/**********
Name : .
PRN : .
Batch : B3
Class : L.Y. B.Tech.
Sub : CDL
Aim : Design a lexical analyzer for given language and the
Lexical analyzer should ignore redundant spaces, tabs and new lines.
**********/

```
#include <stdio.h>  //header file
#include <string.h> //header file
int main()          // main function
{
   int c = 1, i, j = 0; // declaring variables
   char file_name[20], s[20];
   scanf("%s", file_name);
   FILE *fp;
   fp = fopen(file_name, "r");
   if (fp == NULL) // if condition
   {
      printf("Failed to open the file");
   }
   else // else condition
   {
      do
      {
         fscanf(fp, "%s", s);
         if (s[0] == '#' && s[7] == 'e')
         {
            printf("%s is a Header file\n", s);
         }
         else if (!strcmp(s, "+") || !strcmp(s, "-") || !strcmp(s, "*") || !strcmp(s, "/") ||
!strcmp(s, ">") || !strcmp(s, "<") ||
                  !strcmp(s, "=") || !strcmp(s, "<=") ||
                  !strcmp(s, ">=") || !strcmp(s, "!") || !strcmp(s, "!=") ||
                  !strcmp(s, "==") ||
                  !strcmp(s, "++") || !strcmp(s, "--") ||
                  !strcmp(s, "&") || !strcmp(s, "|") || !strcmp(s, "||") || !strcmp(s, "&&") ||
                  !strcmp(s, "%"))
         {
            printf("%s is an Operator\n", s);
         }
```

```
        else if (!strcmp(s, "auto") || !strcmp(s, "break") || !strcmp(s, "case") || !strcmp(s,
"char") || !strcmp(s, "const") ||
                !strcmp(s, "continue") || !strcmp(s, "default") ||
                !strcmp(s, "do") || !strcmp(s, "double") || !strcmp(s, "else") ||
                !strcmp(s, "enum") || !strcmp(s, "extern") ||
                !strcmp(s, "float") || !strcmp(s, "for") || !strcmp(s, "goto") ||
                !strcmp(s, "if") || !strcmp(s, "int") ||
                !strcmp(s, "long") || !strcmp(s, "register") || !strcmp(s, "return") ||
                !strcmp(s, "short") || !strcmp(s, "signed") ||
                !strcmp(s, "sizeof") || !strcmp(s, "static") || !strcmp(s, "struct") ||
                !strcmp(s, "switch") || !strcmp(s, "typedef") ||
                !strcmp(s, "union") || !strcmp(s, "unsigned") || !strcmp(s, "void") ||
                !strcmp(s, "volatile") || !strcmp(s, "while"))
        {
            printf("%s is a Keyword\n", s);
        }
        else if (!strcmp(s, "\""))
        {
            fscanf(fp, "%s", s);
            while (strcmp(s, "\""))
            {
                printf("%s ", s);
                fscanf(fp, "%s", s);
            }
            printf("is an Argument\n");
        }
        else if (!strcmp(s, "scanf") || !strcmp(s, "printf") ||
                !strcmp(s, "main"))
        {
            printf("%s is an Identifier\n", s);
        }
        else if (!strcmp(s, ",") || !strcmp(s, ";") || !strcmp(s, "{") || !strcmp(s, "}") ||
!strcmp(s, "(") || !strcmp(s, ")"))
        {
            continue;
        }
        else
        {
            printf("%s is an Identifier\n", s);
        }
    } while (c != EOF);
  }
  fclose(fp);
```

```
    return 0;
}
```

INPUT FILE:- abc.c

```
#include<stdio.h>
#include<stdlib.h>
int main ( )
{
int a , b , c , ch ;
printf ( " \n ENTER numbers as an argument " ) ;
scanf ( " %d%d " , & a , & b ) ;
printf ( " \nMENU\n1.ADD\n2.Sub\n3.MUL " ) ;
scanf ( " %d " , & ch ) ;
c = a + b ;
if ( ch == 0 )
{
printf ( " bye " ) ;
}
}
```

/**********OUTPUT**********

```
abc.c
#include<stdio.h> is a Header file
#include<stdlib.h> is a Header file
int is a Keyword
main is an Identifier
int is a Keyword
a is an Identifier
b is an Identifier
c is an Identifier
%d%d is an Argument
& is an Operator
a is an Identifier
& is an Operator
b is an Identifier
printf is an Identifier
\nMENU\n1.ADD\n2.Sub\n3.MUL is an Argument
scanf is an Identifier
%d is an Argument
& is an Operator
ch is an Identifier
c is an Identifier
= is an Operator
```

a is an Identifier
+ is an Operator
b is an Identifier
if is a Keyword
ch is an Identifier
== is an Operator
0 is an Identifier
printf is an Identifier
bye is an Argument
****

PROBLEMS TERMINAL ··· Code

```
PS A:\7th sem1\CD\practicals\CD Pratical\CD Pratical>
cd "a:\7th sem1\CD\practicals\CD Pratical\CD Pratical\
" ; if ($?) { gcc A1.c -o A1 } ; if ($?) { .\A1 }
abc.c
#include<stdio.h> is a Header file
#include<stdlib.h> is a Header file
int is a Keyword
main is an Identifier
int is a Keyword
a is an Identifier
b is an Identifier
c is an Identifier
%d%d is an Argument
& is an Operator
a is an Identifier
& is an Operator
b is an Identifier
printf is an Identifier
\nMENU\n1.ADD\n2.Sub\n3.MUL is an Argument
scanf is an Identifier
%d is an Argument
& is an Operator
ch is an Identifier
c is an Identifier
= is an Operator
a is an Identifier
+ is an Operator
```

```
\nMENU\n1.ADD\n2.Sub\n3.MUL is an Argument
scanf is an Identifier
%d is an Argument
& is an Operator
ch is an Identifier
c is an Identifier
= is an Operator
a is an Identifier
+ is an Operator
b is an Identifier
if is a Keyword
ch is an Identifier
== is an Operator
0 is an Identifier
printf is an Identifier
bye is an Argument
```

```
/*************************************************************
Name : .
PRN : .
Class : L.Y.B.TECH.
Batch : B3
Sub : CDL
Aim : Write a C program to identify whether a given line is comment
or not.
*************************************************************/

#include<stdio.h> //header file
void main() //main function
{
    char source_file[30],ch,ch1,ch2,ch3,ch4;
    int i=0,j=0;
    FILE *source; //file pointer
    printf("\nEnter the File Name:");
    scanf("%s",source_file);
    source=fopen(source_file,"r");
    if(source==NULL)
    {
        printf("File does not exists");
    }
    /*
    While Loop
    */
    while((ch=getc(source))!=EOF)
    {
        if(ch=='/')
        {
            ch1=getc(source);
            if(ch1=='/' || ch1=='*')
            {
            if(ch1=='/')
            {
                i++;
                printf("\nSingle Line Comment : %d",i);
                printf("\n%c%c",ch,ch1);
                while((ch2=getc(source))!='\n')
                {
                    printf("%c",ch2);
                }
                printf("\n");
```

```
            }
        else
        {
            j++;
            printf("\nMultiline comment : %d",j);
            printf("\n%c%c",ch,ch1);
            while((ch3=getc(source))!='*')
            {
                printf("%c",ch3);
            }
            printf("%c",ch3);
            if((ch4=getc(source))=='/')
                printf("%c\n",ch4);
        }
        }
        }
    }
    /*
    End Of While Loop
    */
}
```

```
/************INPUT FILE: asm.c
#include<stdio.h>
#include<stdlib.h>
int main( )
{
int a , b , c , ch ;
printf ( " \n Enter the two numbers " ) ;
scanf( " %d%d " , & a , & b ) ;
printf ( " \nMENU\n1.ADD\n2.Sub\n3.MUL " ) ;
scanf ( " %d " , & ch ) ;
c = a + b ;
//adding two numbers
if ( ch == 0 )
{
printf ( " Goodbye " ) ;
}
}
*/
```

/*********OUTPUT*********

Enter the File Name:asm.c

Single Line Comment : 1
//adding two numbers
PS A:\7th sem1\CD\practicals\CD Pratical\CD Pratical> */

```
PROBLEMS    OUTPUT    TERMINAL    ···    Code

PS A:\7th sem1\CD\practicals\CD Pratical\CD Pratica> c
d "a:\7th sem1\CD\practicals\CD Pratical\CD Pratical\"
 ; if ($?) { gcc A2.c -o A2 } ; if ($?) { .\A2 }

Enter the File Name:asm.c

Single Line Comment : 1
//adding two numbers
PS A:\7th sem1\CD\practicals\CD Pratical\CD Pratical>
```

```
/*********************************************************
Name : .
PRN : .
Class : L.Y.B.TECH.
Batch : B3
Sub : CDL
Aim : Write a C program to recognize strings under
'a*','a*b+','abb'.
*********************************************************/
// Program:
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
void main()
{
    char s[20],c;
    int state=0,i=0;
    printf("\n Enter a string:");
    gets(s);
    while(s[i]!='\0')
    {
        switch(state)
        {
            case 0: c=s[i++];
                if(c=='a')
                    state=1;
                else if(c=='b')
                    state=2;
                else
                    state=6;
                break;
            case 1: c=s[i++];
                if(c=='a')
                    state=3;
                else if(c=='b')
                    state=4;
                else
                    state=6;
                break;
            case 2: c=s[i++];
                if(c=='a')
                    state=6;
                else if(c=='b')
```

```
                state=2;
            else
                state=6;
            break;
        case 3: c=s[i++];
            if(c=='a')
                state=3;
            else if(c=='b')
                state=2;
            else
                state=6;
            break;
        case 4: c=s[i++];
            if(c=='a')
                state=6;
            else if(c=='b')
                state=5;
            else
                state=6;
            break;
        case 5: c=s[i++];
            if(c=='a')
                state=6;
            else if(c=='b')
                state=2;
            else
                state=6;
            break;
        case 6: printf("\n %s is not recognised.",s);
            exit(0);
    }
  }
  if((state==1) ||(state==3))
    printf("\n %s is accepted under rule 'a*'",s);
  else if((state==2)||(state==4))
    printf("\n %s is accepted under rule 'a*b+'",s);
  else if(state==5)
    printf("\n %s is accepted under rule 'abb'",s);
  else
    printf("\n String not accepted by automata.");
}
```

/**********OUTPUT**********

```
PROBLEMS    OUTPUT    TERMINAL    SQL CONSOLE    DEBUG CONSOLE

PS A:\7th sem1\CD\practicals\CD Pratical\CD Pratica> cd "a:\7th sem1\CD\practicals\CD
 if ($?) { gcc A3.c -o A3 } ; if ($?) { .\A3 }

 Enter a string:aaaaa

 aaaaa is accepted under rule 'a*'
PS A:\7th sem1\CD\practicals\CD Pratical\CD Pratical> cd "a:\7th sem1\CD\practicals\C
; if ($?) { gcc A3.c -o A3 } ; if ($?) { .\A3 }

 Enter a string:abbbb

 abbbb is accepted under rule 'a*b+'
PS A:\7th sem1\CD\practicals\CD Pratical\CD Pratical> cd "a:\7th sem1\CD\practicals\C
; if ($?) { gcc A3.c -o A3 } ; if ($?) { .\A3 }

 Enter a string:abb

 abb is accepted under rule 'abb'
PS A:\7th sem1\CD\practicals\CD Pratical\CD Pratical>
```

```
/*************************************************************
Name : .
PRN : .
Class : L.Y. Computer
Batch : B3
Sub : CDL
Aim :Write a C program to simulate lexical analyzer for validating
operators.
*************************************************************/

#include<stdio.h>
int main()
{
   char s[10];
   int c;
   do
   {
      printf("Enter any operator:");
      scanf("%s",s);
      switch(s[0])
      {
         case '<':if(s[1]=='=')
                  printf("\nless than or equal\n");
               else
                  printf("\nless than");
               break;
         case '>':if(s[1]=='=')
                  printf("\ngreater than or equal");
               else
                  printf("\ngreater than");
               break;
         case '+':if(s[1]=='+')
                  printf("\nunary increament operator");
               else
                  printf("\nadd is an binary arithmatic operator");
               break;
         case '-':if(s[1]=='-')
               printf("\nunary decreament operator");
               else
               printf("\nminus is an binary arithmatic operator");
               break;
         case '/':if(s[1]=='*')
                  printf("\nit is not an operator");
```

```
            else
                printf("\ndivision is an binary arithmatic operator");
            break;
        case '*':printf("\nmultiplication is an binary arithmatic operator");
            break;
        case '%':printf("\nmodulus is an arithmatic operator");
            break;
        case '!':if(s[1]=='=')
                printf("\nnot equal");
            else
                printf("\nbit not");
            break;
        case '=':if(s[1]=='=')
                printf("\nit is an comparison operator");
            else
                printf("\nassignment operator");
            break;
        case '&':if(s[1]=='&')
                printf("\nlogical AND");
            else
            printf("\nBitwise AND");
            break;
        case '|':if(s[1]=='|')
                printf("\nlogical OR");
            else
                printf("\nBitwise OR");
            break;
        case '~':printf("\nnegation operator");
            break;
        case '?':if(s[1]==':')
                printf("\nternary operator is an unary operator");
            else
                printf("\nnot an operator");
            break;
        default:printf("\nInvalid input!!");
            break;
    }
    printf("\nDo you want to continue 1/0\n");
    scanf("%d",&c);
  }while(c==1);
  return(0);
}
```

/********** OUTPUT **********/

```
PROBLEMS    TERMINAL    ...                Code

PS A:\7th sem1\CD\practicals\CD Pratical\CD Pratical>
cd "a:\7th sem1\CD\practicals\CD Pratical\CD Pratical\
" ; if ($?) { gcc A4.c -o A4 } ; if ($?) { .\A4 }
Enter any operator:+

add is an binary arithmatic operator
Do you want to continue 1/0
1
Enter any operator:-

minus is an binary arithmatic operator
Do you want to continue 1/0
1
Enter any operator:>

greater than
Do you want to continue 1/0
1
Enter any operator:/

division is an binary arithmatic operator
Do you want to continue 1/0
1
Enter any operator:*

multiplication is an binary arithmatic operator
Do you want to continue 1/0
```

```
Enter any operator:*

multiplication is an binary arithmatic operator
Do you want to continue 1/0
1
Enter any operator:<=

less than or equal

Do you want to continue 1/0
1
Enter any operator:>=

greater than or equal
Do you want to continue 1/0
0
PS A:\7th sem1\CD\practicals\CD Pratical\CD Pratical>
```

```
/*************************************************************
Name : .
PRN : .
Class : L.Y. Computer
Batch : B3
Sub : CDL
Aim : Simulate First and Follow of a Grammar.
*************************************************************/

#include <stdio.h>
#include<string.h>
int numOfProd;
char prods[10][10],f[10];
int m=0;
void first(char a);
void follow(char a);
int main()
{
    printf("\nEnter the number of Productions :- ");
    scanf("%d",&numOfProd);
    printf("\nEnter the Productions :- ");
    for(int i=0;i<numOfProd;i++)
        scanf("%s",prods[i]);
    int choice;
    char choi;
    do
    {
    m = 0;
    printf("\nEnter the element to find First and Follow :- ");
    getchar();
    scanf("%c",&choi);
    first(choi);
    printf("First(%c)={",choi);
    for(int i=0;i<m;i++)
    printf("%c",f[i]);
    printf("}\n");
    strcpy(f," ");
    m = 0;
    follow(choi);
    printf("Follow(%c)={",choi);
    for(int i=0;i<m;i++)
    printf("%c",f[i]);
    printf("}\n");
```

```
    printf("Do you want to continue?(1/0)");
    scanf("%d",&choice);
    }
    while(choice==1);
    return 0;
}
void first(char a)
{
    if((a>='a'&&a<='z')||a=='$')
    {
        f[m++] = a;
    }
    for(int k=0;k<numOfProd;k++)
    {
        if(prods[k][0]==a)
        {
            if(prods[k][2]=='$')
                follow(prods[k][0]);
            else if((prods[k][2]>='a'&&prods[k][2]<='z')||prods[k][2]=='$')
                f[m++] = prods[k][2];
            else
                first(prods[k][2]);
        }
    }
}
void follow(char a)
{
    if (prods[0][0] == a)
        f[m++] = '$';
    for (int i = 0; i < numOfProd; i++)
    {
        for (int j = 2; j < strlen(prods[i]); j++)
        {
            if (prods[i][j] == a)
            {
                if (prods[i][j + 1] != '\0')
                    first(prods[i][j + 1]);
                if (prods[i][j + 1] == '\0' && a != prods[i][0])
                    follow(prods[i][0]);
            }
        }
    }
}
```

OUTPUT:-

```
PS A:\7th sem1\CD\practicals\CD Pratical\CD Pratical> cd "a:\7t
cal\CD Pratical\" ; if ($?) { gcc A5.c -o A5 } ; if ($?) { .\A5

Enter the number of Productions :- 3

Enter the Productions :- S=CC
C=eC
C=d

Enter the element to find First and Follow :- S
First(S)={ed}
Follow(S)={$}
Do you want to continue?(1/0)1

Enter the element to find First and Follow :- C
First(C)={ed}
Follow(C)={ed$}
Do you want to continue?(1/0)1

Enter the element to find First and Follow :- d
First(d)={d}
Follow(d)={ed$}
Do you want to continue?(1/0)0
PS A:\7th sem1\CD\practicals\CD Pratical\CD Pratical>
```

```
/*************************************************************
Name : .
PRN : .
Class : L.Y. Computer
Batch : B3
Sub : CDL
Aim : Write a C program to implement operator precedence parsing.
*************************************************************/
```

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    char stack[20], ip[20], opt[10][10][1], ter[10];
    int i, j, k, n, top = 0, col, row;
    for (i = 0; i < 10; i++)
    {
        stack[i] = NULL;
        ip[i] = NULL;
        for (j = 0; j < 10; j++)
        {
            opt[i][j][1] = NULL;
        }
    }
    printf("Enter the no.of terminals:\n");
    scanf("%d", &n);
    printf("\nEnter the terminals:\n");
    for (i = 0; i < n; i++)
    {
        scanf("%s", &ter[i]);
    }
    printf("\nEnter the table values:\n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            printf("Enter the value for %c %c: ", ter[i], ter[j]);
            scanf("%s", opt[i][j]);
        }
    }
    printf("\n** OPERATOR PRECEDENCE TABLE **\n");
```

```
for (i = 0; i < n; i++)
{
    printf("\t%c", ter[i]);
}
printf("\n");
for (i = 0; i < n; i++)
{
    printf("\n%c", ter[i]);
    for (j = 0; j < n; j++)
    {
        printf("\t%c", opt[i][j][0]);
    }
}
stack[top] = '$';
printf("\nEnter the input string: ");
scanf("%s", ip);
i = 0;
printf("\nSTACK\t\t\tINPUT STRING\t\t\tACTION\n");
printf("\n%s\t\t\t%s\t\t\t", stack, ip);
while (i <= strlen(ip))
{
    for (k = 0; k < n; k++)
    {
        if (stack[top] == ter[k])
            col = k;
        if (ip[i] == ter[k])
            row = k;
    }
    if ((stack[top] == '$') && (ip[i] == '$'))
    {
        printf("\nString is accepted\n");
        break;
    }
    else if ((opt[col][row][0] == '<') || (opt[col][row][0] == '='))
    {
        stack[++top] = opt[col][row][0];
        stack[++top] = ip[i];
        printf("Shift %c", ip[i]);
        i++;
    }
    else
    {
        if (opt[col][row][0] == '>')
```

```
            {
                while (stack[top] != '<')
                {
                    --top;
                }
                top = top - 1;
                printf("Reduce");
            }
            else
            {
                printf("\nString is not accepted");
                break;
            }
        }
        printf("\n");
        for (k = 0; k <= top; k++)
        {
            printf("%c", stack[k]);
        }
        printf("\t\t\t");
        for (k = i; k < strlen(ip); k++)
        {
            printf("%c", ip[k]);
        }
        printf("\t\t\t");
    }
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    SQL CONSOLE    ...    Coc

Enter the no.of terminals:
4

Enter the terminals:
+
*
a
$

Enter the table values:
Enter the value for + +: >
Enter the value for + *: <
Enter the value for + a: <
Enter the value for + $: >
Enter the value for * +: >
Enter the value for * *: >
Enter the value for * a: <
Enter the value for * $: >
Enter the value for a +: >
Enter the value for a *: >
Enter the value for a a: =
Enter the value for a $: >
Enter the value for $ +: <
Enter the value for $ *: <
Enter the value for $ a: <
Enter the value for $ $: A
```

```
** OPERATOR PRECEDENCE TABLE **
        +       *       a       $

+       >       <       <       >
*       >       >       <       >
a       >       >       =       >
$       <       <       <       A
Enter the input string: a+a*a$

STACK                   INPUT STRING                    ACTION

$                       a+a*a$                  Shift a
$<a                     +a*a$                   Reduce
$                       +a*a$                   Shift +
$<+                     a*a$                    Shift a
$<+<a                   *a$                     Reduce
$<+                     *a$                     Shift *
$<+<*                   a$                      Shift a
$<+<*<a                 $                       Reduce
$<+<*                   $                       Reduce
$<+                     $                       Reduce
$                       $
String is accepted
PS A:\7th sem1\CD\practicals\CD Pratical\CD Pratical>
```

/**********
Name : .
PRN : .
Class : L.Y. Computer
Batch : B3
Sub : CDL
Aim : Write a C program to generate machine code from abstract syntax tree generated by the parser.
**********/

PROGRAM:

```
#include<stdio.h>
//#include<conio.h>
#include<string.h>
#include<stdlib.h>
struct quadraple
{
   int pos;
   char op;
   char arg1[5];
   char arg2[5];
   char result[5];
} quad[15];
int n=0;
void assignment(int);
void uminus(int );
void explore();
void codegen(char op[5],int);
char tuple[15][15];
int main(void)
{
   FILE *src;
   int nRetInd,i;
   char str[15];
   // clrscr();
   src=fopen("code.txt","r");
   fscanf(src,"%s",str);
   while(!feof(src))
   {
      strcpy(tuple[n++],str);
      fscanf(src,"%s",str);
   }
   printf("INPUT:\nIntermiate codes:\n");
   for(i=0;i<n;i++)
      printf("%s\n",tuple[i]);
```

```
    explore();
    // getch();
    // clrscr();
    printf("OUTPUT:\n");
    printf("Quadruple: \n");
    printf("pos\topr\targ1\targ2\tresult\n");
    for(i=0;i<n;i++)
        printf("\n%d\t%c\t%s\t%s\t%s",quad[i].pos,quad[i].op,quad[i].arg1,quad[i].arg2,qua
d[i].result);
    i=0;
    printf("\n\ncode generated :\n");
    while(i<n)
    {
        if(quad[i].op=='+')
            codegen("ADD",i);
        if(quad[i].op=='=')
            assignment(i);
        if(quad[i].op=='-')
            if(!strcmp(quad[i].arg2,"\0"))
                uminus(i);
            else
                codegen("SUB",i);
        if(quad[i].op=='*')
            codegen("MUL",i);
        if(quad[i].op=='/')
            codegen("DIV",i);
        i++;
    }
    // getch();
    _fcloseall();
    return 0;
}
void codegen(char op[5],int t)
{
    char str[25];
    printf("MOV %s,R0\n",quad[t].arg1);
    printf("%s %s,R0\n",op,quad[t].arg2);
    printf("MOV R0,%s\n",quad[t].result);
}
void assignment(int t)
{
    char str[25];
    printf("MOV %s,%s\n",quad[t].arg1,quad[t].result);
}
void uminus(int t)
```

```
{
    char str[25];
    printf("MOV R0,0\n");
    printf("SUB %s,R0\n",quad[t].arg1);
    printf("MOV R0,%s\n",quad[t].result);
}
void explore()
{
    int i,j,t,t1,t2;
    for(i=0;i<n;i++)
    {
        quad[i].pos=i;
        for(j=0,t=0;j<strlen(tuple[i])&&tuple[i][j]!='=';j++)
        {
            quad[i].result[t++]=tuple[i][j];
        }
        t1=j;
        quad[i].result[t]='\0';
        if(tuple[i][j]=='=')
        {
            quad[i].op='=';
        }
        if(tuple[i][j+1]=='+'||tuple[i][j+1]=='-'||tuple[i][j+1]=='*'||tuple[i][j+1]=='/')
        {
            quad[i].op=tuple[i][j+1];
            t1=j+1;
        }
        for(j=t1+1,t=0;j<strlen(tuple[i])&&tuple[i][j]!='+'&&tuple[i][j]!='-
'&&tuple[i][j]!='*'&&tuple[i][j]!='/';j++)
        {
            quad[i].arg1[t++]=tuple[i][j];
        }
        t2=j;
        quad[i].arg1[t]='\0';
        if(tuple[i][j]=='+'||tuple[i][j]=='-'||tuple[i][j]=='*'||tuple[i][j]=='/')
        {
            quad[i].op=tuple[i][j];
        }
        for(j=t2+1,t=0;j<strlen(tuple[i]);j++)
        {
            quad[i].arg2[t++]=tuple[i][j];
        }
        quad[i].arg2[t]='\0';
    }
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    SQL CONSOLE    ...    Code

PS A:\7th sem1\CD\practicals\CD Pratical\CD Pratical> cd "a:\7th s
racticals\CD Pratical\CD Pratical\" ; if ($?) { gcc B3.c -o B3 } ;
 { .\B3 }
INPUT:
Intermiate codes:
t1=b*c
t2=d*f
t3=t1+a
t4=t3+t2
t5=t4+g
OUTPUT:
Quadruple:
pos     opr     arg1    arg2    result

0       *       b       c       t1
1       *       d       f       t2
2       +       t1      a       t3
3       +       t3      t2      t4
4       +       t4      g       t5

code generated :
MOV b,R0
MUL c,R0
MOV R0,t1
MOV d,R0
MUL f,R0
MOV R0,t2
MOV t1,R0
ADD a,R0
MOV R0,t3
MOV t3,R0
ADD t2,R0
MOV R0,t4
MOV t4,R0
ADD g,R0
MOV R0,t5
PS A:\Ashish 7th sem\CD\practicals\CD Pratical\CD Pratical>
Reader Optimized   Ln 20, Col 2   Spaces: 4   UTF-8   CRLF   C   Go Live   Win32
```

/**********
Name : .
PRN : .
Class : L.Y. Computer
Batch : B3
Sub : CDL
Aim : Write a C program to check whether a string belongs to grammar or not.
**********/

PROGRAM:

```
#include<string.h>
#include<stdio.h>
int main() {
   int c;
   char string[20];
   int state=0,count=0;
   //printf("\n The string must begin with a and terminate with b"); printf("\n The
Given Grammar is:\n");
   printf("\tS->aS \n\tS->bR \n\tR->aR \n\tR->bS \n\tR->b\n"); printf("Enter a String
To Be Checked: ");
   scanf("%s",string);

   while(string[count]!='\0')
   {
      switch(state)
      {
         case 0: if (string[count]=='a') state=1;
                 else state=3;
                 break;
         case 1: if (string[count]=='a') state=1;
                 else if(string[count]=='b') state=2;
                 else state=3;
                 break;
         case 2: if (string[count]=='b') state=2;
                 else state=3;
                 break;
         default: break;
      }
      count++;
      if(state==3)
         break;
   }
```

```
    if(state==2)
        printf("\nString is accepted.\n");
    else
        printf("\nString is not accepted.\n");
    return 0;
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    SQL CONSOLE    ...    Code

PS A:\7th sem1\CD\practicals\CD Pratical\CD Pratical> cd "a:\7th sem1\CD\p
racticals\CD Pratical\CD Pratical\" ; if ($?) { gcc B4.c -o B4 } ; if ($?)
 { .\B4 }
        S->aS
        S->bR
        R->aR
        R->bS
        R->b
Enter a String To Be Checked: abaa

String is not accepted.
PS A:\7th sem1\CD\practicals\CD Pratical\CD Pratical>
```

/**********
Name : .
PRN : .
Class : L.Y. Computer
Batch : B3
Sub : CDL
Aim : Implementation of Deterministic Finite Automata.
**********/

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int main (int argc, char **argv)
{
   int a, b, s, x, i, len, Q[100][100], initial, final;
   char str[100], c[2];
   printf("\nEnter total number of inputs: ");
   scanf("%d", &i);
   printf("\nEnter total number of states: ");
   scanf("%d", &s);
   printf("\nEnter initial state for DFA: ");
   scanf("%d", &initial);
   printf("\nEnter final state for DFA: ");
   scanf("%d", &final);
   printf("\n\n Initial State: {Q%d}", initial);
   printf("\n Final State: {Q%d}", final);
   printf("\n Set of Finite States: {");
   for (a = 0; a < s; a++)
   {
      printf("Q%d", a);
      if(a < s-1)
         printf(", ");
   }
   printf("}");
   printf("\n Set of Inputs : {");
   for(a = 0; a < i; a++)
   {
      printf("%d ",a);
      if(a < i-1)
         printf(", ");
   }
   printf("}\n\n");
```

```
    printf(" Enter the transition table INPUT:\n");
    printf("Transition-> state");
    for(a = 0; a < s; a++)
    {
        for(b = 0; b < i; b++)
        {
            printf("\n Q%d, %d -> ", a, b);
            scanf("%d", &Q[a][b]);
        }
    }
    do
    {
        printf("\nEnter the string to check: ");
        scanf("%s", str);
        len = strlen(str);
        c[1] = '\0';
        x = initial;
        printf("\n -> Q0");
        for(a = 0; a < len; a++)
        {
            c[0] = str[a];
            x = Q[x][atoi(c)];
            printf(" --%d--> Q%d", atoi(c), x);
        }
        if(x == final)
            printf("\n\n***[String Accepted for this grammar]***\n\n");
        else
            printf("\n\n###[String Not Accepted]###\n\n");
        printf("Do you want to check another string [Yes = 1 / No = 0]: ");
        scanf("%d", &a);
    }while(a);
    return 0;
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    SQL CONSOLE    ...    Code

PS A:\7th sem1\CD\practicals\CD Pratical\CD Pratical> cd "a:\7th sem
1\CD\practicals\CD Pratical\CD Pratical\" ; if ($?) { gcc B5.c -o B5
 } ; if ($?) { .\B5 }

Enter total number of inputs: 2

Enter total number of states: 3

Enter initial state for DFA: 0

Enter final state for DFA: 2


 Initial State: {Q0}
 Final State: {Q2}
 Set of Finite States: {Q0, Q1, Q2}
 Set of Inputs : {0 , 1 }
 Enter the transition table INPUT:
Transition-> state
 Q0, 0 -> 1

 Q0, 1 -> 2
```

```
 Enter the transition table INPUT:
Transition-> state
 Q0, 0 -> 1

 Q0, 1 -> 2

 Q1, 0 -> 1

 Q1, 1 -> 2

 Q2, 0 -> 1

 Q2, 1 -> 2

Enter the string to check: 110101

 -> Q0 --1--> Q2 --1--> Q2 --0--> Q1 --1--> Q2 --0--> Q1 --1--> Q2

***[String Accepted for this grammar]***

Do you want to check another string [Yes = 1 / No = 0]: 0
PS A:\7th sem1\CD\practicals\CD Pratical\CD Pratical>
```

/**********
Name : .
PRN : .
Class : L.Y. Computer
Batch : B3
Sub : CDL
Aim : Implementation of shift reduce parsing algorithm.
**********/

PROGRAM:

```
#include<stdio.h> //including header files
#include<stdlib.h>
#include<string.h>
char ip_sym[15],stack[15]; //declaring global variables
int ip_ptr=0,st_ptr=0,len,i;
char temp[2],temp2[2];
char act[15];
void check(); //declaring function
void main() //defining main
{
   printf("\n\t\t SHIFT REDUCE PARSER\n");
   printf("\n GRAMMER\n"); //defining grammar
   printf("\n E->E+E\n E->E/E");
   printf("\n E->E*E\n E->a/b");
   printf("\n Enter the input symbol:\t");
   gets(ip_sym); //take input string
   printf("\n\t Stack implementation table");
   printf("\n Stack\t\t Input symbol\t\t Action");
   //construct parse table
   printf("\n______\t\t _____________\t\t ______\n");
   printf("\n $\t\t%s$\t\t\t--",ip_sym);
   strcpy(act,"shift ");
   temp[0]=ip_sym[ip_ptr];
   temp[1]='\0';
   strcat(act,temp);
   len=strlen(ip_sym);
   for(i=0;i<=len-1;i++) //using for loop
   {
      stack[st_ptr]=ip_sym[ip_ptr]; //inserting in stack
      stack[st_ptr+1]='\0';
      ip_sym[ip_ptr]=' ';
      ip_ptr++;
      printf("\n $%s\t\t%s$\t\t\t%s",stack,ip_sym,act);
```

```
        strcpy(act,"shift ");
        temp[0]=ip_sym[ip_ptr];
        temp[1]='\0';
        strcat(act,temp);
        check(); //checking with grammar
        st_ptr++;
    }
    st_ptr++;
    check();
}
void check() //function definition
{
    int flag=0;
    temp2[0]=stack[st_ptr];
    temp2[1]='\0';
    if((!strcmp(temp2,"a"))||(!strcmp(temp2,"b")))
    {
        stack[st_ptr]='E';
        if(!strcmp(temp2,"a")) //checking for third production
            printf("\n $%s\t\t%s$\t\t\tE->a",stack, ip_sym);
        else
            printf("\n $%s\t\t%s$\t\t\tE->b\n",stack,ip_sym);
        flag=1;
    }
    if((!strcmp(temp2,"+"))||(strcmp(temp2,"*"))||(!strcmp(temp2,"/")))
    {
        flag=1;
    }
    if((!strcmp(stack,"E+E"))||(!strcmp(stack,"E\E"))||(!strcmp(stack,"E*E")))
    {
        strcpy(stack,"E");
        st_ptr=0;
        if(!strcmp(stack,"E+E")) //using if condition
            printf("\n $%s\t\t%s$\t\t\tE->E+E",stack,ip_sym);
        else if(!strcmp(stack,"E\E"))
            printf("\n $%s\t\t %s$\t\t\tE->E\E",stack,ip_sym);
        else
            printf("\n $%s\t\t%s$\t\t\tE->E*E",stack,ip_sym);
        flag=1;
    }
    if(!strcmp(stack,"E")&&ip_ptr==len)
    {
        printf("\n $%s\t\t%s$\t\t\tACCEPT\n",stack,ip_sym);
```

```
        exit(0);
    }
    if(flag==0)
    {
        printf("\n%s\t\t\t%s\t\t reject\n",stack,ip_sym);
        exit(0);
    }
    return;
}
```

OUTPUT:

```
PROBLEMS    OUTPUT    TERMINAL    SQL CONSOLE    DEBUG CO

PS A:\7th sem1\CD\practicals\CD Pratical\CD Pratica
 if ($?) { gcc B6.c -o B6 } ; if ($?) { .\B6 }

                    SHIFT REDUCE PARSER

 GRAMMER

 E->E+E
 E->E/E
 E->E*E
 E->a/b
 Enter the input symbol:        a+b

          Stack implementation table
 Stack            Input symbol            Action
------            ------------            ------

 $               a+b$                    --
 $a               +b$                    shift a
 $E               +b$                    E->a
 $E+               b$                    shift +
 $E+b               $                    shift b
 $E+E               $                    E->b
```

```
if ($?) { gcc B6.c -o B6 } ; if ($?) { .\B6 }

                    SHIFT REDUCE PARSER

GRAMMER

E->E+E
E->E/E
E->E*E
E->a/b
Enter the input symbol:        a/b

        Stack implementation table
Stack           Input symbol            Action
------          ------------            ------

$              a/b$                     --
$a              /b$                     shift a
$E              /b$                     E->a
$E/              b$                     shift /
$E/b              $                     shift b
$E/E              $                     E->b
PS A:\7th sem1\CD\practicals\CD Pratical\CD Pratical>
```