// write a program to check whether the given input is keyword or not

```
#include <stdio.h>
#include <string.h>
int main() {
        char inputKeyword[100];
        char keyword[32][10]={
             "auto","double","int","struct","break","else","long",
             "switch","case","enum","register","typedef","char",
             "extern","return","union","const","float","short",
             "unsigned","continue","for","signed","void","default",
             "goto","sizeof","voltile","do","if","static","while"
           } ;
        printf("Enter your keyword:\n");
        scanf("%s", inputKeyword);
        int flag=0,i;
        for(i = 0; i < 32; i++) {
             if(strcmp(inputKeyword,keyword[i])==0) {
                flag=1;
             }
        }
        if(flag==1)
             printf("%s is a keyword",inputKeyword);
        else
             printf("%s is not a keyword",inputKeyword);
        return 0;
}
```

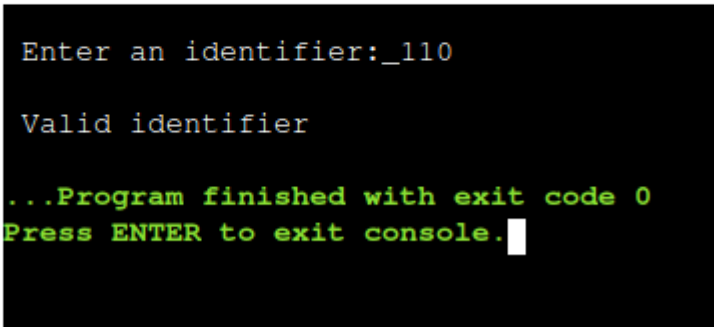
Output:

```
Enter your keyword:
case
case is a keyword

...Program finished with exit code 0
Press ENTER to exit console.
```

// c program to check whether the input is identifier or not

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>

int main()
{
	char a[10];
	int flag, i=1;
	printf("\n Enter an identifier:");
	scanf("%s", a);
	if(isalpha(a[0]) || a[0] =='_')
		flag=1;
	else
		printf("\n Not a valid identifier");
	while(a[i]!='\0')
	{
	if(!isdigit(a[i])&&!isalpha(a[i]))
	{
		flag=0;
		break;
	}
		i++;
	}
	if(flag==1)
		printf("\n Valid identifier");
	return 0;
}
```

Output:



// C program to check if the input is comment

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

int main()
{
   char com [30];
   int i=2,a=0;
   printf("\n Enter Text : ");
   scanf("%s",com);
   if(com[0]=='/' )
   {
      if(com[1]=='/')
         printf("\n It is a Comment.");

      else if(com [1]=='*')
      {
         for(i=2;i<strlen(com);i++)
         {
            if(com[i]=='*'&&com[i+1]=='/')
            {
               printf("\n It is a Comment.");
               a=1;
               break;
            }
            else continue;
         }
         if(a==0)
            printf("\n It is Not a Comment.");
      }
      else
         printf("\n It is Not a Comment.");
   }
   else
   printf("\n It is Not a Comment.");
   return 0;
}
```

Output:

```
 Enter Text : /**p

 It is Not a Comment.

...Program finished with exit code 0
Press ENTER to exit console.
```

// lexical analyzer in c

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
int isKeyword(char buffer[])
{
   char keywords[32][10] = {"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 i, flag = 0;
                             for(i = 0; i < 32; ++i)
                             {
                                if(strcmp(keywords[i], buffer) == 0)
                                {    flag = 1;
                                      break;

                                }
                             }
   return flag;
}
int main()
{
   char ch, buffer[15], operators[] = "+-*/%=";
   FILE *fp;  int i,j=0;
   fp = fopen("aa.txt","r");
   if(fp == NULL)
   {
      printf("error while opening the file\n");
      exit(0);
   }
   while((ch = fgetc(fp)) != EOF)
   {      for(i = 0; i < 6; ++i)
      {
         if(ch == operators[i])
         printf("%c is operator\n", ch);

      }
      if(isalnum(ch))
      {
         buffer[j++] = ch;

      }
      else if((ch == ' ' || ch == '\n') && (j != 0))
      {
```

```
            buffer[j] = '\0';
            j = 0;
            if(isKeyword(buffer) == 1)
                printf("%s is keyword\n", buffer);
            else
                printf("%s is identifier\n", buffer);
        }
    }
    fclose(fp);
    return 0;
}
```

Input File:

```
void main(){
        int a,b;
        a = b*a;
}
```

Output:

```
void is keyword
main is identifier
int is keyword
a is identifier
b is identifier
a is identifier
= is operator
b is identifier
* is operator
a is identifier
```

// c program to create dfa that accepts string ending with 101

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
#define max 100
int main() {
   char str[max],f='a';
   int i;
   printf("enter the string to be checked: ");
   scanf("%s",str);
   for(i=0;str[i]!='\0';i++) {
      switch(f) {
         case 'a': if(str[i]=='1') f='b';
            else if(str[i]=='0') f='a';
         break;
         case 'b': if(str[i]=='1') f='a';
            else if(str[i]=='0') f='c';
         break;
         case 'c': if(str[i]=='1') f='d';
            else if(str[i]=='0') f='a';
         break;
          case 'd': if(str[i]=='1') f='d';
            else if(str[i]=='0') f='a';
         break;
      }
   }
   if(f=='d')
      printf("\nString is accepted", f);
   else printf("\nString is not accepted", f);
      return 0;
}
```

output:

```
enter the string to be checked: 1000101

String is accepted

...Program finished with exit code 0
Press ENTER to exit console.
```

// c program to create dfa that accepts string ending with aba

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
#define max 100
int main() {
   char str[max],f='A';
   int i;
   printf("enter the string to be checked: ");
   scanf("%s",str);
   for(i=0;str[i]!='\0';i++) {
      switch(f) {
         case 'A': if(str[i]=='a') f='B';
            else if(str[i]=='b') f='A';
         break;
         case 'B': if(str[i]=='a') f='A';
            else if(str[i]=='b') f='C';
         break;
         case 'C': if(str[i]=='a') f='D';
            else if(str[i]=='b') f='A';
         break;
          case 'D': if(str[i]=='a') f='D';
            else if(str[i]=='b') f='A';
         break;
      }
   }
   if(f=='D')
      printf("\nString is accepted", f);
   else printf("\nString is not accepted", f);
      return 0;
}
```

Output:

```
enter the string to be checked: abaaabb

String is not accepted

...Program finished with exit code 0
Press ENTER to exit console.
```

// c program that removes left recursion from given grammar

```
#include<stdio.h>
#include<string.h>
#define SIZE 10
int main () {
    char non_terminal;
    char beta,alpha;
    int num;
    char production[10][SIZE];
    int index=3; /* starting of the string following "->" */
    printf("Enter Number of Production : ");
    scanf("%d",&num);
    printf("Enter the grammar as E->E-A :\n");
    for(int i=0;i<num;i++){
        scanf("%s",production[i]);
    }
    for(int i=0;i<num;i++){
        printf("\nGRAMMAR : : : %s",production[i]);
        non_terminal=production[i][0];
        if(non_terminal==production[i][index]) {
            alpha=production[i][index+1];
            printf(" is left recursive.\n");
            while(production[i][index]!=0 && production[i][index]!='|')
                index++;
            if(production[i][index]!=0) {
                beta=production[i][index+1];
                printf("Grammar without left recursion:\n");
                printf("%c->%c%c\'",non_terminal,beta,non_terminal);
                printf("\n%c\'->%c%c\'|E\n",non_terminal,alpha,non_terminal);
            }
            else
                printf(" can't be reduced\n");
        }
        else
            printf(" is not left recursive.\n");
        index=3;
    }
    return 0;
}
```

Output:

```
Enter Number of Production : 4
Enter the grammar as E->E-A :
E->EA|A
A->AT|a
T=a
E->i

GRAMMAR : : : E->EA|A is left recursive.
Grammar without left recursion:
E->AE'
E'->AE'|E

GRAMMAR : : : A->AT|a is left recursive.
Grammar without left recursion:
A->aA'
A'->TA'|E

GRAMMAR : : : T=a is not left recursive.

GRAMMAR : : : E->i is not left recursive.


...Program finished with exit code 0
Press ENTER to exit console.
```

//C program that computes first of given grammar

```
#include<stdio.h>
#include<string.h>
#define SIZE 10
#include<stdio.h>
#include<ctype.h>
void FIRST(char[],char );
void addToResultSet(char[],char);
int numOfProductions;
char productionSet[10][10];
int main()
{
   int i;
   char choice;
   char c;
   char result[20];
   printf("How many number of productions ? :");
   scanf(" %d",&numOfProductions);
   for(i=0;i<numOfProductions;i++)//read production string eg: E=E+T
   {
      printf("Enter productions Number %d : ",i+1);
      scanf(" %s",productionSet[i]);
   }
   do
   {
      printf("\n Find the FIRST of  :");
      scanf(" %c",&c);
      FIRST(result,c); //Compute FIRST; Get Answer in 'result' array
      printf("\n FIRST(%c)= { ",c);
      for(i=0;result[i]!='\0';i++)
      printf(" %c ",result[i]);       //Display result
      printf("}\n");
       printf("press 'y' to continue : ");
      scanf(" %c",&choice);
   }
   while(choice=='y'||choice =='Y');
   return 0;
}
/*
 *Function FIRST:
 *Compute the elements in FIRST(c) and write them
 *in Result Array.
 */
void FIRST(char* Result,char c)
{
```

```
    int i,j,k;
    char subResult[20];
    int foundEpsilon;
    subResult[0]='\0';
    Result[0]='\0';
    //If X is terminal, FIRST(X) = {X}.
    if(!(isupper(c)))
    {
        addToResultSet(Result,c);
              return ;
    }
    //If X is non terminal
    //Read each production
    for(i=0;i<numOfProductions;i++)
    {
//Find production with X as LHS
        if(productionSet[i][0]==c)
        {
//If X → ε is a production, then add ε to FIRST(X).
 if(productionSet[i][2]=='$') addToResultSet(Result,'$');
            //If X is a non-terminal, and X → Y1 Y2 … Yk
            //is a production, then add a to FIRST(X)
            //if for some i, a is in FIRST(Yi),
            //and ε is in all of FIRST(Y1), …, FIRST(Yi-1).
      else
            {
                j=2;
                while(productionSet[i][j]!='\0')
                {
                foundEpsilon=0;
                FIRST(subResult,productionSet[i][j]);
                for(k=0;subResult[k]!='\0';k++)
                    addToResultSet(Result,subResult[k]);
                 for(k=0;subResult[k]!='\0';k++)
                     if(subResult[k]=='$')
                     {
                         foundEpsilon=1;
                         break;
                     }
                 //No ε found, no need to check next element
                 if(!foundEpsilon)
                     break;
                 j++;
                }
            }
    }
}
    return ;
```

```
}
/* addToResultSet adds the computed
 *element to result set.
 *This code avoids multiple inclusion of elements
  */
void addToResultSet(char Result[],char val)
{
   int k;
   for(k=0 ;Result[k]!='\0';k++)
      if(Result[k]==val)
         return;
   Result[k]=val;
   Result[k+1]='\0';
}
```

Output:

```
How many number of productions ? :8
Enter productions Number 1 : E=TD
Enter productions Number 2 : D=+DT
Enter productions Number 3 : D=$
Enter productions Number 4 : T=FS
Enter productions Number 5 : S=*FS
Enter productions Number 6 : S=$
Enter productions Number 7 : F=(E)
Enter productions Number 8 : F=a

 Find the FIRST of  :E

 FIRST(E)= {  (  a }
press 'y' to continue : y

 Find the FIRST of  :T

 FIRST(T)= {  (  a }
press 'y' to continue : y

 Find the FIRST of  :D

 FIRST(D)= {  +  $ }
press 'y' to continue : y

 Find the FIRST of  :F

 FIRST(F)= {  (  a }
press 'y' to continue : y

 Find the FIRST of  :S

 FIRST(S)= {  *  $ }
press 'y' to continue :
```

// C program to compute follow of grammar

```
#include<stdio.h>
#include<string.h>
#define SIZE 10
#include<stdio.h>
#include<ctype.h>
#include<stdio.h>
#include<string.h>
int n,m=0,p,i=0,j=0;
char a[10][10],followResult[10];
void follow(char c);
void first(char c);
void addToResult(char);
int main()
{
 int i;
 int choice;
 char c,ch;
 printf("Enter the no.of productions: ");
scanf("%d", &n);
 printf(" Enter %d productions\nProduction with multiple terms should be give as separate
productions \n", n);
 for(i=0;i<n;i++)
  scanf("%s%c",a[i],&ch);
    // gets(a[i]);
 do
 {
  m=0;
  printf("Find FOLLOW of -->");
  scanf(" %c",&c);
  follow(c);
  printf("FOLLOW(%c) = { ",c);
  for(i=0;i<m;i++)
   printf("%c ",followResult[i]);
  printf(" }\n");
  printf("Do you want to continue(Press 1 to continue....)?");
 scanf("%d%c",&choice,&ch);
 }
 while(choice==1);
  return 0;

}
void follow(char c)
{
    if(a[0][0]==c)addToResult('$');
 for(i=0;i<n;i++)
 {
  for(j=2;j<strlen(a[i]);j++)
```

```
  {
   if(a[i][j]==c)
   {
    if(a[i][j+1]!='\0')first(a[i][j+1]);
    if(a[i][j+1]=='\0'&&c!=a[i][0])
     follow(a[i][0]);
   }
  }
 }
}
void first(char c)
{
     int k;
            if(!(isupper(c)))
               //f[m++]=c;
               addToResult(c);
            for(k=0;k<n;k++)
            {
            if(a[k][0]==c)
            {
            if(a[k][2]=='$') follow(a[i][0]);
            else if(islower(a[k][2]))
               //f[m++]=a[k][2];
               addToResult(a[k][2]);
            else first(a[k][2]);
            }
            }}
void  addToResult(char c){
    int i;
    for( i=0;i<=m;i++)
        if(followResult[i]==c)
            return;
   followResult[m++]=c;
}
```

Output:

```
Enter the no.of productions: 8
 Enter 8 productions
Production with multiple terms should be give as separate productions
E=TD
D=+TD
D=$
T=FS
S=*FS
S=$
F=(E)
F=a
Find FOLLOW of -->E
FOLLOW(E) = { $ )  }
Do you want to continue(Press 1 to continue....)?1
Find FOLLOW of -->S
FOLLOW(S) = { + $ )  }
Do you want to continue(Press 1 to continue....)?1
Find FOLLOW of -->F
FOLLOW(F) = { * + $ )  }
Do you want to continue(Press 1 to continue....)?1
Find FOLLOW of -->D
FOLLOW(D) = { $ )  }
Do you want to continue(Press 1 to continue....)?
```

//C program to include shift reduce parser

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<string.h>
char ip_sym[15],stack[15];
int ip_ptr=0,st_ptr=0,len,i;
char temp[2],temp2[2];
char act[15];
void check();
int main()
{
printf("\n\t\t SHIFT REDUCE PARSER\n");
printf("\n GRAMMER\n");
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");
scanf("%s",ip_sym);
printf("\n\t stack implementation table");
printf("\n stack \t\t input symbol\t\t action");
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++)
{
stack[st_ptr]=ip_sym[ip_ptr];
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();
st_ptr++;
}
st_ptr++;
check();
return 0;
}
void check()
{
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"))
printf("\n $%s\t\t%s$\t\t\tE->a",stack,ip_sym);
else
printf("\n $%s\t\t%s$\t\t\tE->b",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"))
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
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",stack,ip_sym);
exit(0);
}
if(flag==0)
{
printf("\n%s\t\t\t%s\t\t reject",stack,ip_sym);
exit(0);
}
return;
}
```

Output:

```
                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$                    --
$a               +b$                    shifta
$E               +b$                    E->a
$E+               b$                    shift+
$E+b               $                    shiftb
$E+E               $                    E->b
$E                 $                    E->E+E
$E                 $                    ACCEPT

...Program finished with exit code 0
Press ENTER to exit console.
```