Project Report
On

Compiler for Given Language

Prepared by:

Dhruv Patel - IT092 - 18ITUOS122

Dirgh Patel - IT093 - 18ITUOS123

Pranay Raycha - IT108 - 18ITUOS038

Guided by:

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

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

2020-2021

DHARMSINH DESAI UNIVERSITY
NADIAD-387001, GUJARAT

CERTIFICATE

This is to certify that the project entitled “**Compiler Design Project”** is a bonafide report of the work carried out by :

1) Mr. Dhruv Patel, Student ID No: 18ITUOS092
2) Mr. Dirgh Patel , Student ID No: 18ITUOS093
3) Mr. Pranay Raycha, Student ID No: 18ITUOS038

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

Prof. N.P. Desai
(Lab Incharge)
Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University, Nadiad
Date: 07/04/21

Prof. (Dr.)V K Dabhi,
Head , Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University,
Nadiad
Date: 07/04/21

Index

1.0 INTRODUCTION

1.0.1 Project Details

Project definition:

Our Programmer's Language allows following

- Valid operations:-

  1. x=p+q
     Concatenates the p with q and stores result in x

  2. x=p-q
     Removes the last "q" symbols from p and stores answer back in x.

  3. x=p<->q
     Replaces all occurrence of p variable with q variable in x string .

  4. pos=x?p
     Find and return the first position in X where q string was found.

- Variables can be of type: "string" or "char".

- Variables can be assigned values using "=" symbol.

- Variable names should start with "character in small case" and cannot contain any integers. The maximum size of the name can be 3.

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

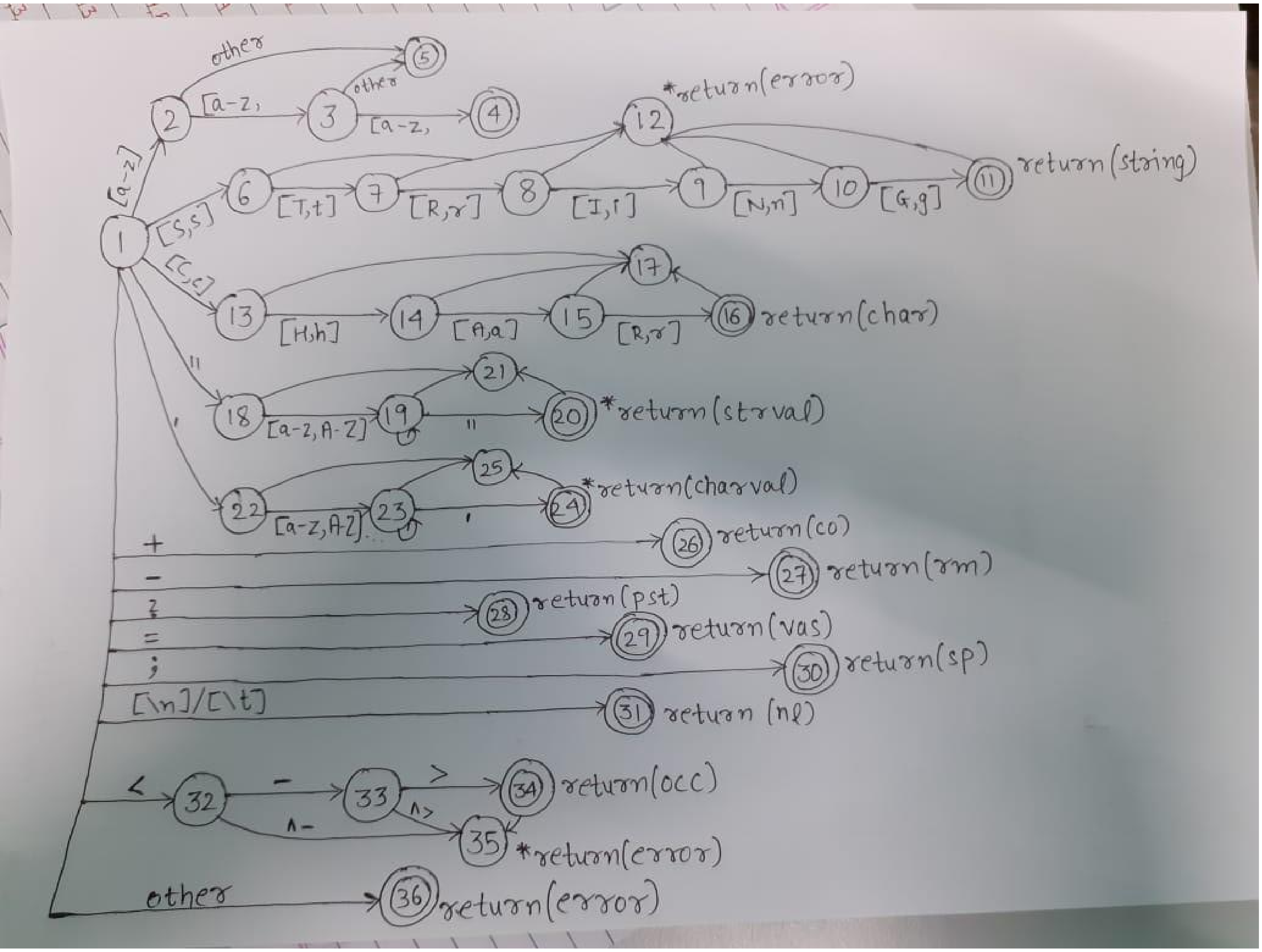
IT108 Pranay Raycha– Grammar Rules, YACC Implementation, Scanner Implementation, Final Code Verification

IT093 Dirgh Patel– Algorithm Design, DFA Design, Scanner Implementation, YACC Implementation, Final Code Verification

IT092 Dhruv Patel– DFA Design, Regular Expression

2.0 LEXICAL PHASE DESIGN

2.0.1 Deterministic Finite Automaton design for lexer



2.0.2 Algorithm of lexer:

```
state:=0;
  input(ch);
  while not eof do
  case state of
     :1
      case ch of [A-Z]:
          state:=2;
          input(ch);
             end case;
          'S':
          state:=6;
          input(ch);
             end case;
          's':
          state:=6;
          input(ch);
            end case;
        'C':
          state:=13;
  input(ch);
    end case;
'c':
  state:=13;
  input(ch);
    end case;
‘"’:
  state:=18;
  input(ch);
    end case;
‘'’:
  state:=22;
  input(ch);
    end case;
'+':
  state:=26;
  end while;
    end case;
'-':
  state:=27;
  end while;
    end case;
'?':
```

```
          state:=28;
          end while;
             end case;
        '=':
          state:=29;
          end while;
             end case;
        ';':
          state:=30;
          end while;
             end case;
        ' ' or '\t' or '\n':
          state:=31;
          end while;
             end case;
        '<':
          state:=32;
          input(ch);
             end case;
        else
          state:=36;
          end while;
     end case;
```

```
2:
 case ch of
      [A-Z,a-z]:
     state:=3;
     input(ch);
        end case;
    else
   state:=5;
   unput(ch);
   end while;
end case;
3:
 case ch of
      [A-Z,a-z]:
     state:=4;
     end while;
        end case;
    else
   state:=5;
   unput(ch);
   end while;
end case;
6:
 case ch of
      'T':
    state:=7;
    input(ch);
       end case;
      't':
    state:=7;
    input(ch);
         end case;
      else
    state:=12;
    unput(ch);
    end while;
end case;

7:
 case ch of
      'R':
    state:=8;
    input(ch);
       end case;
   'r':
    state:=8;
```

```
          input(ch);
             end case;
           else
          state:=12;
          unpput(ch);
          end while;
    end case;
     8:
      case ch of
           'I':
          state:=9;
          input(ch);
             end case;
      case ch of
           'i':
          state:=9;
          input(ch);
             end case;
           else
          state:=12;
          unput(ch);
          end while;
     end case;
```

```
 9:
  case ch of
       'N':
      state:=10;
      input(ch);
         end case;
       else
      state:=12;
      unput(ch);
      end while;
 end case;
10:
  case ch of
       'G':
      state:=11;
      end while;
         end case;
       else
      state:=12;
      unput(ch);
      end while;
 end case;
 13:
```

```
 case ch of
      'H':
    state:=14;
    input(ch);
       end case;
      'h':
    state:=14;
    input(ch);
       end case;
      else
    state:=17;
    unput(ch);
    end while;
 end case;
14:
 case ch of
      'A':
    state:=15;
    input(ch);
       end case;
   'a':
    state:=15;
    input(ch);
```

```
            end case;
          else
         state:=17;
         unput(ch);
         end while;
     end case;
    15:
     case ch of
       'R':
         state:=16;
         input(ch);
       end case;
       'r':
         state:=16;
         input(ch);
       end case;
          else
         state:=17;
         unput(ch);
         end while;
     end case;
    18:
     case ch of
```

```
          [A-Z,a-z]:
         state:=19;
         input(ch);
            end case;
       else
         state:=21;
         unput(ch);
         end while;
     end case;
   19:
     case ch of
          [A-Z,a-z]:
         state:=19;
         input(ch);
            end case;
       '"':
         state:=20;
         end while;
       else
         state:=21;
         unput(ch);
         end while;
     end case
```

```
22:
  case ch of
       [A-Z,a-z]:
    state:=23;
    input(ch);
       end case;
   else
    state:=25;
    unput(ch);
    end while;
  end case;
23:
  case ch of
   ‘’:
    state:=24;
    end while;
   else
    state:=25;
    unput(ch);
    end while;
  end case
32:
  case ch of
```

```
          '-':
          state:=33;
          end while;
             end case;
        else
          state:=35;
          unput(ch);
          end while;
        end case
    29:
      case ch of
          '>':
          state:=34;
          end while;
             end case;
        else
          state:=35;
          unput(ch);
          end while;
        end case
       end while;
  case state of
       4:5:
```

```
 ID();
 accept(ID);
  end case;
11:
 accept(STRING);
    end case;
15:
 accept(CHAR);
    end case;
20:
 strval();
 accept(STRVAL);
    end case;
24:
 chrval();
 accept(CHRVAL);
    end case;
26:
 op(CO);
 accept(OP);
    end case;
27:
 op(RM);
 accept(OP);
    end case;
28:
 op(PST);
 accept(OP);
    end case;
29:
 op(VAS);
 accept(assign);
    end case;
30:
 accept(SEP);
    end case;
  31: end case;
34:
 op(OCC);
 accept(OP);
    end case;
36:
 accept(ERROR);
    end case;
12:17:21:25:35:
 accept(ERROR);
```

```
        end case
```

2.0.3 Implementation of lexer

Flex Program:

```
%option noyywrap
%{
#include<stdio.h>
#include<conio.h>
int test();
int total = 0;
%}
%%
("if")|("else")|("while")|("do")|("continue")|("untill")|("for")|("case")|("switch") { total++;
printf("This is Keyword: %s\n",yytext);
}
[a-zA-Z_][a-zA-Z0-9_]* {
total++;
printf("This is Identifier: %s\n",yytext);
}
[0-9]*"."[0-9]+ {
total++;
printf("This is Float : %s\n", yytext);
}
[0-9]+ {
total++;
```

```
printf("This is Integer: %s\n",yytext);
}
"*"." " {
total++;
printf("This is single line Comments: %s\n",&yytext[3]); }
[,|;(|)|{|}|.|\[|\]] {
total++;
printf("This is Punctuation: %s\n",yytext);
}
"+"|"-"|"?"|"<->" {
total++;
printf("This is (mathametical) Operator: %s\n",yytext); }
'[^']+' {
total++;
printf("This is String: %s\n",yytext);
}
[" "\t] {
printf("This is a white space");
}
"$" return 0;
%%

int main(){
yylex();
printf("Total: %d",total);
getch();
return 0;
```

}

C code:

```
#include<stdio.h>
#include<stdbool.h>
#include<string.h>
#include <ctype.h>

void nextToken(char* string);
void print(char* string);
int main()
{
        char inputStr[500];
        while(true){
                        gets(inputStr);
                        nextToken(inputStr);
                }
        return 0;
}
void nextToken(char* string){
                int state = 1;
                char ch;
                int c=0;
                int d;
                bool  b= false;
                int len=strlen(string);
                while(c < len){

                        state=1;
                        while(c < len){
                                if(b){
                                        b=false;
        break;
}
ch=string[c];
switch(state){
        case 1:
                d = string - strchr(string,' ');

                if(isupper(ch)&&((d-c)<=2)){
                        state=2;
                        c++;
                }
                else if((ch=='S') || (ch=='s')){
                        state=6;
                        c++;
                }
                else if((ch=='C') || (ch=='c')){
                        state=12;
                        c++;
                }
                else if(ch=='\n'){
                        state=16;
                        c++;
                        b=true;
                }
                else if((ch==' ') || (ch=='\t')){
                        state=17;
                        c++;
                        b=true;
                }
                else if(ch==';'){
                        state=18;
                        c++;
                        b=true;
                }
                else if(ch=='+'){
                        state=19;
                        c++;
                        b=true;
                }
                else if(ch=='-'){
```

```
                                        state=20;
                                        c++;
                                        b=true;
                                }
                                else if(ch=='?'){
                                        state=21;
                                        c++;
                                        b=true;
                                }
                                else if(ch=='<'){
                                        state=22;
                                        c++;
                                        b=true;
                                }
                                else if(((ch>='a') && (ch<='z'))||((ch>='A') &&
(ch<='Z'))){
                                        state=25;
                                        c++;
                                        b=true;
                                }
                                else if(ch=='='){
                                        state=27;
                                        c++;
                                        b=true;
                                }
                                else{
                                        state=25;
                                        b=true;
                                }
                                break;
                        case 2:
                                if((ch>=64) && (ch<=123)){
                                        state=3;
                                        c++;
                                }
                                else{
                                        state=5;
                                        c++;
                                        b=true;
                                }
```

```
        break;
case 3:
        if((ch>=64) && (ch<=123)){
                state=4;
                c++;
                b=true;
        }
        else{
                state=5;
                c++;
                b=true;
        }
        break;
case 6:
        if((ch=='t') || (ch=='T')){
                state=7;
                c++;
        }
        else{
                b=true;
        }
        break;
case 7:
        if((ch=='r')||( ch=='R')){
                state=8;
                c++;
        }
        else{
                b=true;
        }
        break;
case 8:
        if((ch=='i') || (ch=='I')){
                state=9;
                c++;
        }
        else{
                b=true;
        }
        break;
```

```
case 9:
        if((ch=='n') || (ch=='N')){
                state=10;
                c++;
        }
        else{
                b=true;
        }
        break;
case 10:
        if((ch=='g') || (ch=='G')){
                state=11;
                c++;
                b=true;
        }
        else{
                b=true;
        }
        break;
case 12:
        if((ch=='h') || (ch=='H')){
                state=13;
                c++;
        }
        else{
                b=true;
        }
        break;
case 13:
        if((ch=='a')|| (ch=='A')){
                state=14;
                c++;
        }
        else{
                b=true;
        }
        break;
case 14:
        if((ch=='r') || (ch=='R')){
                state=15;
```

```
                            c++;
                            b=true;
                        }
                        else{
                            b=true;
                        }
                        break;
                    case 22:
                        if(ch=='-'){
                            state=23;
                            c++;
                        }
                        else{
                            b=true;
                        }
                        break;
                    case 23:
                        if(ch=='>'){
                            state=24;
                            c++;
                            b=true;
                        }
                        else{
                            b=true;
                        }
                        break;
                    case 25:
                        if(((ch>='a') && (ch<='z'))||((ch>='A') &&
(ch<='Z'))){
                            c++;
                        }
                        else if(ch=='"'){
                            state=26;
                            c++;
                            b=true;
                        }
                        break;
                }
            }
            switch(state){
```

```
case 26:
        print("String");
        break;
case 4:case 5:
        print(" ID ");
        break;
case 11:
        print(" STR ");
        break;
case 15:
        print(" CHR ");
        break;
case 16:
        print(" NEW ");
        break;
case 17:
        print(" Space ");
        break;
case 18:
        print(" DEL ");
        break;
case 19:
        print(" CON ");
        break;
case 20:
        print(" REM ");
        break;
case 21:
        print(" OCCU ");
        break;
case 24:
        print(" POS ");
        break;
case 27:
        print(" VAS ");
        break;
default:
        print("?");
        c++;
        break;
```

```
                    }
            }
            print("\n");
    }
void print(char* string){
            printf(string);
}
```

2.0.4 Execution environment setup

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

Operating system: Windows 10

Step 1 /*For downloading CODEBLOCKS */

- Open your Browser and type in "codeblocks"

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

- Download codeblocks-20.03mingw-setup.exe

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

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

- Downloading will start automatically

- Install the software keep clicking on next

/*SAVE IT INSIDE C FOLDER*

Step 2 /*PATH SETUP FOR CODEBLOCKS*/

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

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

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

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

- Click on edit

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

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

- Press Ok!

Step 3 /*PATH SETUP FOR GnuWin32*/

- After successful installation Goto C folder

- Goto GnuWin32-->Bin

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

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

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

- C:\GnuWin32\bin

- Press Ok

Step 4

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

- Open notepad type in a flex program

- Save it inside the folder like filename.l

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

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

Step 5

/*To RUN FLEX PROGRAM*/

- Goto to Command Prompt(cmd)

- Goto the directory where you have saved the program

- Type in command :- flex filename.l

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

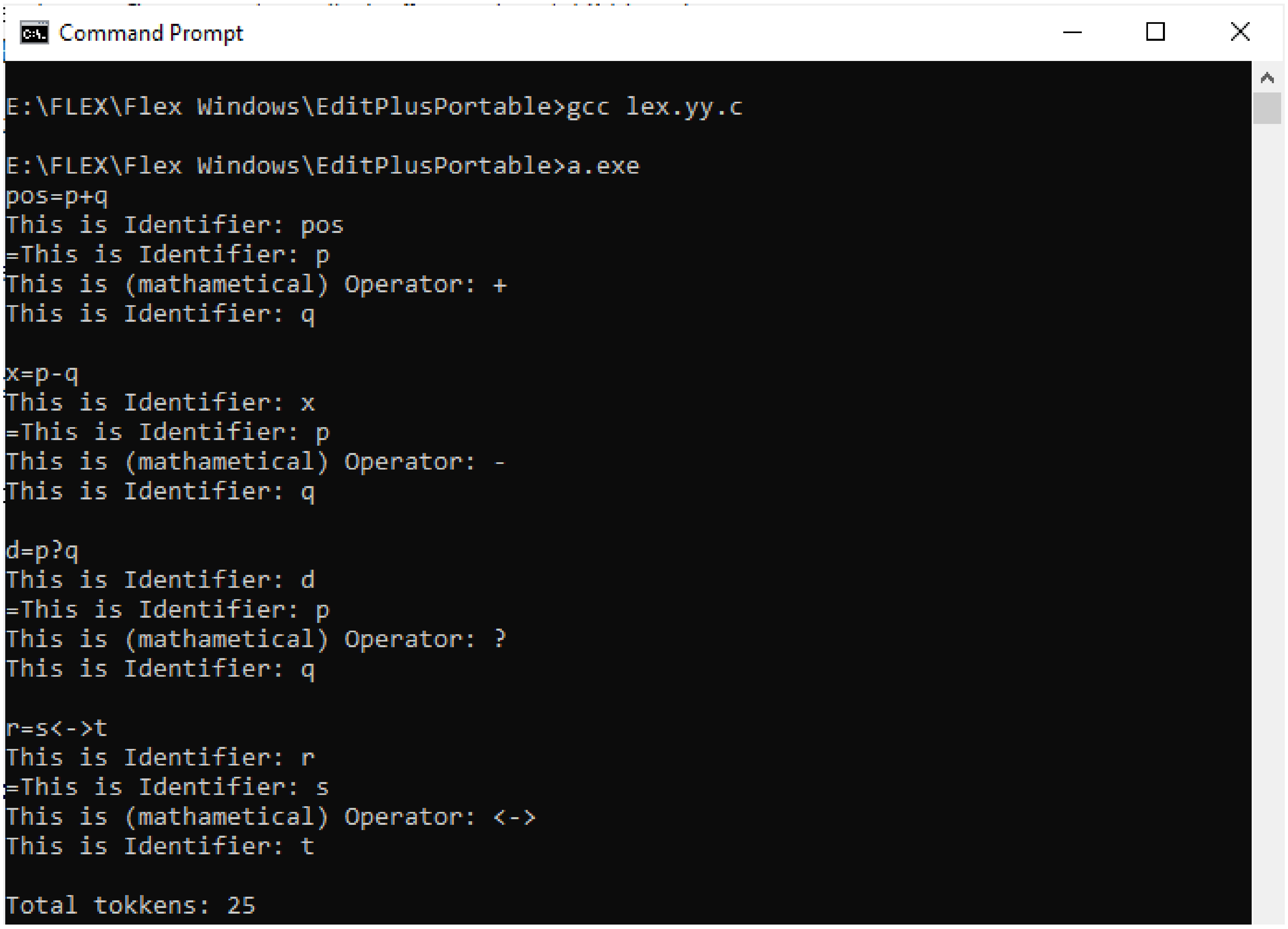
- Execute/Run for windows command prompt :- **a.exe**

Step 6

- Finished

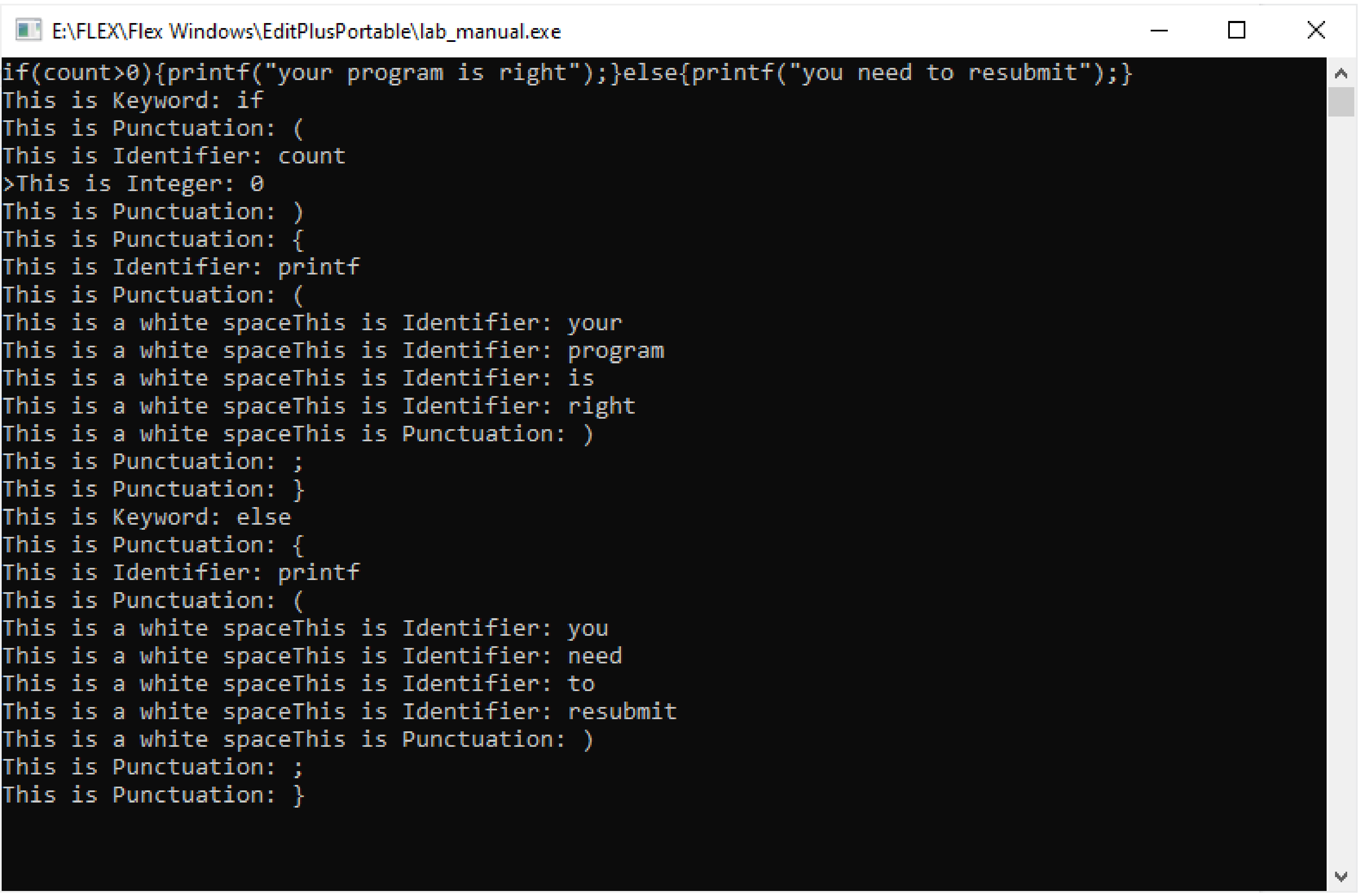
2.0.5 Output screenshots of Lexer:

1.



```
E:\FLEX\Flex Windows\EditPlusPortable>gcc lex.yy.c

E:\FLEX\Flex Windows\EditPlusPortable>a.exe
pos=p+q
This is Identifier: pos
=This is Identifier: p
This is (mathametical) Operator: +
This is Identifier: q

x=p-q
This is Identifier: x
=This is Identifier: p
This is (mathametical) Operator: -
This is Identifier: q

d=p?q
This is Identifier: d
=This is Identifier: p
This is (mathametical) Operator: ?
This is Identifier: q

r=s<->t
This is Identifier: r
=This is Identifier: s
This is (mathametical) Operator: <->
This is Identifier: t

Total tokkens: 25
```

2.


E:\FLEX\Flex Windows\EditPlusPortable\lab_manual.exe

```
if(count>0){printf("your program is right");}else{printf("you need to resubmit");}
This is Keyword: if
This is Punctuation: (
This is Identifier: count
>This is Integer: 0
This is Punctuation: )
This is Punctuation: {
This is Identifier: printf
This is Punctuation: (
This is a white spaceThis is Identifier: your
This is a white spaceThis is Identifier: program
This is a white spaceThis is Identifier: is
This is a white spaceThis is Identifier: right
This is a white spaceThis is Punctuation: )
This is Punctuation: ;
This is Punctuation: }
This is Keyword: else
This is Punctuation: {
This is Identifier: printf
This is Punctuation: (
This is a white spaceThis is Identifier: you
This is a white spaceThis is Identifier: need
This is a white spaceThis is Identifier: to
This is a white spaceThis is Identifier: resubmit
This is a white spaceThis is Punctuation: )
This is Punctuation: ;
This is Punctuation: }
```

3.0 SYNTAX ANALYZER DESIGN

3.0.1 Grammar rules

```
ID : ID ;
EQU : = ;
CON : + ;
REM = - ;
POS = ? ;
CHAR = Character Value ;
STR = String Value;
REP = <->
COL :

S -> ID EQU ID CON ID
       | ID EQU ID REM ID
       | ID EQU VSTR COL
       | ID EQU VCHAR COL
       | ID EQU ID REP ID
       | ID EQU ID POS ID
       | STRING L COL
       | CHAR L COL

L -> ID REP L | ID
```

Following are different indications used:

p for POS	q for REP	s for VSTR
i for ID	c for CON	c for VCHAR
: for COL	u for STRING	
e for EQU	r for REM	
t for CHAR		

3.0.2 Yacc Implementation

Flex Code :

```
%{

#include "y.tab.h"
void yyerror (char *s);
int yylex();
%}
%%
"print" {return print;}
"<->" {return eop;}
"?" {return qop;}
"exit" {return exit_command;}
[a-zA-Z] {yylval.id = yytext[0]; return
identifier;} [a-z]+ {yylval.no = yytext; return
number;}
[ \t\n] ;
[-+=;] {return yytext[0];}
. {ECHO; yyerror ("unexpected character");}

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

YACC Code:

```
%{

#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
void yyerror (char *s);
int yylex();

char* symbols[100][100];
int indexOf(char * str, char Find);
char* symbolVal(char symbol);
char* replaceWord(char* s, char* old, char*new);
void updateSymbolVal(char symbol, char* val);
char* removeAll(char * str, char * toRemove);

%}

%union {char* no; char id;}
%start line
%token eop
%token qop
%token print
%type <id> assignment
%token <no> number
%token <id> identifier
%type <no> line term
%token exit_command


%%

line : assignment ';' {printf("Valid Statement\n");}
        | exit_command ';' {exit(EXIT_SUCCESS);}
        | print term ';' {printf("Ping %s\n", $2);}
        | line assignment ';' {printf("Valid Statement\n");}
        | line print term ';' {printf("Printing %s\n", $3);}
        | line exit_command ';' {exit(EXIT_SUCCESS);}
```

```
 ;



      assignment : identifier '=' term {
  updateSymbolVal($1,$3); } | identifier
            '=' identifier eop identifier
          {updateSymbolVal($1,replaceWord(symbolVal($
1),symbolVal($3),sy mbolVal($5)) );}
                                  ;
term : number {strcpy($$,$1);}
              | identifier {strcpy($$,symbolVal($1));}
              | identifier '+' identifier
          {strcpy($$,strcat(symbolVal($1),symbolVal($3)));}
              | identifier '-' identifier
          {strcpy($$,removeAll(symbolVal($1),symbolVal($3)));}
 | identifier qop identifier {printf("%c found at index :
%d\n",$3,indexOf(symbolVal($1),$3));}
 ;

%%

int computeSymbolIndex(char token)
{
          int idx = -1;
          if(islower(token)) {
                    idx = token - 'a' + 26;
          } else if(isupper(token)) {
                    idx = token - 'A';
          }
          return idx;
}


char* symbolVal(char symbol)
{
          int bucket = computeSymbolIndex(symbol);
          return symbols[bucket];
}
void updateSymbolVal(char symbol, char* val)
{
          int bucket = computeSymbolIndex(symbol);
```

```
        strcpy(symbols[bucket],val);
}

char* replaceWord(char* s, char* old, char*new)
{
 char* answer;
 int i, cnt = 0;
 int newLength = strlen(newW);
 int oldLength = strlen(old);

 for (i = 0; s[i] != '\0'; i++) {
 if (strstr(&s[i], old) == &s[i]) {
 cnt++;
 i += oldLength - 1;
 }
 }

 answer = (char*)malloc(i
+ cnt * (newLength -
oldLength) + 1);
 i = 0;
 while (*s) {
  if (strstr(s, old) == s) {
 strcpy(&answer[i],new);
 i += newLength;
 s += oldLength;
 }
 else
 answer[i++] = *s++;
 }

 answer[i] = '\0';
 return answer;
}

int indexOf(char * str, char Find)
{
 int i = 0;

 while(str[i] != '\0')
 {
 if(str[i] == Find)
 return i;
 i++;
 }

// Return -1 if char not found
 return -1;
}
char* removeAll(char *
str, char * toRemove) {
 int i, j, stringLen, toRemoveLen;
 int found;

 stringLen = strlen(str);

 toRemoveLen = strlen(toRemove);


 for(i=0; i <= stringLen -
 toRemoveLen; i++) {

 found = 1;
 for(j=0; j<toRemoveLen; j++)
 {
 if(str[i + j] != toRemove[j])
 {
 found = 0;
 break;
 }
 }


 if(str[i + j] != ' ' && str[i + j] != '\t' && str[i + j]
!= '\n' && str[i + j] != '\0') {
 found = 0;
 }

 /*
 * If word is found then shift all characters to left
 * and decrement the string length
 */
 if(found == 1)
 {
 for(j=i; j<=stringLen - toRemoveLen; j++)
 {
 str[j] = str[j + toRemoveLen];
 }

 stringLen = stringLen - toRemoveLen;

i- -;
 }
 }
        return str;
}

int main (void) {

        int i;
        for(i=0; i<52; i++) {
                strcpy(symbols[i],"");
        }

        return yyparse ( );
}

void yyerror (char *s) {fprintf (stderr, "%s\n", s);}
```

<u>3.0.3 Execution Environment setup</u>

Step 1: Installing YACC/Bison

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

Step 2: Saving Flex file

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

Step 3: Running the Flex code

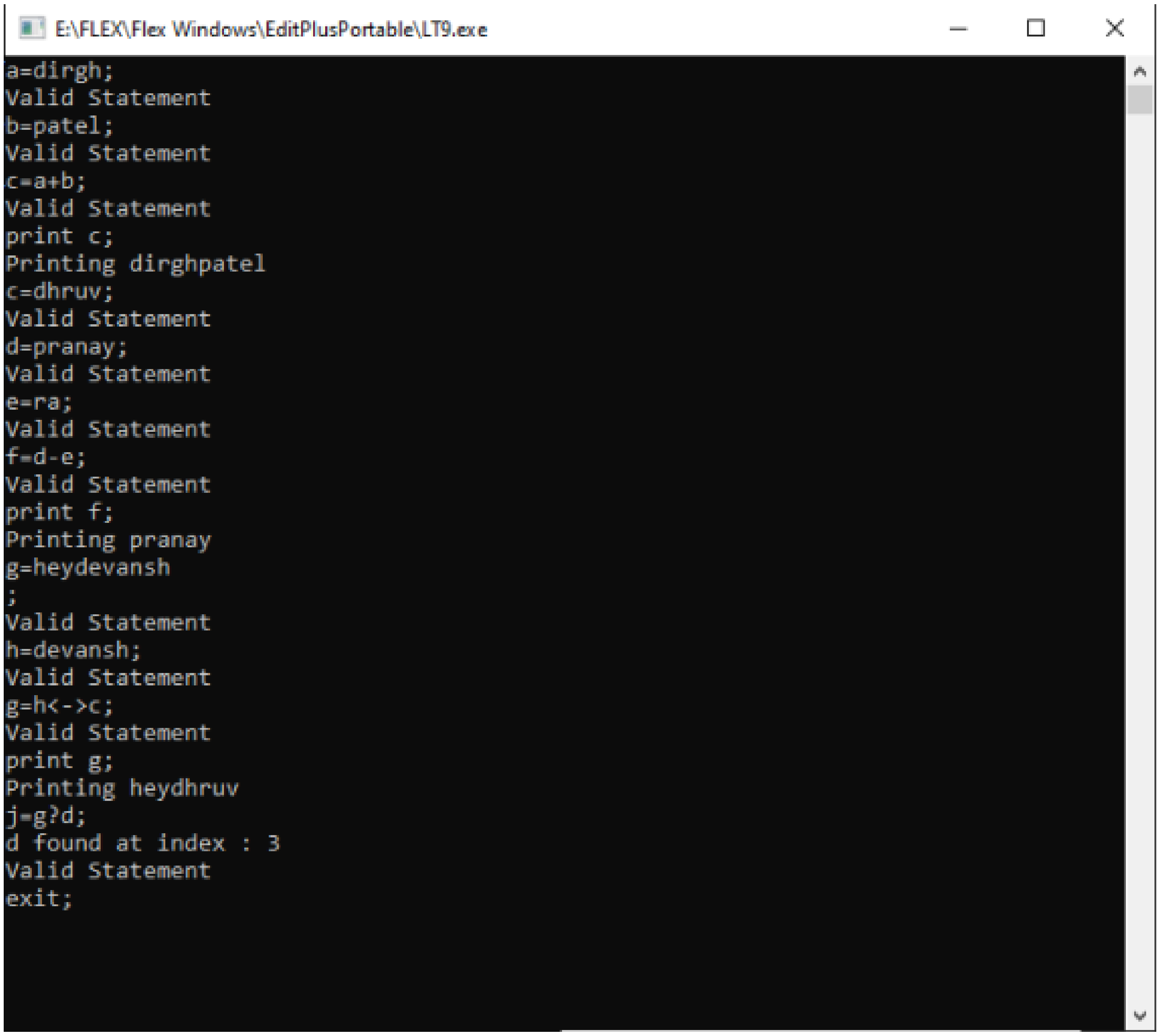
- Run following commands:

1. flex <fileName>.l

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

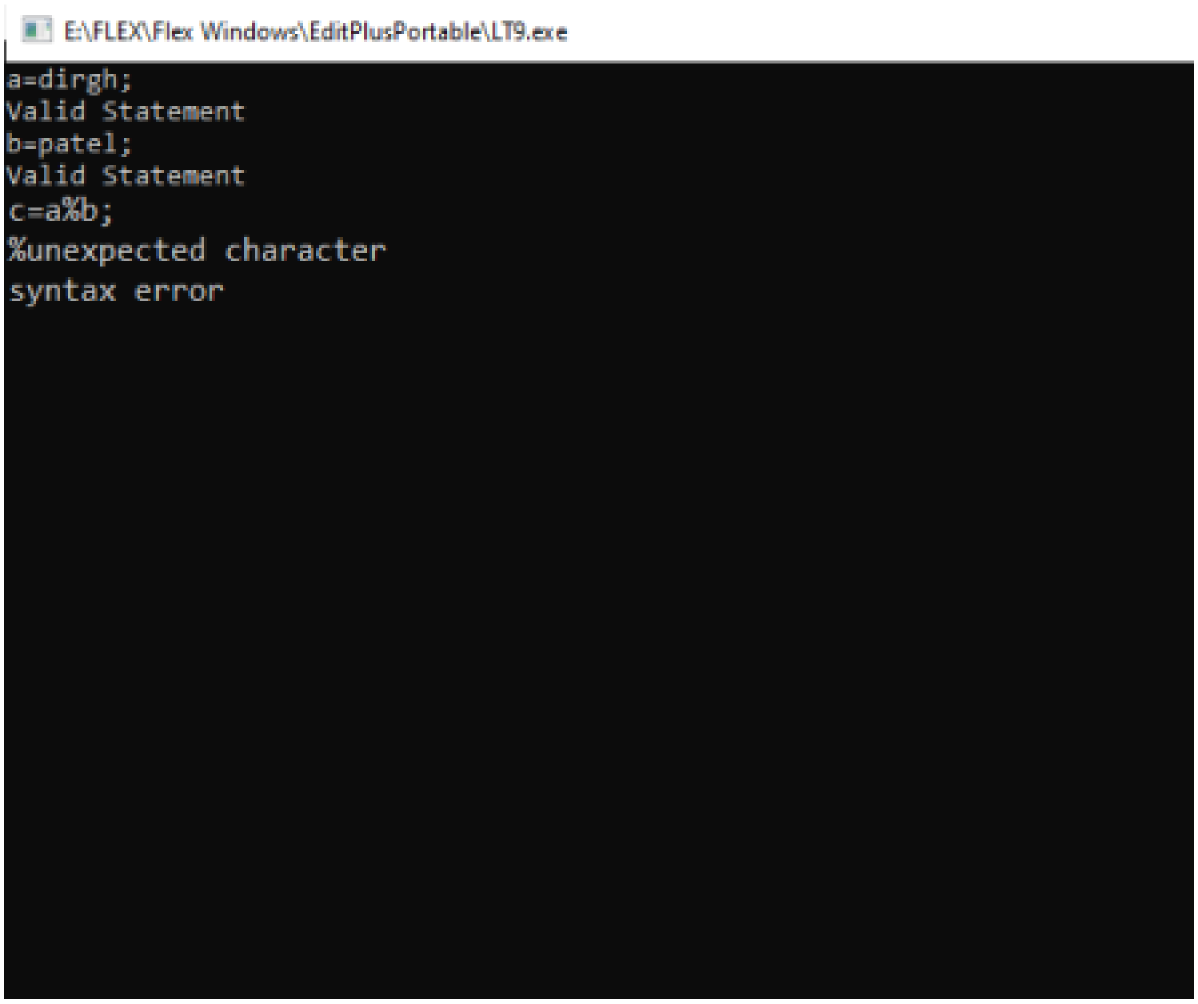
3.0.4 Output screenshots of YACC based implementation

Valid Inputs & their Outputs:


E:\FLEX\Flex Windows\EditPlusPortable\LT9.exe

```
a=dirgh;
Valid Statement
b=patel;
Valid Statement
c=a+b;
Valid Statement
print c;
Printing dirghpatel
c=dhruv;
Valid Statement
d=pranay;
Valid Statement
e=ra;
Valid Statement
f=d-e;
Valid Statement
print f;
Printing pranay
g=heydevansh
;
Valid Statement
h=devansh;
Valid Statement
g=h<->c;
Valid Statement
print g;
Printing heydhruv
j=g?d;
d found at index : 3
Valid Statement
exit;
```

Invalid Inputs & their Outputs:



```
E:\FLEX\Flex Windows\EditPlusPortable\LT9.exe
a=dirgh;
Valid Statement
b=patel;
Valid Statement
c=a%b;
%unexpected character
syntax error
```

4.0 CONCLUSION

Our compiler finds out all the tokens for Given Language and also prints in which groups they belong. And also shows errors if the token is not valid in this language. This project has been built using the knowledge that we have gained from college curriculum and the internet.

This project gave us motivation to have a better understanding of the lexical and parsing phase of compiler designing concepts. We would like to thanks to Prof. Nikita P. Desai for encouraging us to implement this project. This project has helped us build our problem solving, debugging and team work skills and has given us a better insight of how actually the compiler works in real terms.