Aim: C program to identify C Tokens.

1) **Write a C Program to scan reserved word, Identifiers,special characters of C Language.**

Input.c

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
int is_reserved(char *word);
int is_operator(char *word);

int main() {
    FILE *fp;
    char filename[100];
    char ch;
    char buffer[100];
    int buf_idx = 0;
        printf("Enter the name of the file: ");
            scanf("%s", filename);

            fp = fopen(filename, "r");
            if (fp == NULL) {
                printf("Error opening file\n");
                return 1;
            }
  while ((ch = fgetc(fp)) != EOF) {
          if (isalnum(ch) || ch == '_') {
              buffer[buf_idx++] = ch;
          } else {
              buffer[buf_idx] = '\0';
              if (strlen(buffer) > 0) {
                  if (is_reserved(buffer)) {
                      printf("Reserved word: %s\n", buffer);
                  } else {
                      printf("Identifier: %s\n", buffer);} }
buf_idx = 0;

            if (is_operator(&ch)) {
                printf("Operator: %c\n", ch);
            } else if (ispunct(ch)) {
                printf("Special character: %c\n", ch);
            }
        }
    }

    fclose(fp);
    return 0;
}
int is_reserved(char *word) {
    char *reserved_words[] = {"auto", "break", "case", "char", "const", "continue",
        "default", "do", "double", "else", "enum", "extern", "float", "for",
        "goto", "if", "int", "long", "register", "return", "short", "signed",
        "sizeof", "static", "struct", "switch", "typedef", "union", "unsigned",
        "void", "volatile", "while"
    };
    int num_reserved = sizeof(reserved_words) / sizeof(reserved_words[0]);
    for (int i = 0; i < num_reserved; i++) {
        if (strcmp(word, reserved_words[i]) == 0) {
            return 1;
        }
    }
    return 0;
}
int is_operator(char *word) {
    char *operators[] = {"+", "-", "*", "/", "%", "++", "--", "==", "!=", "<", ">", "<=", ">=",
        "&&", "||", "!", "&", "|", "^", "~", "<<", ">>", "=", "+=", "-=", "*=", "/=", "%=",
        "<<=", ">>=", "&=", "^=", "|="
    };

int num_operators = sizeof(operators) / sizeof(operators[0]);
```

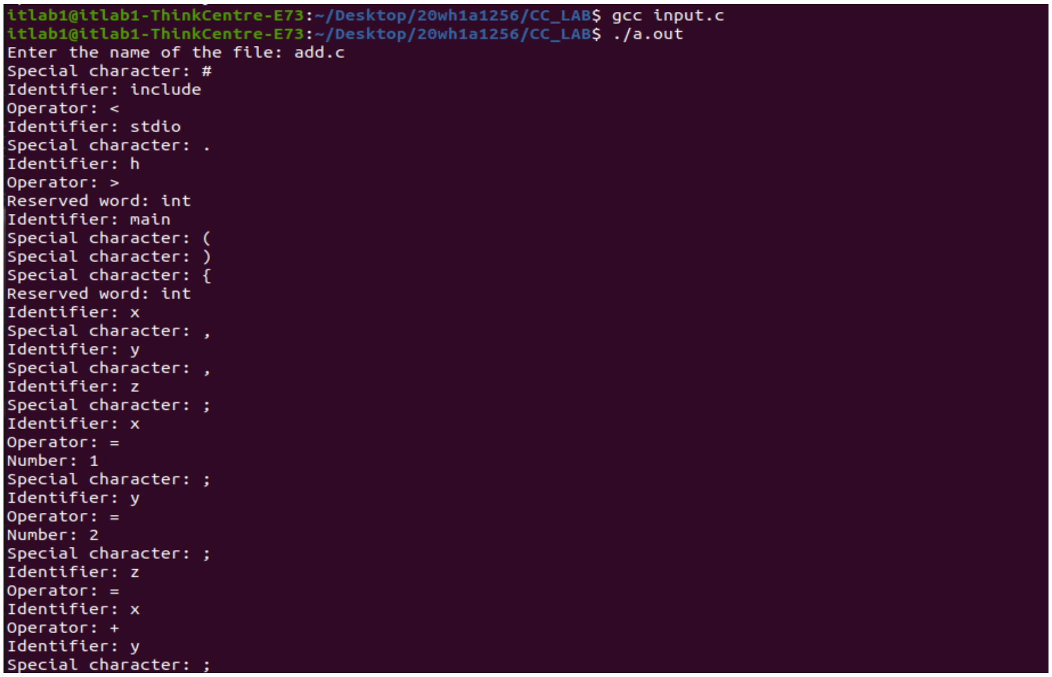
```
for (int i = 0; i < num_operators; i++) {
        if (strcmp(word, operators[i]) == 0) {
            return 1;
        }
    }
    return 0;
}
```

Add.c

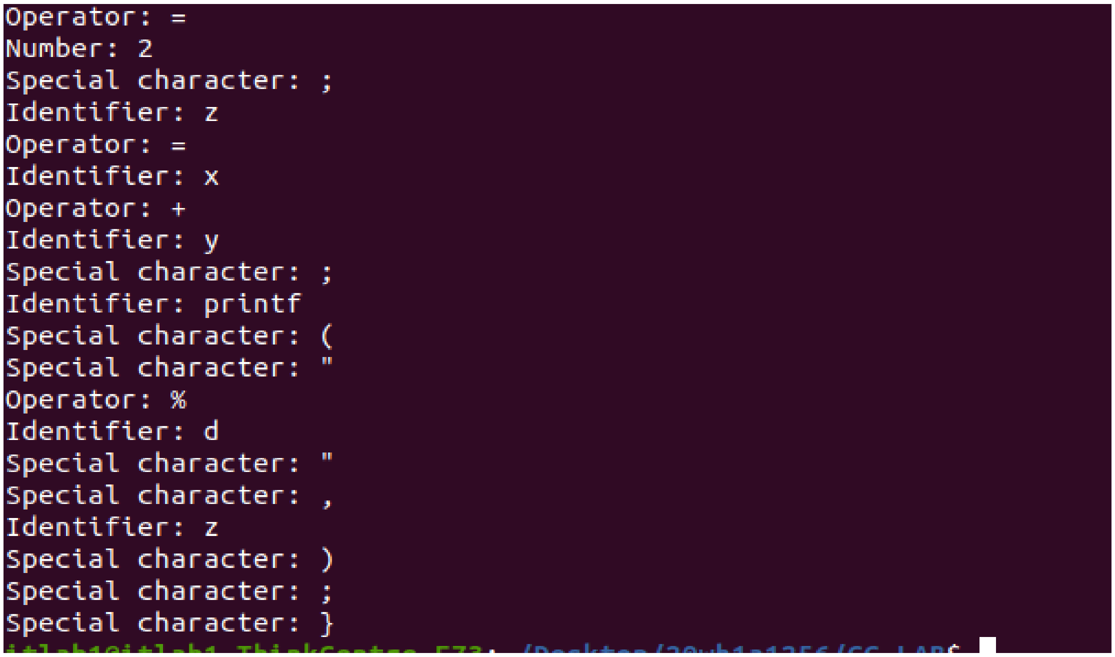
```
 #include<stdio.h>
 int main(){
        int x,y,z;
        x = 1;
        y = 2;
        z = x + y;
        printf("%d",z);
 }
```

Input: C program is given as imput for identifying tokens.

Output : Displays the tokens from add.c program.



```
itlab1@itlab1-ThinkCentre-E73:~/Desktop/20wh1a1256/CC_LAB$ gcc input.c
itlab1@itlab1-ThinkCentre-E73:~/Desktop/20wh1a1256/CC_LAB$ ./a.out
Enter the name of the file: add.c
Special character: #
Identifier: include
Operator: <
Identifier: stdio
Special character: .
Identifier: h
Operator: >
Reserved word: int
Identifier: main
Special character: (
Special character: )
Special character: {
Reserved word: int
Identifier: x
Special character: ,
Identifier: y
Special character: ,
Identifier: z
Special character: ;
Identifier: x
Operator: =
Number: 1
Special character: ;
Identifier: y
Operator: =
Number: 2
Special character: ;
Identifier: z
Operator: =
Identifier: x
Operator: +
Identifier: y
Special character: ;
```



```
Operator: =
Number: 2
Special character: ;
Identifier: z
Operator: =
Identifier: x
Operator: +
Identifier: y
Special character: ;
Identifier: printf
Special character: (
Special character: "
Operator: %
Identifier: d
Special character: "
Special character: ,
Identifier: z
Special character: )
Special character: ;
Special character: }
itlab1@itlab1-ThinkCentre-E73:~/Desktop/20wh1a1256/CC LAB$
```

Aim: LEX program to identify C Tokens.

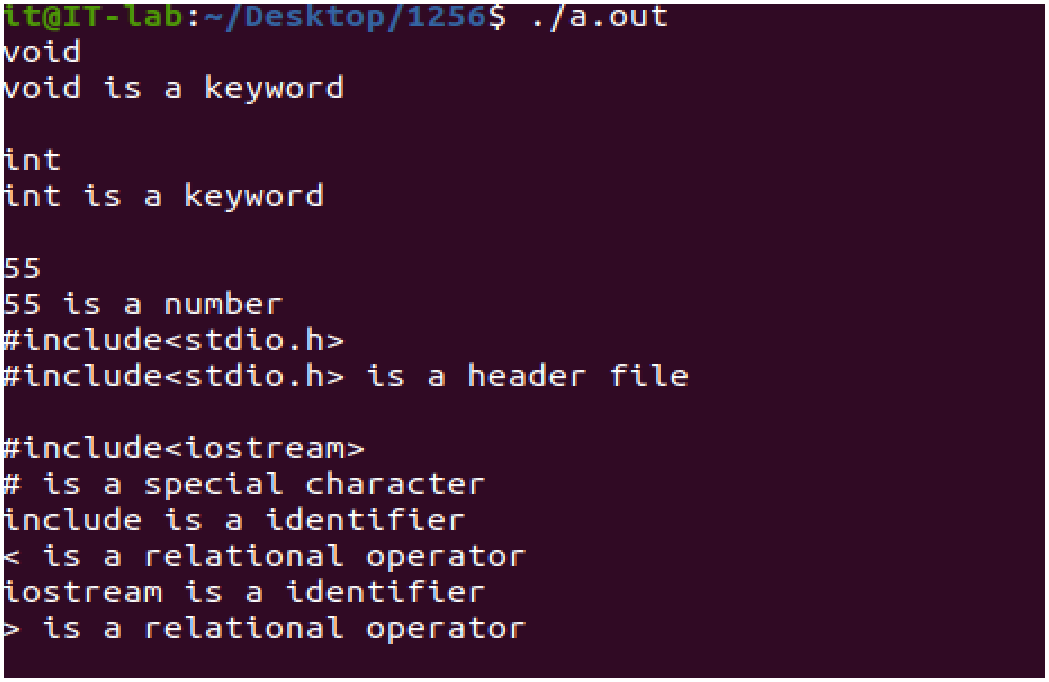
2) Write a LEX Program to scan reserved word & Identifiers of C Language.

```
%%
"if" |
"else" |
"while" |
"for" |
"do" |
"switch" |
"goto" |
"break" |
"case" |
"const" |
"float" |
"double" |
"int" |
"long" |
"short" |
"signed" |
"unsigned" |
"register" |
"typedef" |
"return" |
"enum" |
"sizeof" |
"static" |
"struct" |
"union" |
"void" |
"main" |
"continue" |
"default" |
"printf" |
"scanf" {printf("%s is a keyword\n",yytext);}
[a-zA-Z][a-zA-Z0-9_]* {printf("%s is a identifier\n",yytext);}
[0-9]+ {printf("%s is a number",yytext);}
[+/*%\-] {printf("%s is an arithmetic operator\n",yytext);}
[< >][=]? |
[!=][=] {printf("%s is a relational operator\n",yytext);}
[&][&]|[|][|]|[!] {printf("%s is a logical operator\n",yytext);}
[&][~][|][^][<<][>>] {printf("%s is a bitwise operator\n",yytext);}
[,] {printf("%s is a seperator\n",yytext);}
[;] {printf("%s is a terminator\n",yytext);}
[()] {printf("%s is a braces\n",yytext);}
[{}] {printf("%s is a paranthesis\n",yytext);}
[[]] {printf("%s is a squarebrace\n",yytext);}
"%d" |
"%c" |
"%s" |
"%f" {printf("%s is a formatspecifier\n",yytext);}
```

```
[/][/][a-zA-Z]* {printf("%s is a line comment\n",yytext);}
[/][*][a-z0-9A-Z #()%&+-<>!~;".,]*[*][/] {printf("%s is a block comment\n",yytext);}
["][a-z0-9A-Z #()%&+-<>!~;".,]*["] {printf("%s is a string\n",yytext);}
[@#$] {printf("%s is a special character\n",yytext);}
"\\n" {printf("%s is a new line character\n",yytext);}
"#include<stdio.h>" |
"#include<stdbool.h>" |
"#include<string.h>" |
"#include<math.h>" {printf("%s is a header file\n",yytext);}
%%
int yywrap(void){}
int main()
{
yylex();
return 0;
}
```

Input: LEX specification files for the token.

Output: Produces the source code for the Lexical Analyzer with the name lex.yy.c and displays the tokens from an input file.



```
it@IT-lab:~/Desktop/1256$ ./a.out
void
void is a keyword

int
int is a keyword

55
55 is a number
#include<stdio.h>
#include<stdio.h> is a header file

#include<iostream>
# is a special character
include is a identifier
< is a relational operator
iostream is a identifier
> is a relational operator
```

Aim: Design a Predictive Parser.

3) Implement Predictive Parsing algorithm for the following grammer:

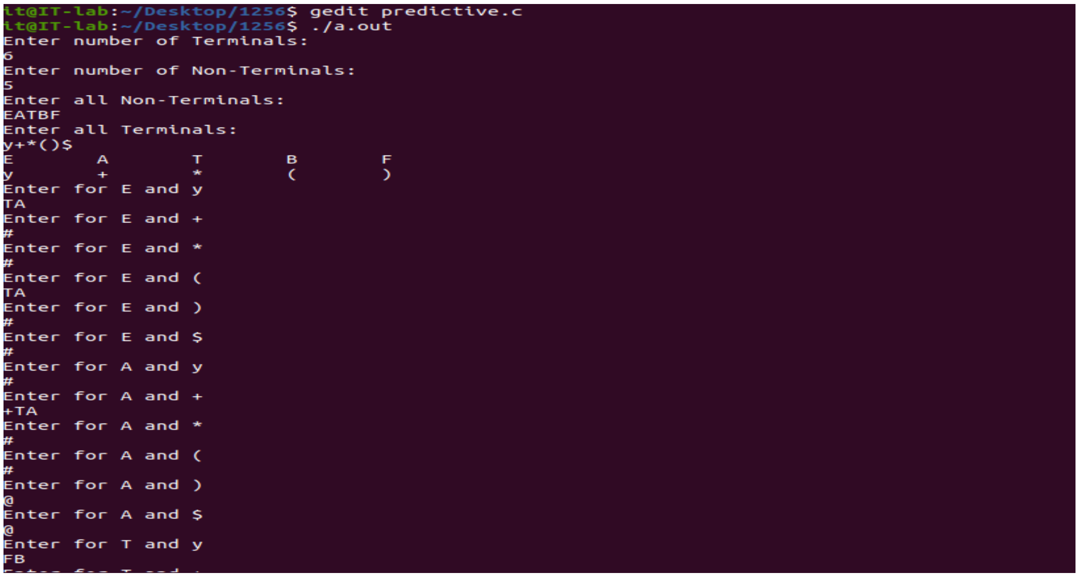
G: { E-> TE’ , E’ -> +TE’ | 0, T-> FT’ , T’-> *FT’|0 , F-> (E) | id }.

```
 #include <stdio.h>
 #include <ctype.h>
 #include <string.h>
 #include <stdlib.h>
 char table[10][10][10], nter[10], ter[10];
 char inp[20], stack[20];
 int nut, nun, i = 0, top = 0;
 int get_nter(char);
 int get_ter(char);
 void replace(char, char);
void main() {
 int i, j;
 printf("Enter number of Terminals:\n");
 scanf("%d", &nut);
 printf("Enter number of Non-Terminals:\n");
 scanf("%d", &nun);
 printf("Enter all Non-Terminals:\n");
 scanf("%s", nter);
 printf("Enter all Terminals:\n");
 scanf("%s", ter);
 for (i = 0; i < nut; i++)
printf("%c\t", nter[i]);
printf("\n");
for (j = 0; j < nun; j++)
 printf("%c\t", ter[j]);
 printf("\n");
 for (i = 0; i < nun; i++)
for (j = 0; j < nut; j++) {
 printf("Enter for %c and %c \n", nter[i], ter[j]);
 scanf("%s", table[i][j]);
}
for (j = 0; j < nut; j++)
printf("\t %c", ter[j]);
 printf("\n");
 for (i = 0; i < nun; i++) {
printf("%c", nter[i]);
for (j = 0; j < nut; j++) {
 printf("\t %s", table[i][j]);
}
printf("\n");
 }
 printf("Enter the string to parse:\n");
 scanf("%s", inp);
 stack[top++] = '$';
 stack[top++] = nter[0];
 i = 0;
while(1) {
if ((stack[top - 1] == '$') && (inp[i] == '$')) {
 printf("String Accepted\n");
 return;
}
else if (!isupper(stack[top - 1])) {
 if (stack[top - 1] == inp[i]) {
i++;
top--;
}
 else {
printf("Error not accepted\n");
return;
 }
}
else {
 replace(stack[top - 1], inp[i]);
}
 }
}
```

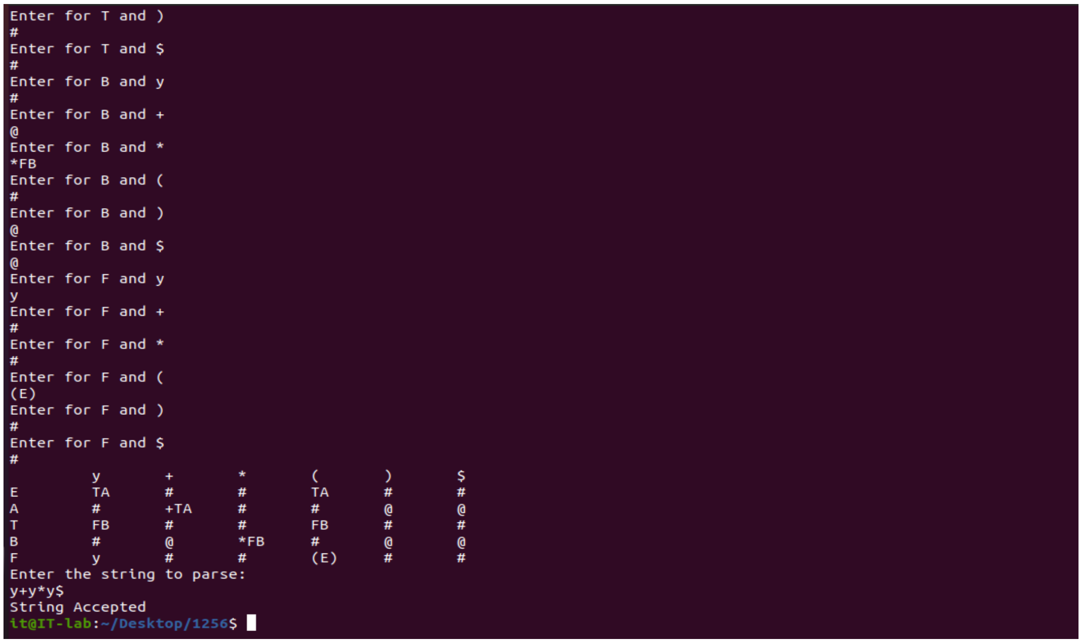
```
 int get_nter(char x) {
  int a;
  for (a = 0; a < nun; a++)
 if (x == nter[a])
  return a;
  return 100;
 }
  int get_ter(char x) {
   int a;
   for (a = 0; a < nut; a++)
  if (x == ter[a])
   return a;
   return 100;
  }
  void replace (char NT, char T) {
   int in1, it1, len;
   char str[10];
   in1 = get_nter(NT);
   it1 = get_ter(T);
   if ((in1 != 100) && (it1 != 100)) {
  strcpy(str, table[in1][it1]);
  if(strcmp(str, "#") == 0) {
  printf("Error\n");
  exit(0);}
if (strcmp(str, "@") == 0)
top--;
 else {
top--;
len = strlen(str);
len--;
do {
 stack[top++] = str[len--];
} while (len >= 0);
 }
}
else {
 printf("Not Valid\n");
}}
```

Input:Grammer G: { E-> TE’ , E’ -> +TE’ | 0, T-> FT’ , T’-> *FT’|0 , F-> (E) | id}

Output : Check whether string is accepted or not.



```
it@IT-lab:~/Desktop/1256$ gedit predictive.c
it@IT-lab:~/Desktop/1256$ ./a.out
Enter number of Terminals:
6
Enter number of Non-Terminals:
5
Enter all Non-Terminals:
EATBF
Enter all Terminals:
y+*()$
E       A       T       B       F
y       +       *       (       )
Enter for E and y
TA
Enter for E and +
#
Enter for E and *
#
Enter for E and (
TA
Enter for E and )
#
Enter for E and $
#
Enter for A and y
#
Enter for A and +
+TA
Enter for A and *
#
Enter for A and (
#
Enter for A and )
@
Enter for A and $
@
Enter for T and y
FB
```



```
Enter for T and )
#
Enter for T and $
#
Enter for B and y
#
Enter for B and +
@
Enter for B and *
*FB
Enter for B and (
#
Enter for B and )
@
Enter for B and $
@
Enter for F and y
y
Enter for F and +
#
Enter for F and *
#
Enter for F and (
(E)
Enter for F and )
#
Enter for F and $
#
        y       +       *       (       )       $
E       TA      #       #       TA      #       #
A       #       +TA     #       #       @       @
T       FB      #       #       FB      #       #
B       #       @       *FB     #       @       @
F       y       #       #       (E)     #       #
Enter the string to parse:
y+y*y$
String Accepted
it@IT-lab:~/Desktop/1256$ 
```

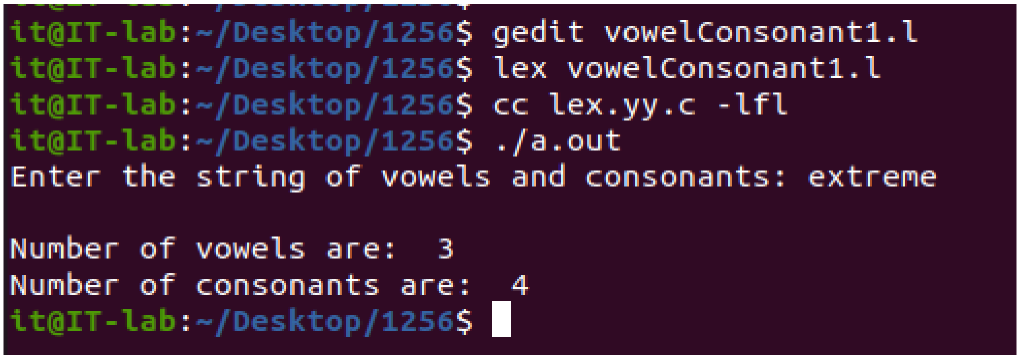
Aim: Count the number of vowels and consonants in a given string.

4) Lex Program to count the number of vowels and consonants in a given string.

```
%{
   int vow_count=0;
   int const_count =0;
%}
 %%
[aeiouAEIOU] {vow_count++;}
[a-zA-Z] {const_count++;}
%%
int yywrap(){}
int main()
{
   printf("Enter the string of vowels and consonants:");
   yylex();
   printf("Number of vowels are:  %d\n", vow_count);
   printf("Number of consonants are:  %d\n", const_count);
   return 0;
}
```

Input:String consisting vowels and consonants are given.

Output:Prints number of vowels and consonants for the given string.



```
it@IT-lab:~/Desktop/1256$ gedit vowelConsonant1.l
it@IT-lab:~/Desktop/1256$ lex vowelConsonant1.l
it@IT-lab:~/Desktop/1256$ cc lex.yy.c -lfl
it@IT-lab:~/Desktop/1256$ ./a.out
Enter the string of vowels and consonants: extreme

Number of vowels are:  3
Number of consonants are:  4
it@IT-lab:~/Desktop/1256$ 
```

Aim: Predictive parser (Recursive)

5)Implement predictive parser in recursion.

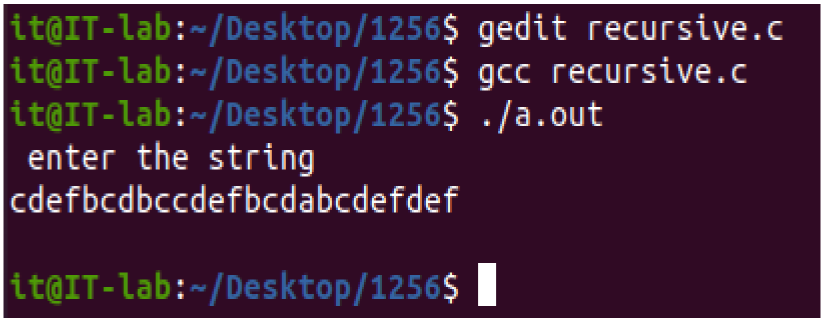
```
#include<stdio.h>
#include<string.h>
int S();
int A();
int B();
int C();
int D();
char input[100];
int i;
void main()
{
printf(" enter the string\n");
scanf("%s",input);
    if(S()==1)
    {
        printf(" \n string is accepted");
    }
    else
    {
        printf("\n string is not accepted\n");
    }
}
int S()
{
    if(A()==1)
        if(B()==1)
           if(input[i]=='a')
           {
               i++;
           if(input[i]=='b')
           {
               i++;
if(C()==1)
if(D()==1)
if(input[i]=='e')
{
```

```
i++;
if(input[i]=='f')
{
i++;
return 1;
}
else return 0;
}
else return 0;
else return 0;
else return 0;
}
else return 0;
}
else return 0;
else return 0;
else return 0;
}
int A()
{
if(B()==1)
if(input[i]=='b')
{
i++;
if(input[i]=='c')
{
i++;
return 1;
}
else return 0;
}
else return 0;
else return 0;
}
```

```
int B()
{
if(C()==1)
if(input[i]=='b')
{
i++;
if(input[i]=='c')
{
i++;
if(input[i]=='d')
{
i++;
return 1;
}
else return 0;
}
else return 0;
}
else return 0;
else return 0;
}
int C()
{
if(input[i]=='c')
{
i++;
if(input[i]=='d')
{
i++;
if(input[i]=='e')
{
i++;
if(input[i]=='f')
{
i++;
return 1;
}
else return 0;
}
else return 0;
}
else return 0;
}
else return 0;
}
int D()
{
if(input[i]=='d')
{
i++;
return 1;
}
else return 0;
}
```

Input:A string is entered as input

Output: It prints whether the string is accepted or not.



```
it@IT-lab:~/Desktop/1256$ gedit recursive.c
it@IT-lab:~/Desktop/1256$ gcc recursive.c
it@IT-lab:~/Desktop/1256$ ./a.out
 enter the string
cdefbcdbccdefbcdabcdefdef

it@IT-lab:~/Desktop/1256$ 
```

Aim: Write a program that can analyze the contents of a text file and count the number of characters, words, spaces, and lines in the file.

6)Program to count the number of characters, words, spaces, end of lines in a given input file.

p1.l

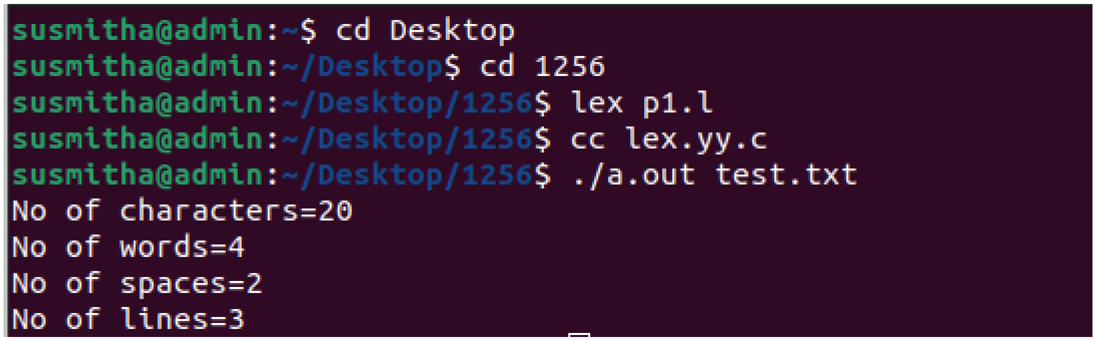
```
%{
#include<stdio.h>
int c = 0, w = 0, s = 0, l = 0;
%}
WORD [^ \t\n,\.:]+
EOL [\n]
BLANK [ ]
%%
{WORD} {w++; c = c + yyleng;}
{BLANK} {s++;}
{EOL} {l++;}
. {c++;}
%%

int yywrap() {
   return 1;
}

int main(int argc, char *argv[]) {

   if (argc != 2) {

      printf("Usage: <./a.out> <sourcefile>\n");

      exit(0);

   }

   yyin = fopen(argv[1], "r");

   yylex();

   printf("No of characters=%d\nNo of words=%d\nNo of spaces=%d\nNo of
lines=%d\n", c, w, s, l);

   return 0;

}
```

Test.txt:
hello world

hello world

Input: An input file consisting of text**.**

Output: Prints the number of characters,words,spaces and lines for the given input text file.



```
susmitha@admin:~$ cd Desktop
susmitha@admin:~/Desktop$ cd 1256
susmitha@admin:~/Desktop/1256$ lex p1.l
susmitha@admin:~/Desktop/1256$ cc lex.yy.c
susmitha@admin:~/Desktop/1256$ ./a.out test.txt
No of characters=20
No of words=4
No of spaces=2
No of lines=3
```

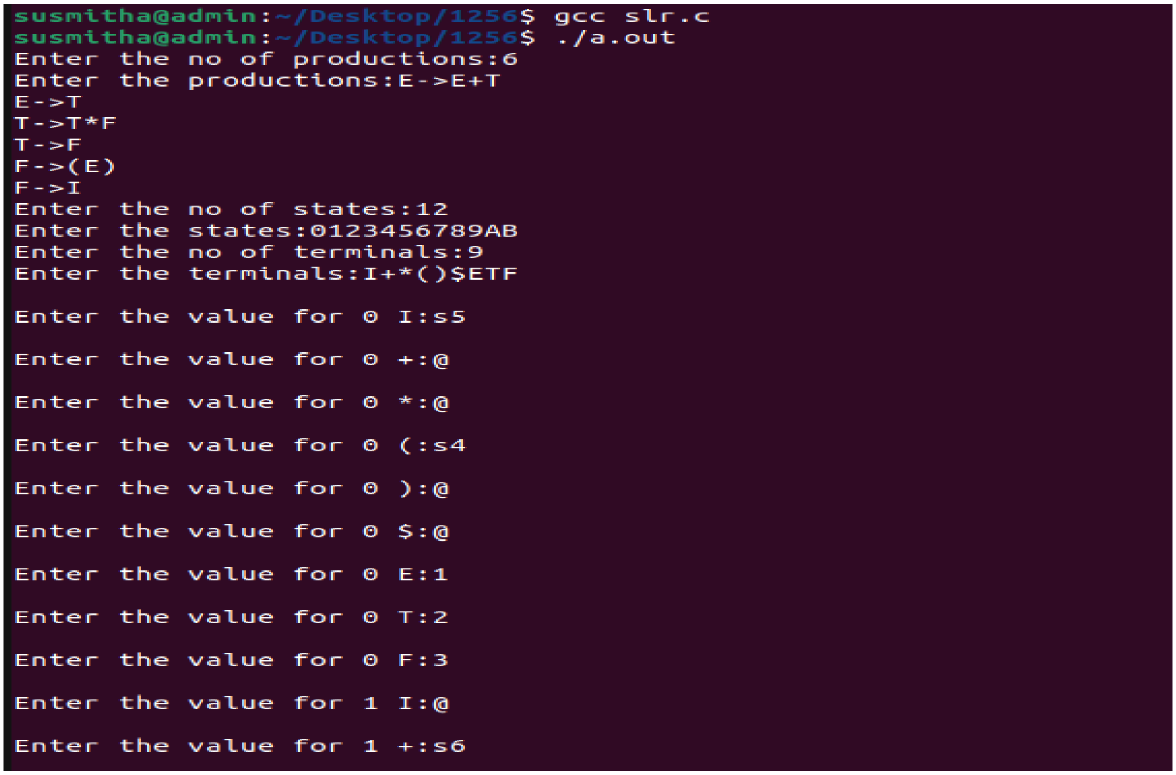
Aim :Implement SLR(1) Parsing algorithm

7)Program to implement SLR(1) parsing algorithm.

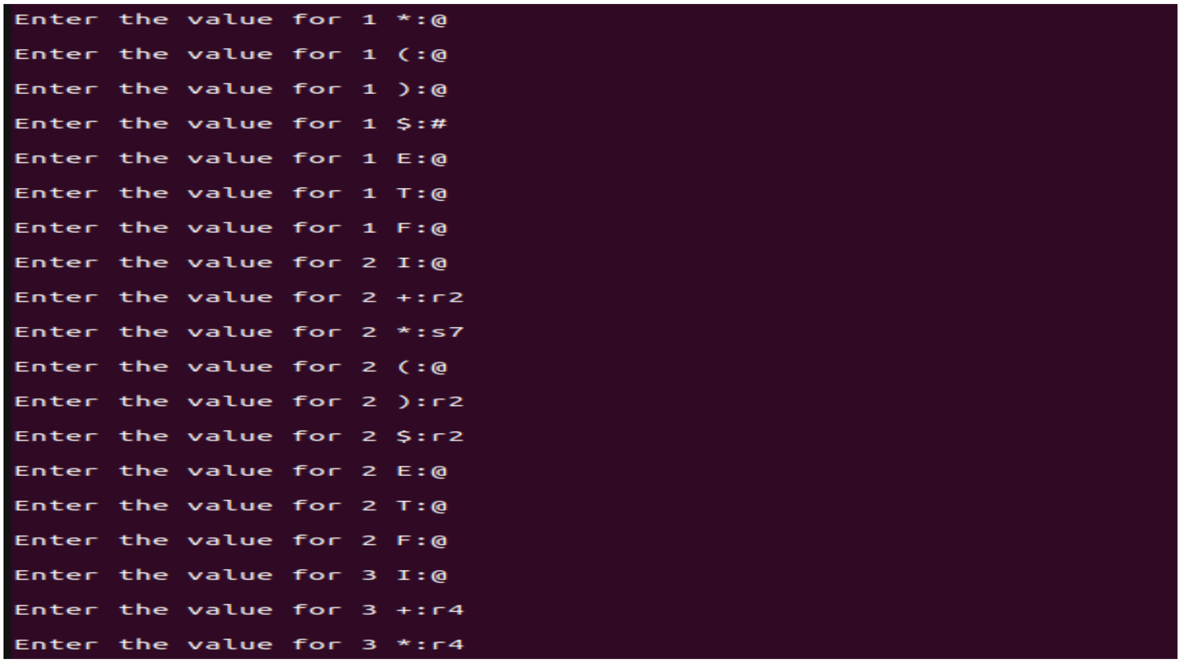
```
#include<stdio.h>
#include<string.h>

void main() {
        char table[20][20][20],ter[20],stack[20],ip[20],st1[20],pro[20][20],num;
int  i,j,t,k,top=0,st,col,row,pop,np,no,len;for(i=0;i<20;i++){
ter[i] = (char)0; stack[i] = (char)0;ip[i] =
(char)0;
st1[i] = (char)0; for(j=0;j<20;j++) {
      pro[i][j] = (char)0;for(k=0;k<20;k++) {
              table[i][j][k] = (char)0;
      }
}

}
printf("Enter the no of productions:");scanf("%d",&np);
printf("Enter the productions:");for(i=0;i<np;i++) {
scanf("%s",pro[i]);
}
printf("Enter the no of states:");
scanf("%d",&st);
printf("Enter the states:");
scanf("%s",st1);
printf("Enter the no of terminals:");
scanf("%d",&t);

  printf("Enter the terminals:");

  scanf("%s",ter);
for(i=0;i<st;i++){
      for(j=0;j<t;j++) {
              printf("\nEnter the value for %c%c:",st1[i],ter[j]);
                scanf("%s",table[i][j]);
      }
}
```

```
printf("\nSLR TABLE:\n");
for(i=0;i<t;i++) {
      printf("\t%c",ter[i]);
}
for(i=0;i<st;i++) {
       printf("\n\n%c",st1[i]);
for(j=0;j<t;j++) {
      printf("\t%s",table[i][j]);

}
}
stack[top] = '0';
printf("\nEnter the input string:");scanf("%s",ip);
i = 0;
printf("\n\nSTACK\t\t\tINPUTSTRING\t\t\tACTION\n");
printf("\n%s\t\t%s\t\t",stack,ip); while(i<=strlen(ip)) {
        for(j=0;j<st;j++) { if(stack[top] == st1[j])
                        col = j;
        }
        for(j=0;j<t;j++) {
                if(ip[i] == ter[j]) {row = j;
                }
        }
        if(stack[top]== '@') {top--;
                continue;
        }
        if((stack[top] == '1')&&(ip[i] == '$')){printf("\nString
                accepted\n"); break;
        }
        else if(table[col][row][0] == 's') {top++;
                stack[top] = ter[row];top++;
                stack[top] = table[col][row][1];i++;
                int p =(int) table[col][row][1];p = p-48;
                printf("   %s\t\t",table[col][row]);
          //printf("hello"); printf("%c%c",ter[row],table[col][row][1]);
        }
        else if(table[col][row][0] == 'r'){no
                =table[col][row][1];
                no = no-48;no=no-
                1;
                len = strlen(pro[no]);
        //printf("\n prod no-%d length is-%d\n",no,len);len = len-3;
                pop = 2*len; printf("POP
                %d",pop);for(j=0;j<pop;j++) {
                        top = top -1;
                }
                top++;
                stack[top] = pro[no][0];
                k = top;k = k-1;
                printf("   Push %c",pro[no][0]);
                for(j=0;j<st;j++) {
                        if(stack[k] == st1[j]){col = j;
                        }
                } k++;
                for(j=0;j<t;j++){
                        if(stack[k] == ter[j]) {
                                row = j;
                        }
                }
                top++;
                stack[top] = table[col][row][0];
                printf("%c",table[col][row][0]);
                }
                else{
                        printf("\nError\nThe string not accepted");
                        break;
                 }
                 printf("\n"); for(j=0;j<=top;j++) {
                  printf("%c",stack[j]);
                  }
                  printf("\t \t"); for(j=i;j<strlen(ip);j++) {
                  printf("%c",ip[j]);
                  }
                  printf("\t \t");
                  } }
```

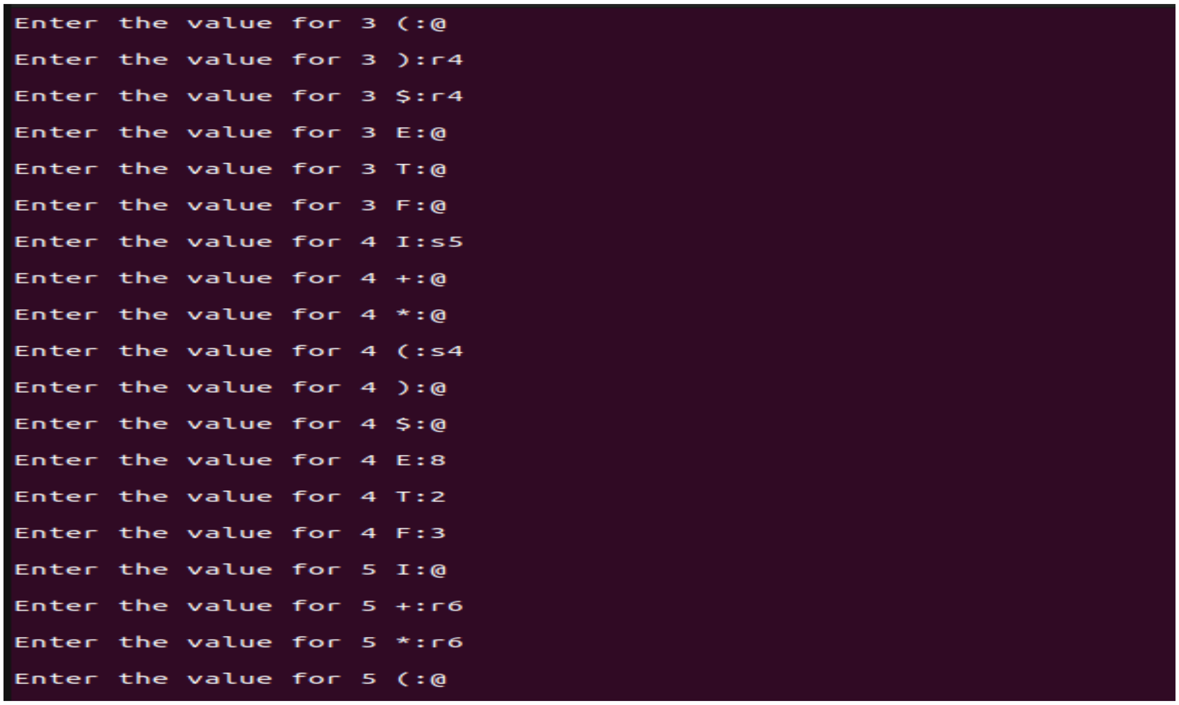
Input:



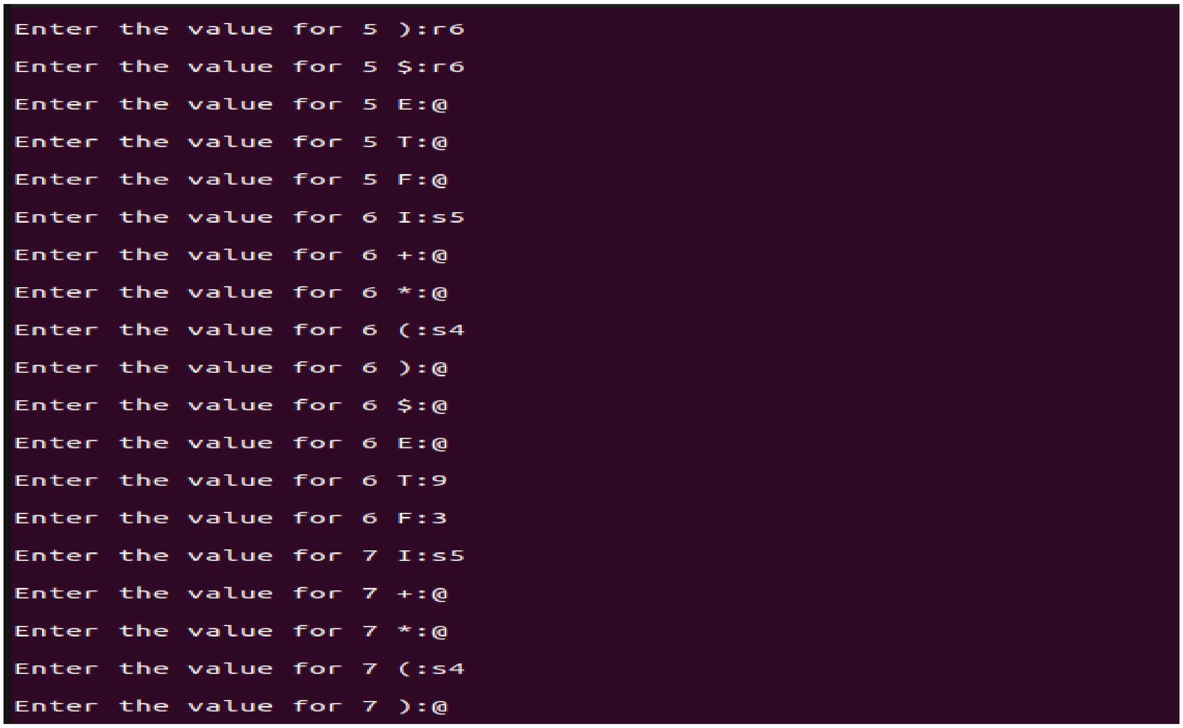
```
susmitha@admin:~/Desktop/1256$ gcc slr.c
susmitha@admin:~/Desktop/1256$ ./a.out
Enter the no of productions:6
Enter the productions:E->E+T
E->T
T->T*F
T->F
F->(E)
F->I
Enter the no of states:12
Enter the states:0123456789AB
Enter the no of terminals:9
Enter the terminals:I+*()$ETF

Enter the value for 0 I:s5

Enter the value for 0 +:@

Enter the value for 0 *:@

Enter the value for 0 (:s4

Enter the value for 0 ):@

Enter the value for 0 $:@

Enter the value for 0 E:1

Enter the value for 0 T:2

Enter the value for 0 F:3

Enter the value for 1 I:@

Enter the value for 1 +:s6
```



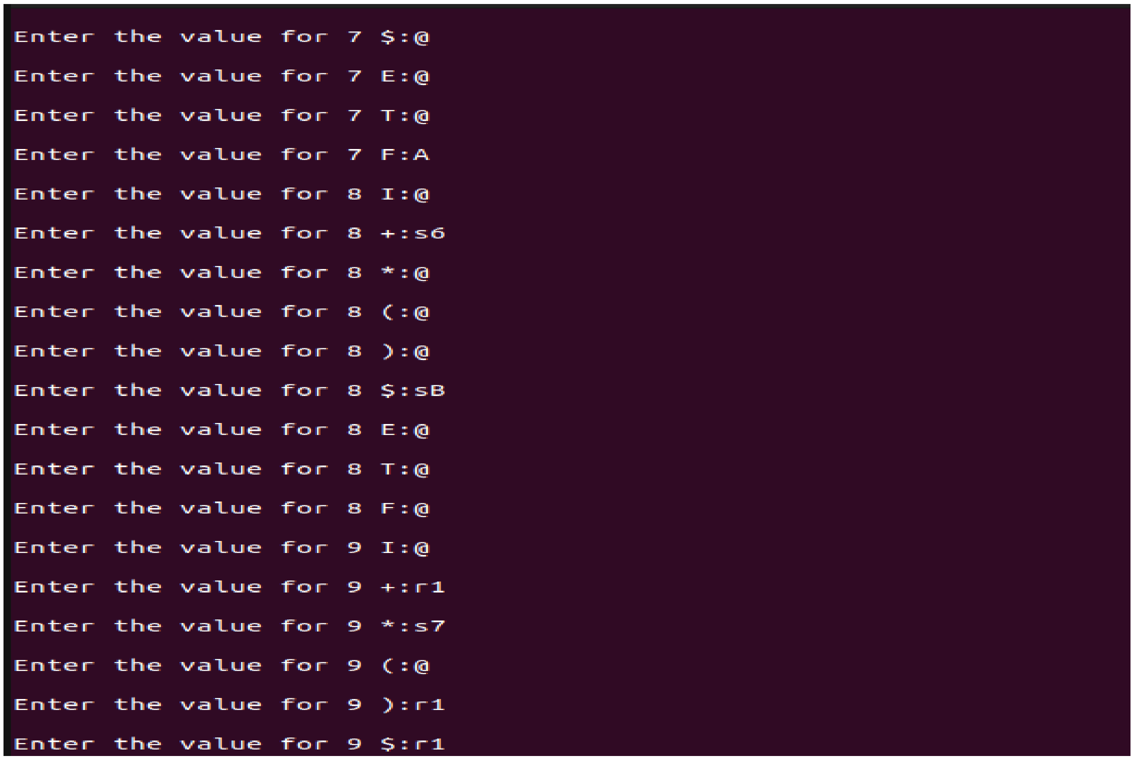
```
Enter the value for 1 *:@
Enter the value for 1 (:@
Enter the value for 1 ):@
Enter the value for 1 $:#
Enter the value for 1 E:@
Enter the value for 1 T:@
Enter the value for 1 F:@
Enter the value for 2 I:@
Enter the value for 2 +:r2
Enter the value for 2 *:s7
Enter the value for 2 (:@
Enter the value for 2 ):r2
Enter the value for 2 $:r2
Enter the value for 2 E:@
Enter the value for 2 T:@
Enter the value for 2 F:@
Enter the value for 3 I:@
Enter the value for 3 +:r4
Enter the value for 3 *:r4
```



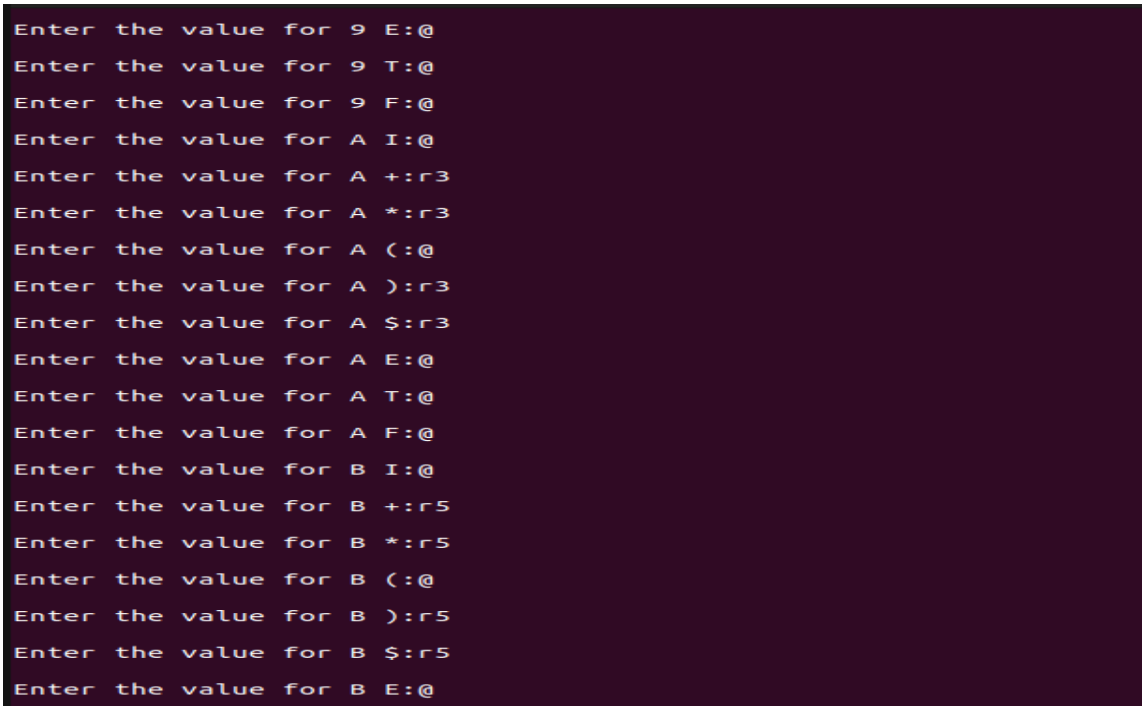
```
Enter the value for 3 (:@
Enter the value for 3 ):r4
Enter the value for 3 $:r4
Enter the value for 3 E:@
Enter the value for 3 T:@
Enter the value for 3 F:@
Enter the value for 4 I:s5
Enter the value for 4 +:@
Enter the value for 4 *:@
Enter the value for 4 (:s4
Enter the value for 4 ):@
Enter the value for 4 $:@
Enter the value for 4 E:8
Enter the value for 4 T:2
Enter the value for 4 F:3
Enter the value for 5 I:@
Enter the value for 5 +:r6
Enter the value for 5 *:r6
Enter the value for 5 (:@
```



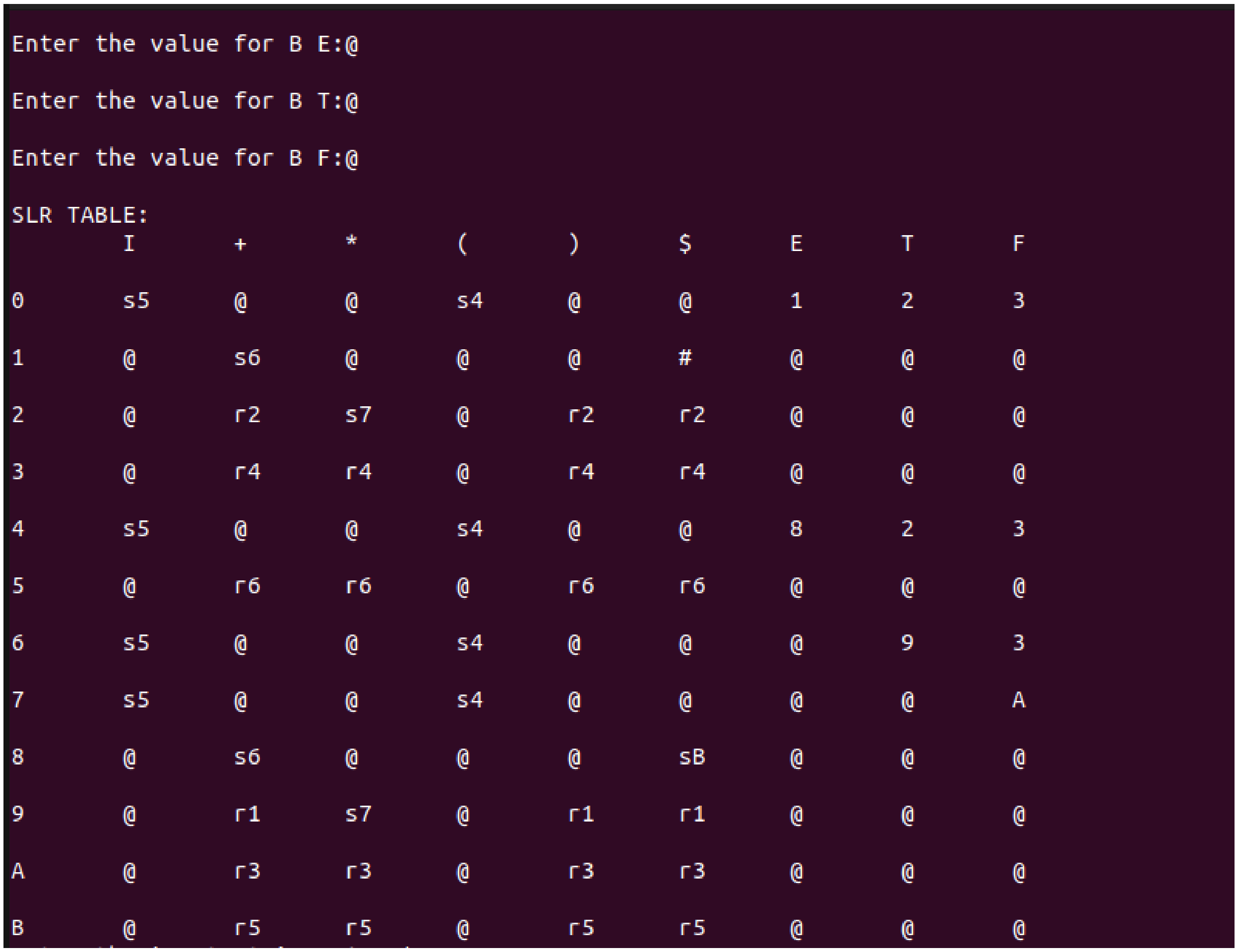
```
Enter the value for 5 ):r6
Enter the value for 5 $:r6
Enter the value for 5 E:@
Enter the value for 5 T:@
Enter the value for 5 F:@
Enter the value for 6 I:s5
Enter the value for 6 +:@
Enter the value for 6 *:@
Enter the value for 6 (:s4
Enter the value for 6 ):@
Enter the value for 6 $:@
Enter the value for 6 E:@
Enter the value for 6 T:9
Enter the value for 6 F:3
Enter the value for 7 I:s5
Enter the value for 7 +:@
Enter the value for 7 *:@
Enter the value for 7 (:s4
Enter the value for 7 ):@
```



```
Enter the value for 7 $:@
Enter the value for 7 E:@
Enter the value for 7 T:@
Enter the value for 7 F:A
Enter the value for 8 I:@
Enter the value for 8 +:s6
Enter the value for 8 *:@
Enter the value for 8 (:@
Enter the value for 8 ):@
Enter the value for 8 $:sB
Enter the value for 8 E:@
Enter the value for 8 T:@
Enter the value for 8 F:@
Enter the value for 9 I:@
Enter the value for 9 +:r1
Enter the value for 9 *:s7
Enter the value for 9 (:@
Enter the value for 9 ):r1
Enter the value for 9 $:r1
```

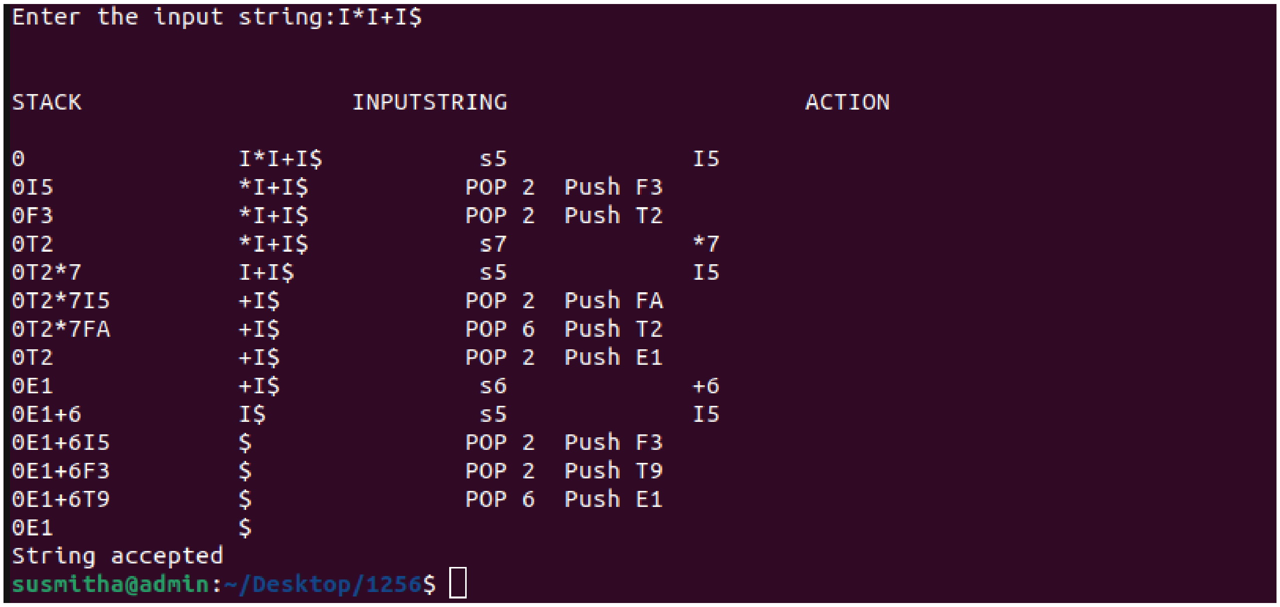


```
Enter the value for 9 E:@
Enter the value for 9 T:@
Enter the value for 9 F:@
Enter the value for A I:@
Enter the value for A +:r3
Enter the value for A *:r3
Enter the value for A (:@
Enter the value for A ):r3
Enter the value for A $:r3
Enter the value for A E:@
Enter the value for A T:@
Enter the value for A F:@
Enter the value for B I:@
Enter the value for B +:r5
Enter the value for B *:r5
Enter the value for B (:@
Enter the value for B ):r5
Enter the value for B $:r5
Enter the value for B E:@
```



```
Enter the value for B E:@

Enter the value for B T:@

Enter the value for B F:@

SLR TABLE:
        I       +       *       (       )       $       E       T       F

0       s5      @       @       s4      @       @       1       2       3

1       @       s6      @       @       @       #       @       @       @

2       @       r2      s7      @       r2      r2      @       @       @

3       @       r4      r4      @       r4      r4      @       @       @

4       s5      @       @       s4      @       @       8       2       3

5       @       r6      r6      @       r6      r6      @       @       @

6       s5      @       @       s4      @       @       @       9       3

7       s5      @       @       s4      @       @       @       @       A

8       @       s6      @       @       @       sB      @       @       @

9       @       r1      s7      @       r1      r1      @       @       @

A       @       r3      r3      @       r3      r3      @       @       @

B       @       r5      r5      @       r5      r5      @       @       @
```

OUTPUT:



```
Enter the input string:I*I+I$


STACK                   INPUTSTRING                     ACTION

0               I*I+I$           s5            I5
0I5             *I+I$           POP 2  Push F3
0F3             *I+I$           POP 2  Push T2
0T2             *I+I$            s7            *7
0T2*7           I+I$             s5            I5
0T2*7I5         +I$             POP 2  Push FA
0T2*7FA         +I$             POP 6  Push T2
0T2             +I$             POP 2  Push E1
0E1             +I$              s6            +6
0E1+6           I$               s5            I5
0E1+6I5         $               POP 2  Push F3
0E1+6F3         $               POP 2  Push T9
0E1+6T9         $               POP 6  Push E1
0E1             $
String accepted
susmitha@admin:~/Desktop/1256$
```