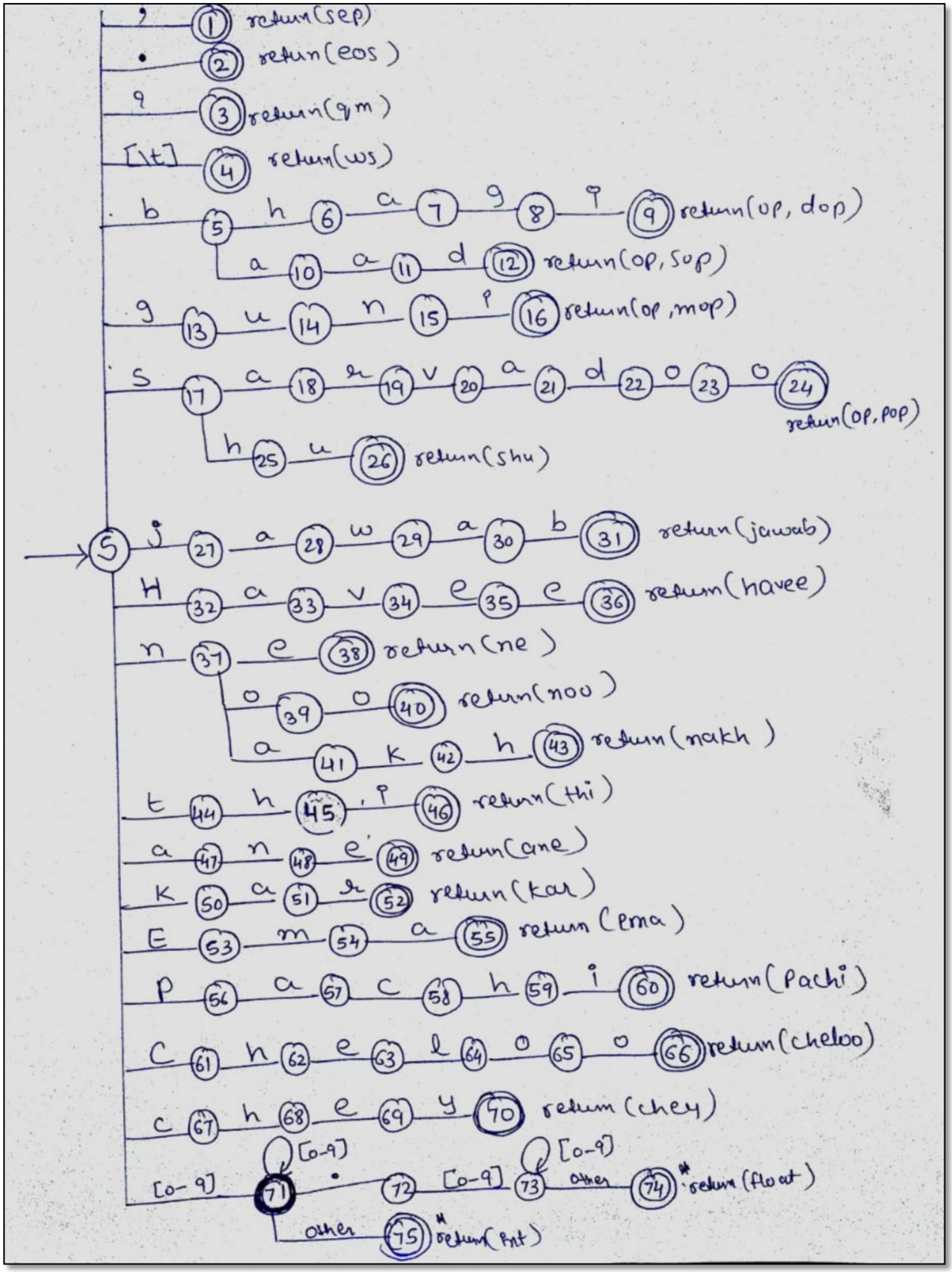
Keywords :

Token	**RE**
subjecttypeone	"He"|"She"|"It"
subjecttypetwo	"I"
subjecttypethree	"We"|"They"|"You"
am	"am"
is	"is"
are	"are"
have	"have"
has	"has"
had	"had"
been	"been"
was	"was"
were	"were"
willshall	"will"|"shall"
be	"be"
verb	"play"|"smash"|"give"|"feel"|"write"|"watch"|"ask"|"learn"|"begin"|"choose"|"drive"
verbwiths	"plays"|"smashes"|"gives"|"feels"|"writes"|"watches"|"askes"|"learns"|"begins"|"chooses"|"drives"
verbwithing	"playing"|"smashing"|"giving"|"feeling"|"writing"|"watching"|"asking"|"learning"|"beginning"|"choosing"|"driving"
verbpastandparti	"played"|"smashed"|"felt"|"Watched"|"asked"|"learned"|"learnt"
verbpast	"gave"|"wrote"|"began"|"chose"|"drove"
verbpastparti	"given"|"written"|"begun"|"chosen"|"driven"
object	"cricket"|"car"|"football"|"teacher"|"anime"|"maths"|"violence"|"glass"|"letter"|"journal"|"dog"|"blessings"|"tension"|"Netflix"|"bike"

Delimiters : {. , and \t \n}

Token	**RE**
eos	"."
separator	","|"and"
ws	[\t\n]

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;

               '.':
                     state = "2";
                     ch = nextchar();
                     f = true;
                       break;

                  '?':
                     state = "3";
```

```
        ch = nextchar();
        f = true;
          break;

[\t ]:
        state = "4";
        ch = nextchar();
        f= true;
           break;

‘b’:
        state = "5";
        ch = nextchar();
          break;

‘g’:
        state = "13";
        ch = nextchar();
          break;

‘s’:
        state = "17";
        ch = nextchar();
           break;

   ‘j’:
        state = "27";
        ch = nextchar();
          break;
   'H’:
        state = "32";
        ch = nextchar();
          break;
   'n':
        state = "37";
```

```
        ch = nextchar();
           break;
  ‘t’:
        state = "44";
        ch = nextchar();
          break;
‘a’:
        state = "47";
        ch = nextchar();
          break;
‘k’:
        state = "50";
        ch = nextchar();
          break;
‘E’:
        state = "53";
        ch = nextchar();
          break;
‘P’:
        state = "56";
        ch = nextchar();
          break;
‘C’:
        state = "61";
        ch = nextchar();
          break;
‘c’:
        state = "67";
        ch = nextchar();
          break;

[0-9]:
        state = "71";
        ch = nextchar();
          break;
```

```
                    }
                default:
                    f= true;
            end case

case state of "5":
                case state of 'h':
                    state = "6";
                    ch = nextchar();
                        break;
                case state of 'a':
                    state = "10";
                    ch = nextchar();
                        break;
                    default:
                    f = true;

case state of "6":
                case state of 'a':
                    state = "7";
                    ch = nextchar();

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

case state of "8":
                case state of 'i':
                    state = "9";
                    ch = nextchar();
                    f= true;

case state of "10":
```

```
                case state of 'a':
                    state = "11";
                    ch = nextchar();

case state of "11":
                case state of 'd':
                    state = "12";
                    ch = nextchar();
                    default:
                    f= true;

case state of "13":
                case state of 'u':
                    state = "14";
                    ch = nextchar();

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

case state of "15":
                case state of 'i':
                    state = "16";
                    ch = nextchar();
                    f= true;

case state of "17":
                case state of 'h':
                    state = "25";
                    ch = nextchar();
                          break;
                  case state of 'a':
                    state = "18";
                    ch = nextchar();
```

```
                    break;

        case state of "18":
                case state of 'r':
                    state = "19";
                    ch = nextchar();

case state of "19":
                case state of 'v':
                    state = "20";
                    ch = nextchar();

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

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

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

case state of "23":
                case state of 'o':
                    state = "24";
                    ch = nextchar();
                        f = true;

case state of "25":
                case state of 'u':
```

```
                state = "26";
                ch = nextchar();
                    f= true;

case state of "27":
            case state of 'a':
                state = "28";
                ch = nextchar();

 case state of "28":
            case state of 'w':
                state = "29";
                ch = nextchar();

 case state of "29":
            case state of 'a':
                state = "30";
                ch = nextchar();

 case state of "30":
            case state of 'b':
                state = "31";
                ch = nextchar();
                f= true;

 case state of "32":
            case state of 'a':
                state = "33";
                ch = nextchar();

 case state of "33":
            case state of 'v':
                state = "34";
                ch = nextchar();
```

```
case state of "34":
        case state of 'e':
            state = "35";
            ch = nextchar();

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

case state of "37":
        case state of 'e':
            state = "38";
            ch = nextchar();
            f= true;
                break;
         case state of 'o':
            state = "39";
            ch = nextchar();
                break;
         case state of 'a':
            state = "41";
            ch = nextchar();
                break;

case state of "39":
        case state of 'o':
            state = "40";
            ch = nextchar();
            f = true;

case state of "41":
        case state of 'k':
            state = "2"4;
```

```
                ch = nextchar();

case state of "42":
            case state of 'h':
                state = "43";
                ch = nextchar();
                f = true;

case state of "44":
            case state of 'h':
                state = "45";
                ch = nextchar();

case state of "45":
            case state of 'i':
                state = "46";
                ch = nextchar();
                 f= true;

case state of "47":
            case state of 'n':
                state = "48";
                ch = nextchar();

case state of "48":
            case state of 'e':
                state = "49";
                ch = nextchar();
                 f = true;


case state of "50":
            case state of 'a':
                state = "51";
                ch = nextchar();
```

```
case state of "51":
        case state of 'r':
            state = "52";
            ch = nextchar();
             f = true;

case state of "53":
        case state of 'm':
            state = "54";
            ch = nextchar();

case state of "54":
        case state of 'a':
            state = "55";
            ch = nextchar();
             f = true;

case state of "56":
        case state of 'a':
            state = "57";
            ch = nextchar();

case state of "57":
        case state of 'c':
            state = "58";
            ch = nextchar();

case state of "58":
        case state of 'h':
            state = "59";
            ch = nextchar();

case state of "59":
        case state of 'i':
```

```
                    state = "60";
                    ch = nextchar();
                    f = true;

case state of "61":
                case state of 'h':
                    state = "62";
                    ch = nextchar();

case state of "62":
                case state of 'e':
                    state = "63";
                    ch = nextchar();

case state of "63":
                case state of 'l':
                    state = "64";
                    ch = nextchar();

case state of "64":
                case state of 'o':
                    state = "65";
                    ch = nextchar();

case state of "65":
                case state of 'o':
                    state = "66";
                    ch = nextchar();
                    f = true;


case state of "67":
                case state of 'h':
                    state = "68";
                    ch = nextchar();
```

```
case state of "68":
            case state of 'e':
               state = "69";
               ch = nextchar();

case state of "69":
            case state of 'y':
               state = "70";
               ch = nextchar();
                f = true;

case state of "71":
            case state of [0-9]:
               ch = nextchar();
                    break;
            case state of '.':
               state = "72";
               ch = nextchar();
                    break;
            default:
               state = "75";
               f= true;

 case state of "72":
            case state of [0-9]:
               state = "73";
               ch = nextchar();
            default:
              f = true;

 case state of "73":
            case state of [0-9]:
               ch = nextchar();
            default:
```

```
                    state = "74";
                    f = true;
            }
    }
        case state of
          "26"|"31"|”36"|"38"|"40"|"43"|"46"|"49"|"52"|"55"|"60"|"66"|"70":
            print(" keyword");

          "75":
            print(" int");

         "74":
            print(" float");

          "9"|"12"|"16"|"24":
            print("operator");

          "1":
            print(" sep");

          "2":
            print(" eos");

          "3":
            print(" que tag");
         "4":
            print(" ws ");

          default:
            print("invalid input");
            ch := nextchar();
          end case;
      }
    }
```

2.0.4 Implementation of lexer

Flex Program:

```
%{
#include<stdio.h>
#include "y.tab.h"
%}
subjecttypeone  "He"|"She"|"It"
subjecttypetwo  "I"
subjecttypethree    "We"|"They"|"You"
am          "am"
is          "is"
are         "are"
have            "have"
has         "has"
had         "had"
been            "been"
was         "was"
were            "were"
willshall       "will"|"shall"
be          "be"
verb            "play"|"smash"|"give"|"feel"|"write"|"watch"|"ask"|"learn"|"begin"|"choos
e"|"drive"
verbwiths       "plays"|"smashes"|"gives"|"feels"|"writes"|"watches"|"askes"|"learns"|"be
gins"|"chooses"|"drives"
verbwithing     "playing"|"smashing"|"giving"|"feeling"|"writing"|"watching"|"asking"|"le
arning"|"beginning"|"choosing"|"driving"
verbpastandparti    "played"|"smashed"|"felt"|"Watched"|"asked"|"learned"|"learnt"
verbpast        "gave"|"wrote"|"began"|"chose"|"drove"
verbpastparti   "given"|"written"|"begun"|"chosen"|"driven"
object      "cricket"|"car"|"football"|"teacher"|"anime"|"maths"|"violence"|"glass"|"lett
er"|"journal"|"dog"|"blessings"|"tension"|"Netflix"|"bike"
eos         "."
separator       ","|"and"
ws          [ \t\n]
%%
{subjecttypeone}        {printf("subject = %s\n",yytext);return SUBJECTTYPEONE;}
{subjecttypetwo}        {printf("subject = %s\n",yytext);return SUBJECTTYPETWO;}
{subjecttypethree}  {printf("subject = %s\n",yytext);return SUBJECTTYPETHREE;}
{am}                {printf("am = %s\n",yytext);return AM;}
{is}                {printf("is = %s\n",yytext);return IS;}
{are}               {printf("are = %s\n",yytext);return ARE;}
{have}          {printf("have = %s\n",yytext);return HAVE;}
{has}               {printf("has = %s\n",yytext);return HAS;}
{had}               {printf("had = %s\n",yytext);return HAD;}
```

```
{been}          {printf("been = %s\n",yytext);return BEEN;}
{was}               {printf("was = %s\n",yytext);return WAS;}
{were}          {printf("were = %s\n",yytext);return WERE;}
{willshall}         {printf("will/shall = %s\n",yytext);return WILLSHALL;}
{be}                {printf("be = %s\n",yytext);return BE;}
{verbwiths}         {printf("verb-with-s = %s\n",yytext);return VERBWITHS;}
{verbwithing}       {printf("verb-with-ing = %s\n",yytext);return VERBWITHING;}
{verbpastparti}     {printf("verb-past-participle = %s\n",yytext);return VERBPASTPARTI;}
{verbpast}          {printf("verb-past = %s\n",yytext);return VERBPAST;}
{verbpastandparti}  {printf("verb-past-as-well-as-past-participle = %s\n",yytext);return
VERBPASTANDPARTI;}
{verb}          {printf("verb = %s\n",yytext);return VERB;}
{object}            {printf("object = %s\n",yytext);return OBJECT;}
{eos}               {printf("eos = %s\n",yytext);return DOT;}
{separator}         {printf("separator = %s\n",yytext);return SEPERATOR;}
{ws}                {return WHITESPACE;}
.               {printf("Invalid Token : %s\n",yytext);return 0;return *yytext;}
%%
int yywrap()
{
return 1;
}
```

2.0.5 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

2.0.6 Output screenshots of lexer.

Input:

```
C:\Users\Manisha\Desktop\flex programs>project.exe
2000 ,500 ,400 ane 23 noo sarvadoo kar. Ema thi 300 baad kar.
```

Output:

```
Integer          -      2000
Separator        -      ,
Integer          -      500
Separator        -      ,
Integer          -      400
Keyword          -      ane
Integer          -      23
Keyword          -      noo
Operator         -      sarvadoo
Keyword          -      kar
End of sentence  -      .
Keyword          -      Ema
Keyword          -      thi
Integer          -      300
Operator         -      baad
Keyword          -      kar
End of sentence  -      .
```

Input:

```
C:\Users\Manisha\Desktop\flex programs>project.exe
Havee 40 ne 5 thi guni nakh. Pachi jawab ne 4 thi bhagi nakh.Cheloo jawab shu chey?
```

Output:

```
Keyword          -      Havee
Integer          -      40
Keyword          -      ne
Integer          -      5
Keyword          -      thi
Operator         -      guni
Keyword          -      nakh
End of sentence  -      .
Keyword          -      Pachi
Keyword          -      jawab
Keyword          -      ne
Integer          -      4
Keyword          -      thi
Operator         -      bhagi
Keyword          -      nakh
End of sentence  -      .
Keyword          -      Cheloo
Keyword          -      jawab
Keyword          -      shu
Keyword          -      chey
End of program   -      ?
```

Invalid tokens:

1. **Operation starting with capital Letter:**

```
C:\Users\Manisha\Desktop\flex programs>a.exe
5 ,8 noo Sarvadoo kar.
Integer          -      5
Separator        -      ,
Integer          -      8
Keyword          -      noo
Invalid Token : S

C:\Users\Manisha\Desktop\flex programs>
```

2. **Operation is invalid:**

```
C:\Users\Manisha\Desktop\flex programs>a.exe
Havee jawab ne 5 thi gunii nakh.
Keyword          -      Havee
Keyword          -      jawab
Keyword          -      ne
Integer          -      5
Keyword          -      thi
Operator         -      guni
Invalid Token : i

C:\Users\Manisha\Desktop\flex programs>
```

3. **Keyword is invalid:**

```
C:\Users\Manisha\Desktop\flex programs>a.exe
Ema thi 10 baad karjo.
Keyword          -      Ema
Keyword          -      thi
Integer          -      10
Operator         -      baad
Keyword          -      kar
Invalid Token : j

C:\Users\Manisha\Desktop\flex programs>
```

3.0 SYNTAX ANALYZER DESIGN

3.0.1 Grammar rules

```
START -> A
A -> B DOT  | B  SEPERATOR  A | B  DOT  A
B ->SUBJECTTYPEONE  C | SUBJECTTYPETWO  D | SUBJECTTYPETHREE  E
C ->H | I | K | IS  G | HAS  J | HAS  K | HAS  BEEN  G | WAS  G | HAD  J | HAD  K |
HAD  BEEN  G | WILLSHALL  F | WILLSHALL  BE  G | WILLSHALL  HAVE  J | WILLSHALL  HAVE  K
| WILLSHALL  HAVE  BEEN  G
D ->F | I | K | AM  G | HAVE  J | HAS  K | HAVE  BEEN  G | WAS  G | HAD  J | HAD  K |
HAD  BEEN  G | WILLSHALL  F | WILLSHALL  BE  G | WILLSHALL  HAVE  J | WILLSHALL  HAVE  K
| WILLSHALL  HAVE  BEEN  G
E ->F | I | K | ARE  G | HAVE  J | HAS  K | HAVE  BEEN  G | WERE  G | HAD  J | HAD  K |
HAD  BEEN  G | WILLSHALL  F | WILLSHALL  BE  G | WILLSHALL  HAVE  J | WILLSHALL  HAVE  K
| WILLSHALL  HAVE  BEEN  G
F -> VERB  OBJECT
G -> VERBWITHING  OBJECT
H -> VERBWITHS  OBJECT
I -> VERBPAST  OBJECT
J -> VERBPASTPARTI  OBJECT
K -> VERBPASTANDPARTI  OBJECT
```

3.0.2 Yacc based imlementation of syntax analyzer

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

```
%{
#include<stdio.h>
#include "y.tab.h"
%}
subjecttypeone  "He"|"She"|"It"
subjecttypetwo  "I"
subjecttypethree    "We"|"They"|"You"
am          "am"
is          "is"
are         "are"
have            "have"
has         "has"
had         "had"
been            "been"
was         "was"
were            "were"
willshall       "will"|"shall"
be          "be"
verb            "play"|"smash"|"give"|"feel"|"write"|"watch"|"ask"|"learn"|"begin"|"choos
e"|"drive"
verbwiths       "plays"|"smashes"|"gives"|"feels"|"writes"|"watches"|"askes"|"learns"|"be
gins"|"chooses"|"drives"
verbwithing     "playing"|"smashing"|"giving"|"feeling"|"writing"|"watching"|"asking"|"le
arning"|"beginning"|"choosing"|"driving"
verbpastandparti    "played"|"smashed"|"felt"|"Watched"|"asked"|"learned"|"learnt"
verbpast        "gave"|"wrote"|"began"|"chose"|"drove"
verbpastparti   "given"|"written"|"begun"|"chosen"|"driven"
object      "cricket"|"car"|"football"|"teacher"|"anime"|"maths"|"violence"|"glass"|"lett
er"|"journal"|"dog"|"blessings"|"tension"|"Netflix"|"bike"
eos         "."
separator       ","|"and"
ws          [ \t\n]
%%
{subjecttypeone}        {printf("subject = %s\n",yytext);return SUBJECTTYPEONE;}
{subjecttypetwo}        {printf("subject = %s\n",yytext);return SUBJECTTYPETWO;}
{subjecttypethree}  {printf("subject = %s\n",yytext);return SUBJECTTYPETHREE;}
{am}                {printf("am = %s\n",yytext);return AM;}
{is}                {printf("is = %s\n",yytext);return IS;}
{are}               {printf("are = %s\n",yytext);return ARE;}
{have}          {printf("have = %s\n",yytext);return HAVE;}
{has}               {printf("has = %s\n",yytext);return HAS;}
{had}               {printf("had = %s\n",yytext);return HAD;}
{been}          {printf("been = %s\n",yytext);return BEEN;}
{was}               {printf("was = %s\n",yytext);return WAS;}
{were}          {printf("were = %s\n",yytext);return WERE;}
```

```
{willshall}         {printf("will/shall = %s\n",yytext);return WILLSHALL;}
{be}                {printf("be = %s\n",yytext);return BE;}
{verbwiths}         {printf("verb-with-s = %s\n",yytext);return VERBWITHS;}
{verbwithing}       {printf("verb-with-ing = %s\n",yytext);return VERBWITHING;}
{verbpastparti}     {printf("verb-past-participle = %s\n",yytext);return VERBPASTPARTI;}
{verbpast}          {printf("verb-past = %s\n",yytext);return VERBPAST;}
{verbpastandparti}  {printf("verb-past-as-well-as-past-participle = %s\n",yytext);return
VERBPASTANDPARTI;}
{verb}          {printf("verb = %s\n",yytext);return VERB;}
{object}            {printf("object = %s\n",yytext);return OBJECT;}
{eos}               {printf("eos = %s\n",yytext);return DOT;}
{separator}         {printf("separator = %s\n",yytext);return SEPERATOR;}
{ws}                {return WHITESPACE;}
.               {printf("Invalid Token : %s\n",yytext);return 0;return *yytext;}
%%
int yywrap()
{
return 1;
}
```

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

```
4     %{
5     #include<stdio.h>
6     #include<stdlib.h>
7     #define YYERROR_VERBOSE 1
8     void yyerror(char *err);
9     %}
10    %token SUBJECTTYPEONE SUBJECTTYPETWO SUBJECTTYPETHREE AM IS ARE HAVE HAS HAD
      BEEN WAS WERE WILLSHALL BE VERB VERBWITHS VERBWITHING VERBPAST VERBPASTPARTI
      VERBPASTANDPARTI OBJECT SEPERATOR DOT WHITESPACE
11    %%
12    START : A {printf("\nSentence is grammatically correct !!!\n");return 0;};
13    A : B DOT WHITESPACE | B WHITESPACE SEPERATOR WHITESPACE A | B WHITESPACE DOT
      WHITESPACE A;
14    B : SUBJECTTYPEONE WHITESPACE C | SUBJECTTYPETWO WHITESPACE D |
      SUBJECTTYPETHREE WHITESPACE E;
15    C : H | I | K | IS WHITESPACE G | HAS WHITESPACE J | HAS WHITESPACE K | HAS
      WHITESPACE BEEN WHITESPACE G | WAS WHITESPACE G | HAD WHITESPACE J | HAD
      WHITESPACE K | HAD WHITESPACE BEEN WHITESPACE G | WILLSHALL WHITESPACE F |
      WILLSHALL WHITESPACE BE WHITESPACE G | WILLSHALL WHITESPACE HAVE WHITESPACE J |
      WILLSHALL WHITESPACE HAVE WHITESPACE K | WILLSHALL WHITESPACE HAVE WHITESPACE
      BEEN WHITESPACE  G;
16    D : F | I | K | AM WHITESPACE G | HAVE WHITESPACE J | HAS WHITESPACE K | HAVE
      WHITESPACE BEEN WHITESPACE G | WAS WHITESPACE G | HAD WHITESPACE J | HAD
      WHITESPACE K | HAD WHITESPACE BEEN WHITESPACE G | WILLSHALL WHITESPACE F |
      WILLSHALL WHITESPACE BE WHITESPACE G | WILLSHALL WHITESPACE HAVE WHITESPACE J |
      WILLSHALL WHITESPACE HAVE WHITESPACE K | WILLSHALL WHITESPACE HAVE WHITESPACE
      BEEN WHITESPACE  G;
```

```
17    E : F | I | K | ARE WHITESPACE G | HAVE WHITESPACE J | HAS WHITESPACE K | HAVE
      WHITESPACE BEEN WHITESPACE G | WERE WHITESPACE G | HAD WHITESPACE J | HAD
      WHITESPACE K | HAD WHITESPACE BEEN WHITESPACE G | WILLSHALL WHITESPACE F |
      WILLSHALL WHITESPACE BE WHITESPACE G | WILLSHALL WHITESPACE HAVE WHITESPACE J |
      WILLSHALL WHITESPACE HAVE WHITESPACE K | WILLSHALL WHITESPACE HAVE WHITESPACE
      BEEN WHITESPACE  G;
18    F : VERB WHITESPACE OBJECT;
19    G : VERBWITHING WHITESPACE OBJECT;
20    H : VERBWITHS WHITESPACE OBJECT;
21    I : VERBPAST WHITESPACE OBJECT;
22    J : VERBPASTPARTI WHITESPACE OBJECT;
23    K : VERBPASTANDPARTI WHITESPACE OBJECT;
24    %%
25    void yyerror(char *err)
26    {
27    printf("Error: ");
28    fprintf(stderr,"%s\n",err);
29    exit(1);
30    }
31    int main()
32    {
33    printf("Enter Sentence in English :\n");
34    yyparse();
35    }
```

35.0.1 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 project.l
bison -dy project.y
gcc lex.yy.c y.tab.c -o project.exe
```

35.0.2 Output screenshots of yacc based implementation

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

```
C:\Users\sejal\Desktop\LT>project
Enter Sentence in English :
I am playing cricket and You are writing journal.
subject = I
am = am
verb-with-ing = playing
object = cricket
separator = and
subject = You
are = are
verb-with-ing = writing
object = journal
eos = .
```

```
C:\Users\sejal\Desktop\LT>project
Enter Sentence in English :
We will watch anime.
subject = We
will/shall = will
verb = watch
object = anime
eos = .

Sentence is grammatically correct !!!
```

```
C:\Users\sejal\Desktop\LT>project
Enter Sentence in English :
You were learning maths.
subject = You
were = were
verb-with-ing = learning
object = maths
eos = .

Sentence is grammatically correct !!!
```

- **Invalid Syntax:**

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

```
C:\Users\sejal\Desktop\LT>project
Enter Sentence in English :
You were learning maths,
subject = You
were = were
verb-with-ing = learning
object = maths
separator = ,
Error: syntax error, unexpected SEPERATOR, expecting DOT or WHITESPACE
```

2. Dot should be used to mark end of the sentence

```
C:\Users\sejal\Desktop\LT>project
Enter Sentence in English :
You were learning maths ?
subject = You
were = were
verb-with-ing = learning
object = maths
Invalid Token : ?
Error: syntax error, unexpected $end, expecting SEPERATOR or DOT
```

3. Two operations are used consecutively in a sentence.

```
C:\Users\sejal\Desktop\LT>project
Enter Sentence in English :
You were learning maths He is playing cricket.
subject = You
were = were
verb-with-ing = learning
object = maths
subject = He
Error: syntax error, unexpected SUBJECTTYPEONE, expecting SEPERATOR or DOT
```

4. Missing another object

```
C:\Users\sejal\Desktop\LT>project
Enter Sentence in English :
He is playing .
subject = He
is = is
verb-with-ing = playing
eos = .
Error: syntax error, unexpected DOT, expecting OBJECT
```

5. Invalid token

```
C:\Users\sejal\Desktop\LT>project
Enter Sentence in English :
You ware learning maths.
subject = You
Invalid Token : w
Error: syntax error, unexpected $end
```

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.