**CD LAB FILE**

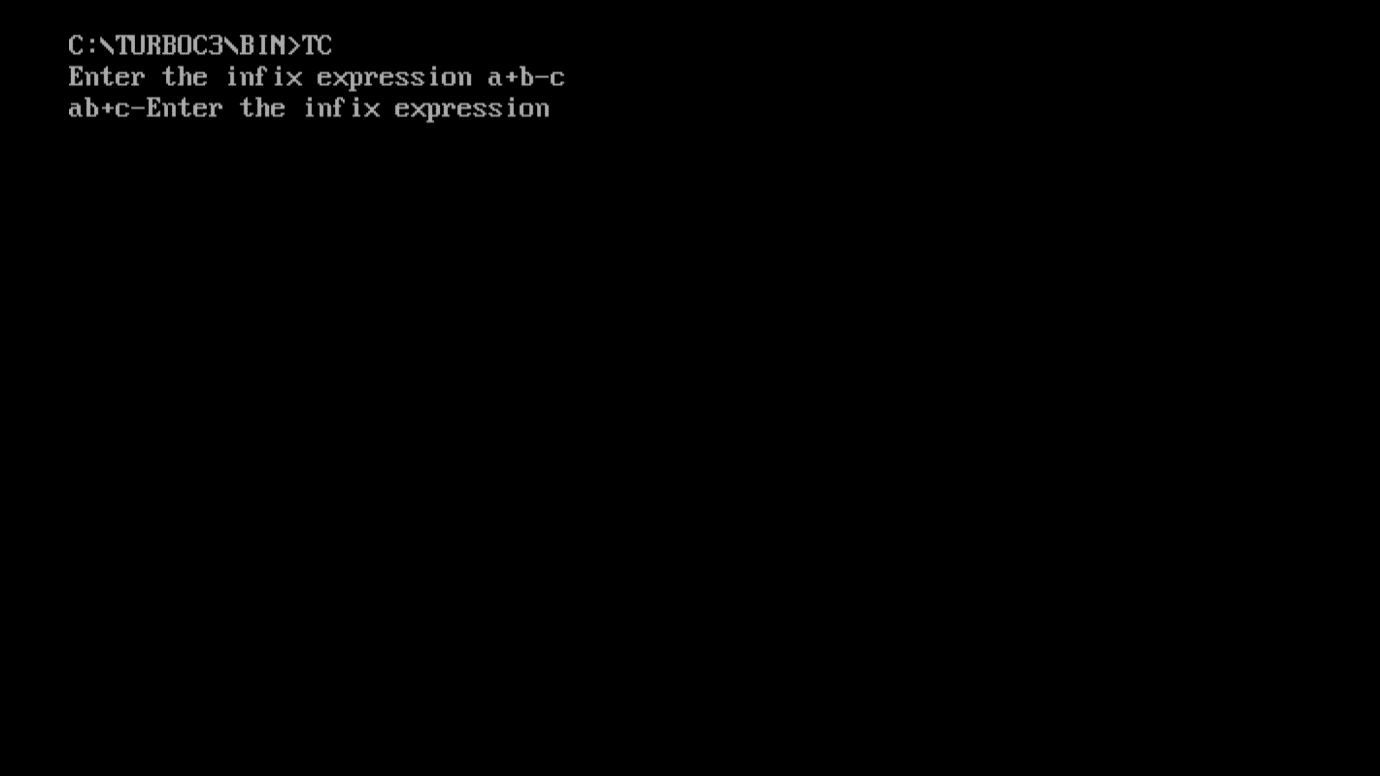
**Nishant Tiwari (1901610109005)**

**1. WAP to convert infix expression to postfix expression.**

**Program:**

|  |
| --- |
|  |
|  |
|  | #include <stdio.h>  #include <ctype.h> |
|  | char stack[20]; |
|  | int top=-1; |
|  |  |
|  | void push(char x) |
|  | { |
|  | stack[++top]=x; |
|  | } |
|  |  |
|  | char pop() |
|  | { |
|  | if(top==-1) |
|  | return -1; |
|  | Else |
|  | return stack[top--]; |
|  | } |
|  |  |
|  | int priority(char x) |
|  | { |
|  | if(x=='(') |
|  | return 0; |
|  | else if(x=='+'|| x=='-') |
|  | return 1; |
|  | else if(x=='\*'|| x=='/') |
|  | return 2; |
|  | } |
|  | int main() |
|  | { |
|  | char exp[20]; |
|  | char \*c,x; |
|  | printf("Enter the infix expression"); |
|  | scanf("%s",exp); |
|  | c=exp; |
|  |  |
|  | while(\*c!='\0') |
|  | { |
|  | if(isalnum(\*c)) |
|  | printf("%c",\*c); |
|  | else if(\*c == '(') |
|  | push(\*c); |
|  | else if (\*c == ')') |
|  | { |
|  | while((x=pop())!='c') |
|  | printf("%c",x); |
|  | } |
|  | Else |
|  | { |
|  | while(priority(stack[top])>=priority(\*c)) |
|  | printf("%c",pop()); |
|  | push(\*c); |
|  | } |
|  | c++; |
|  | } |
|  | while(top!=-1) |
|  | { |
|  | printf("%c",pop()); |
|  | } |
|  | return 0; |
|  | } |

**Output:**



**2. WAP to convert infix expression to prefix expression.**

**Program:**

# include <stdio.h>

# include <string.h>

# define MAX 20

void infixtoprefix(char infix[20],char prefix[20]);

void reverse(char array[30]);

char pop();

void push(char symbol);

int isOperator(char symbol);

int prcd(symbol);

int top=-1;

char stack[MAX];

main() {

char infix[20],prefix[20],temp;

printf("Enter infix operation: ");

gets(infix);

infixtoprefix(infix,prefix);

reverse(prefix);

puts((prefix));

}

//--------------------------------------------------------

void infixtoprefix(char infix[20],char prefix[20]) {

int i,j=0;

char symbol;

stack[++top]='#';

reverse(infix);

for (i=0;i<strlen(infix);i++) {

symbol=infix[i];

if (isOperator(symbol)==0) {

prefix[j]=symbol;

j++;

} else {

if (symbol==')') {

push(symbol);

} else if(symbol == '(') {

while (stack[top]!=')') {

prefix[j]=pop();

j++;

}

pop();

} else {

if (prcd(stack[top])<=prcd(symbol)) {

push(symbol);

} else {

while(prcd(stack[top])>=prcd(symbol)) {

prefix[j]=pop();

j++;

}

push(symbol);

}

//end for else

}

}

//end for else

}

//end for for

while (stack[top]!='#') {

prefix[j]=pop();

j++;

}

prefix[j]='\0';

}

////--------------------------------------------------------

void reverse(char array[30]) // for reverse of the given expression {

int i,j;

char temp[100];

for (i=strlen(array)-1,j=0;i+1!=0;--i,++j) {

temp[j]=array[i];

}

temp[j]='\0';

strcpy(array,temp);

return array;

}

//--------------------------------

char pop() {

char a;

a=stack[top];

top--;

return a;

}

//----------------------------------

void push(char symbol) {

top++;

stack[top]=symbol;

}

//------------------------------------------

int prcd(symbol) // returns the value that helps in the precedence {

switch(symbol) {

case '+':

case '-':

return 2;

break;

case '\*':

case '/':

return 4;

break;

case '$':

case '^':

return 6;

break;

case '#':

case '(':

case ')':

return 1;

break;

}

}

//-------------------------------------------------

int isOperator(char symbol) {

switch(symbol) {

case '+':

case '-':

case '\*':

case '/':

case '^':

case '$':

case '&':

case '(':

case ')':

return 1;

break;

default:

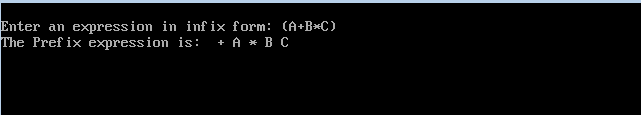
return 0;

// returns 0 if the symbol is other than given above

}

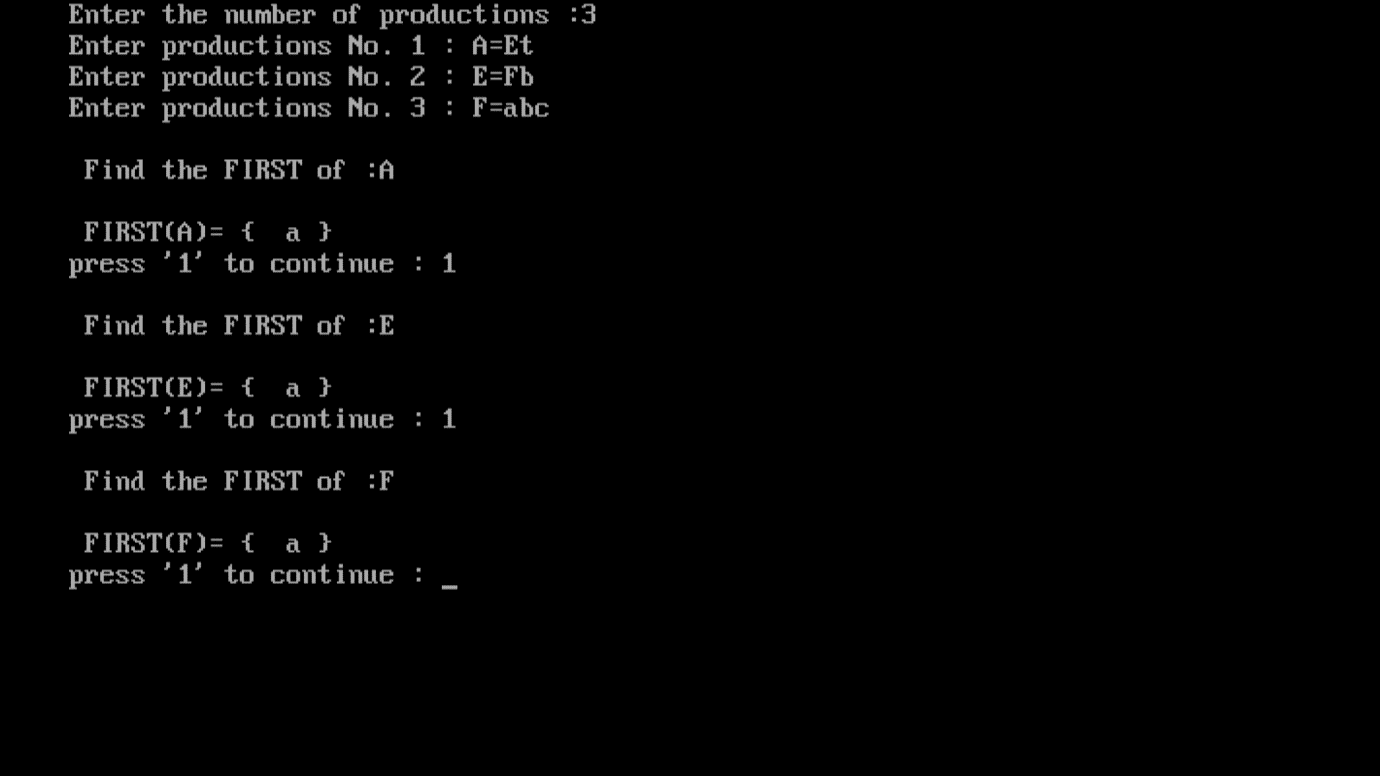
}

**Output:**



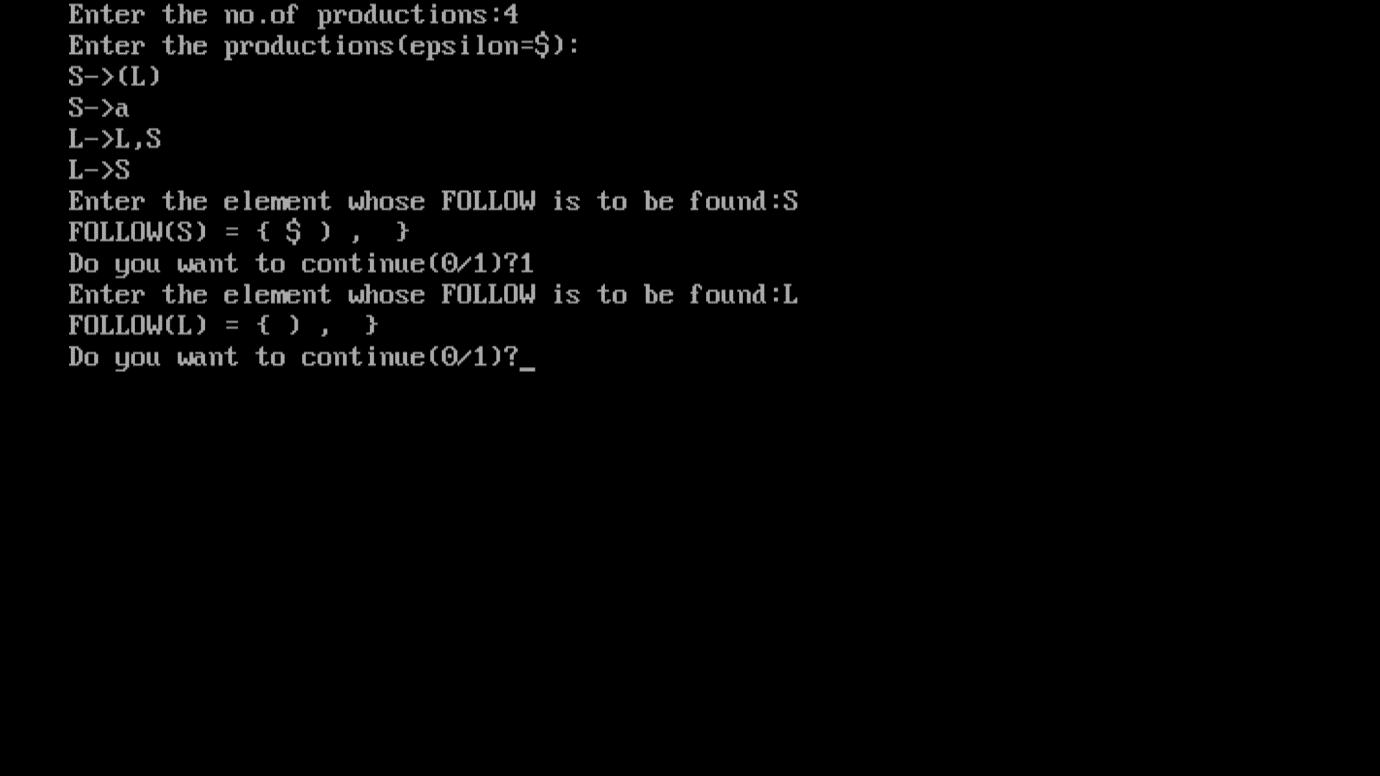
**3. WAP to calculate FIRST of a grammar.**

|  |
| --- |
|  |
|  | #include<ctype.h>  #include<stdio.h> |
|  | #include<conio.h> |
|  | void first(char[],char ); |
|  | void add(char[],char); |
|  | int n; |
|  | char prod[10][10]; |
|  | void main() |
|  | { |
|  | int i; |
|  | char choice; |
|  | char c; |
|  | char result[20]; |
|  | clrscr(); |
|  | printf("Enter the number of productions :"); |
|  | scanf(" %d",&n); |
|  | for(i=0;i<n;i++) |
|  | { |
|  | printf("Enter productions No. %d : ",i+1); |
|  | scanf(" %s",prod[i]); |
|  | } |
|  | do |
|  | { |
|  | printf("\n Find the FIRST of :"); |
|  | scanf(" %c",&c); |
|  | first(result,c); |
|  | printf("\n FIRST(%c)= { ",c); |
|  | for(i=0;result[i]!='\0';i++) |
|  | printf(" %c ",result[i]); |
|  | printf("}\n"); |
|  | printf("press '1' to continue : "); |
|  | scanf(" %c",&choice); |
|  | } |
|  | while(choice=='1'); |
|  | } |
|  |  |
|  | void first(char\* Result,char c) |
|  | { |
|  | int i,j,k; |
|  | char subResult[20]; |
|  | int foundEpsilon; |
|  | subResult[0]='\0'; |
|  | Result[0]='\0'; |
|  | if(!(isupper(c))) |
|  | { |
|  | add(Result,c); |
|  | } |
|  | for(i=0;i<n;i++) |
|  | { |
|  | if(prod[i][0]==c) |
|  | { |
|  | if(prod[i][2]=='$') add(Result,'$'); |
|  |  |
|  | else |
|  | { |
|  | j=2; |
|  | while(prod[i][j]!='\0') |
|  | { |
|  | foundEpsilon=0; |
|  | first(subResult,prod[i][j]); |
|  | for(k=0;subResult[k]!='\0';k++) |
|  | add(Result,subResult[k]); |
|  | for(k=0;subResult[k]!='\0';k++) |
|  | if(subResult[k]=='$') |
|  | { |
|  | foundEpsilon=1; |
|  | break; |
|  | } |
|  |  |
|  | if(!foundEpsilon) |
|  | break; |
|  | j++; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | return ; |
|  | } |
|  |  |
|  | void add(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'; |
|  | } |



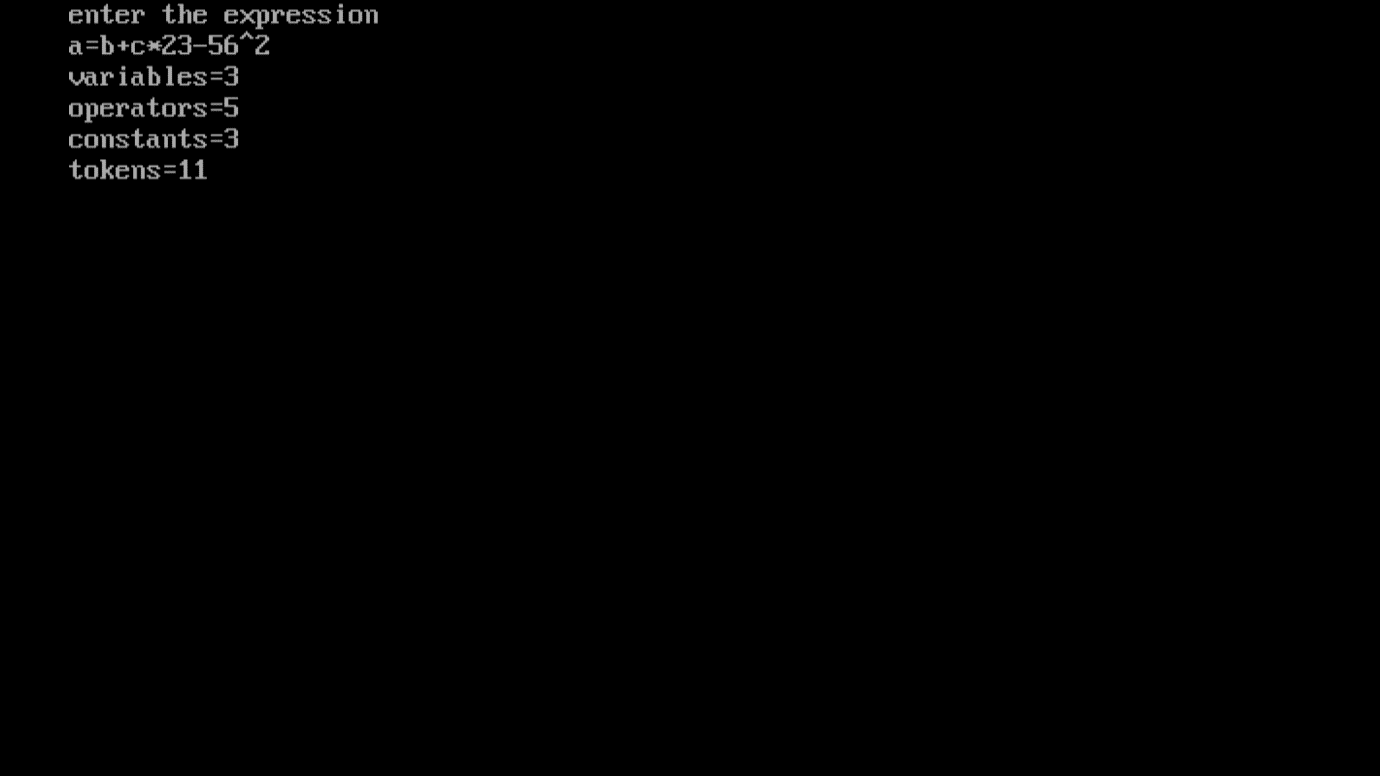
**4. WAP to calculate FOLLOW of a grammar.**

|  |
| --- |
|  |
|  | #include<conio.h>  #include<stdio.h> |
|  | #include<string.h> |
|  | int n,m=0,p,i=0,j=0; |
|  | char a[10][10],f[10]; |
|  | void follow(char c); |
|  | void first(char c); |
|  | void main() |
|  | { |
|  | int i,x; |
|  | char c,ch; |
|  | clrscr(); |
|  | printf("Enter the no.of productions:"); |
|  | scanf("%d",&n); |
|  | printf("Enter the productions(epsilon=$):\n"); |
|  | for(i=0;i<n;i++) |
|  | scanf("%s%c",a[i],&ch); |
|  |  |
|  | do |
|  | { |
|  | m=0; |
|  | printf("Enter the element whose FOLLOW is to be found:"); |
|  |  |
|  | scanf("%c",&c); |
|  | follow(c); |
|  | printf("FOLLOW(%c) = { ",c); |
|  | for(i=0;i<m;i++) |
|  | printf("%c ",f[i]); |
|  | printf(" }\n"); |
|  | printf("Do you want to continue(0/1)?"); |
|  | scanf("%d%c",&x,&ch); |
|  | } |
|  | while(x==1); |
|  | getch(); |
|  | } |
|  | void follow(char c) |
|  | { |
|  |  |
|  | if(a[0][0]==c)f[m++]='$'; |
|  | 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; |
|  | 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]; |
|  | else first(a[k][2]); |
|  | } |
|  | } |
|  |  |
|  | } |



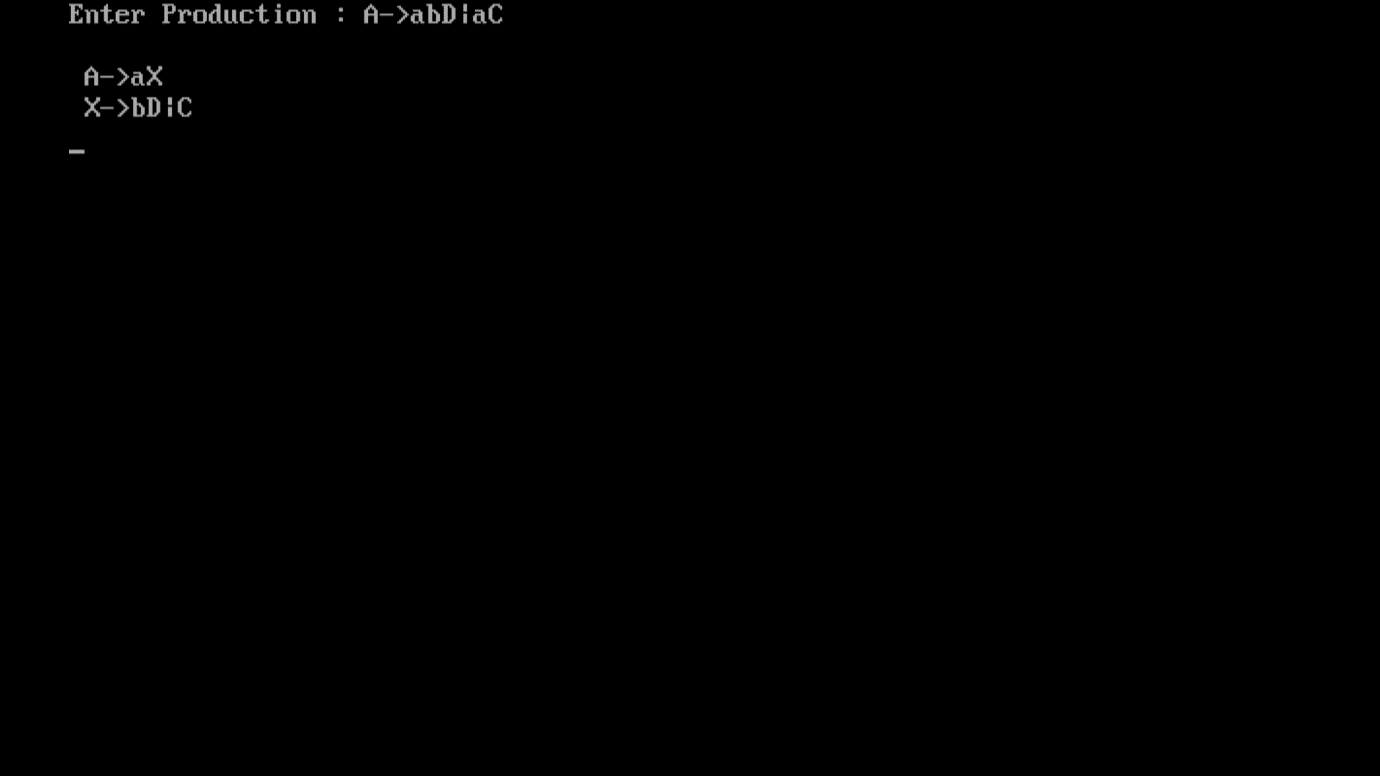
**5. WAP to find the no. of tokens and list them according to their category in an expression (given/entered).**

|  |
| --- |
|  |
|  | #include<conio.h>  #include<stdio.h> |
|  | #include<string.h> |
|  | void main() |
|  | { |
|  | int i,op=0,var=0,con=0; |
|  | char s[100]; |
|  | clrscr(); |
|  | printf("enter the expression\n"); |
|  | scanf("%s",&s); |
|  |  |
|  | for(i=0;i<=strlen(s);i++) |
|  | { |
|  | if((s[i]>=65 && s[i]<=90)||(s[i]<=122 && s[i]>=97)) |
|  | { |
|  | if(s[i+1]==' ' || s[i+1]=='\0' || s[i+1]=='=' || s[i+1]=='+' || s[i+1]=='\*' || s[i+1]=='-' || s[i+1]=='/' || s[i+1]=='^' || s[i+1]=='0' || s[i+1]=='1' || s[i+1]=='2'|| s[i+1]=='3' || s[i+1]=='4' || s[i+1]=='5'|| s[i+1]=='6' || s[i+1]=='7'|| s[i+1]=='8' || s[i+1]=='9' ) |
|  | { |
|  | var+=1; |
|  | } |
|  | continue; |
|  | } |
|  | else if(s[i]=='=' || s[i]=='+' || s[i]=='\*' || s[i]=='-' || s[i]=='/' || s[i]=='^' ) |
|  | { |
|  | op+=1; |
|  | } |
|  | else if(s[i]=='0' || s[i]=='1' || s[i]=='2'|| s[i]=='3' || s[i]=='4' || s[i]=='5'|| s[i]=='6' || s[i]=='7'|| s[i]=='8' || s[i]=='9') |
|  | { |
|  | if(s[i+1]=='\0' || s[i+1]==' ' || s[i+1]=='=' || s[i+1]=='+' || s[i+1]=='\*' || s[i+1]=='-' || s[i+1]=='/' || s[i+1]=='^') |
|  | { |
|  | con+=1; |
|  | } |
|  | continue; |
|  | } |
|  | } |
|  | printf("variables=%d\n",var); |
|  | printf("operators=%d\n",op); |
|  | printf("constants=%d\n",con); |
|  | printf("tokens=%d\n",var+op+con); |
|  | getch(); |
|  | } |



## 6. WAP to remove left factoring.

|  |
| --- |
|  |
|  |
|  | #include<string.h>  #include<stdio.h> |
|  | #include<conio.h> |
|  | void main() |
|  | { |
|  | char gram[20],part1[20],part2[20],modifiedGram[20],newGram[20],tempGram[20]; |
|  | int i,j=0,k=0,l=0,pos; |
|  | clrscr(); |
|  | printf("Enter Production : A->"); |
|  | gets(gram); |
|  | for(i=0;gram[i]!='|';i++,j++) |
|  | part1[j]=gram[i]; |
|  | part1[j]='\0'; |
|  | for(j=++i,i=0;gram[j]!='\0';j++,i++) |
|  | part2[i]=gram[j]; |
|  | part2[i]='\0'; |
|  | for(i=0;i<strlen(part1)||i<strlen(part2);i++) |
|  | { |
|  | if(part1[i]==part2[i]) |
|  | { |
|  | modifiedGram[k]=part1[i]; |
|  | k++; |
|  | pos=i+1; |
|  | } |
|  | } |
|  | for(i=pos,j=0;part1[i]!='\0';i++,j++){ |
|  | newGram[j]=part1[i]; |
|  | } |
|  | newGram[j++]='|'; |
|  | for(i=pos;part2[i]!='\0';i++,j++){ |
|  | newGram[j]=part2[i]; |
|  | } |
|  | modifiedGram[k]='X'; |
|  | modifiedGram[++k]='\0'; |
|  | newGram[j]='\0'; |
|  | printf("\n A->%s",modifiedGram); |
|  | printf("\n X->%s\n",newGram); |
|  | getch(); |
|  |  |
|  | } |
|  |  |



## 7. WAP to implement ****Recursive Decent Parser.****

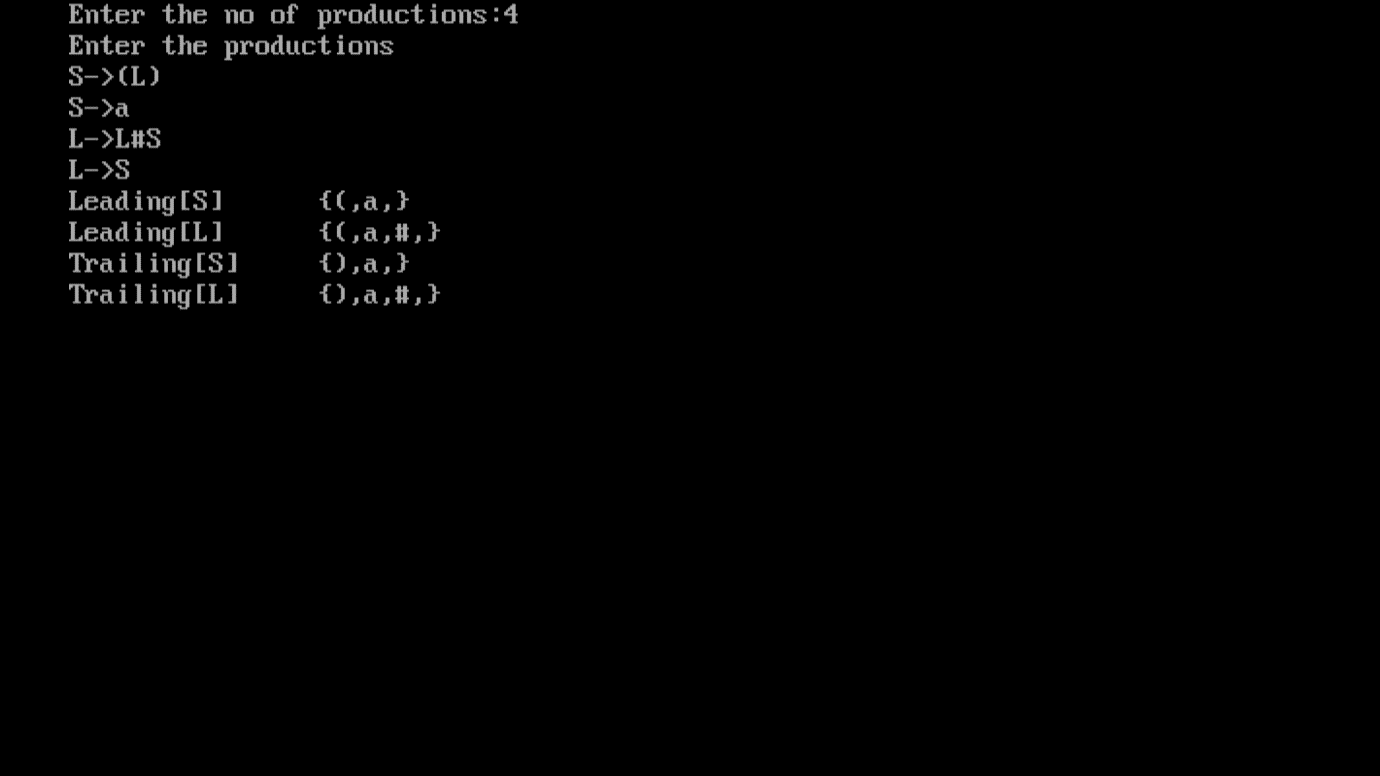
|  |
| --- |
|  |
|  | #include <conio.h>  #include <stdio.h> |
|  | char input[100]; |
|  | char prod[100][100]; |
|  | int pos=-1,l,st=-1; |
|  | char id,num; |
|  | void E(); |
|  | void T(); |
|  | void F(); |
|  | void advance(); |
|  | void Td(); |
|  | void Ed(); |
|  | void advance() |
|  | { |
|  | pos++; |
|  | if(pos<l) |
|  | { |
|  | if(input[pos]>='0'&& input[pos]<='9') |
|  | { |
|  | num=input[pos]; |
|  | id='\0'; |
|  | } |
|  | if((input[pos]>='a' || input[pos]>='A')&&(input[pos]<='z' || input[pos]<='Z')) |
|  | {id=input[pos]; |
|  | num='\0'; |
|  | } |
|  | } |
|  | } |
|  | void E() |
|  | { |
|  | strcpy(prod[++st],"E->TE'"); |
|  | T(); |
|  | Ed(); |
|  | } |
|  | void Ed() |
|  | { |
|  | int p=1; |
|  | if(input[pos]=='+') |
|  | { |
|  | p=0; |
|  | strcpy(prod[++st],"E'->+TE'"); |
|  | advance(); |
|  | T(); |
|  | Ed(); |
|  | } |
|  | if(input[pos]=='-') |
|  | { p=0; |
|  | strcpy(prod[++st],"E'->-TE'"); |
|  | advance(); |
|  | T(); |
|  | Ed(); |
|  | } |
|  |  |
|  | // Recursive Descent Parser |
|  | if(p==1) |
|  | { |
|  | strcpy(prod[++st],"E'->null"); |
|  | } |
|  | } |
|  |  |
|  | void T() |
|  | { |
|  | strcpy(prod[++st],"T->FT'"); |
|  | F(); |
|  | Td(); |
|  | } |
|  | void Td() |
|  | { |
|  | int p=1; |
|  | if(input[pos]=='\*') |
|  | { |
|  | p=0; |
|  | strcpy(prod[++st],"T'->\*FT'"); |
|  | advance(); |
|  | F(); |
|  | Td(); |
|  | } |
|  | if(input[pos]=='/') |
|  | { p=0; |
|  | strcpy(prod[++st],"T'->/FT'"); |
|  | advance(); |
|  | F(); |
|  | Td(); |
|  | } |
|  | if(p==1) |
|  | strcpy(prod[++st],"T'->null"); |
|  | } |
|  | void F() |
|  | { |
|  | if(input[pos]==id) { |
|  | strcpy(prod[++st],"F->id"); |
|  | advance(); } |
|  | if(input[pos]=='(') |
|  | { |
|  | strcpy(prod[++st],"F->(E)"); |
|  | advance(); |
|  | E(); |
|  | if(input[pos]==')') { |
|  | //strcpy(prod[++st],"F->(E)"); |
|  | advance(); } |
|  | } |
|  | if(input[pos]==num) |
|  | { |
|  | strcpy(prod[++st],"F->num"); |
|  | advance(); |
|  | } |
|  | } |
|  | int main() |
|  | { |
|  | int i; |
|  | printf("Enter Input String "); |
|  | scanf("%s",input); |
|  | l=strlen(input); |
|  | input[l]='$'; |
|  | advance(); |
|  | E(); |
|  | if(pos==l) |
|  | { |
|  | printf("String Accepted\n"); |
|  | for(i=0;i<=st;i++) |
|  | { |
|  | printf("%s\n",prod[i]); |
|  | } |
|  | } |
|  | else |
|  | { |
|  | printf("String rejected\n"); |
|  | } |
|  | getch(); |
|  | return 0; |
|  | } |

## 

## 8. WAP to ****find lead and trail.****

|  |
| --- |
|  |
|  | #include<string.h>  #include<iostream.h> |
|  | #include<conio.h> |
|  | int nt,t,top=0; |
|  | char s[50],NT[10],T[10],st[50],l[10][10],tr[50][50]; |
|  | int searchnt(char a) |
|  | { |
|  | int count=-1,i; |
|  | for(i=0;i<nt;i++) |
|  | { |
|  | if(NT[i]==a) |
|  | return i; |
|  | } |
|  | return count; |
|  | } |
|  | int searchter(char a) |
|  | { |
|  | int count=-1,i; |
|  | for(i=0;i<t;i++) |
|  | { |
|  | if(T[i]==a) |
|  | return i; |
|  | } |
|  | return count; |
|  | } |
|  | void push(char a) |
|  | { |
|  | s[top]=a; |
|  | top++; |
|  | } |
|  | char pop() |
|  | { |
|  | top--; |
|  | return s[top]; |
|  | } |
|  | void installl(int a,int b) |
|  |  |
|  | { |
|  | if(l[a][b]=='f') |
|  | { |
|  | l[a][b]='t'; |
|  | push(T[b]); |
|  | push(NT[a]); |
|  | } |
|  | } |
|  | void installt(int a,int b) |
|  | { |
|  | if(tr[a][b]=='f') |
|  | { |
|  | tr[a][b]='t'; |
|  | push(T[b]); |
|  | push(NT[a]); |
|  | } |
|  | } |
|  |  |
|  | void main() |
|  | { |
|  | int i,s,k,j,n; |
|  | char pr[30][30],b,c; |
|  | clrscr(); |
|  | cout<<"Enter the no of productions:"; |
|  | cin>>n; |
|  | cout<<"Enter the productions \n"; |
|  | for(i=0;i<n;i++) |
|  | cin>>pr[i]; |
|  | nt=0; |
|  | t=0; |
|  | for(i=0;i<n;i++) |
|  | { |
|  | if((searchnt(pr[i][0]))==-1) |
|  | NT[nt++]=pr[i][0]; |
|  | } |
|  | for(i=0;i<n;i++) |
|  | { |
|  | for(j=3;j<strlen(pr[i]);j++) |
|  | { |
|  | if(searchnt(pr[i][j])==-1) |
|  | { |
|  | if(searchter(pr[i][j])==-1) |
|  | T[t++]=pr[i][j]; |
|  | } |
|  | } |
|  | } |
|  | for(i=0;i<nt;i++) |
|  | { |
|  | for(j=0;j<t;j++) |
|  | l[i][j]='f'; |
|  | } |
|  | for(i=0;i<nt;i++) |
|  | { |
|  | for(j=0;j<t;j++) |
|  |  |
|  | tr[i][j]='f'; |
|  | } |
|  | for(i=0;i<nt;i++) |
|  | { |
|  | for(j=0;j<n;j++) |
|  | { |
|  | if(NT[(searchnt(pr[j][0]))]==NT[i]) |
|  | { |
|  | if(searchter(pr[j][3])!=-1) |
|  | installl(searchnt(pr[j][0]),searchter(pr[j][3])); |
|  | else |
|  | { |
|  | for(k=3;k<strlen(pr[j]);k++) |
|  | { |
|  | if(searchnt(pr[j][k])==-1) |
|  | { |
|  | installl(searchnt(pr[j][0]),searchter(pr[j][k])); |
|  | break; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  | while(top!=0) |
|  | { |
|  | b=pop(); |
|  | c=pop(); |
|  | for(s=0;s<n;s++) |
|  | { |
|  | if(pr[s][3]==b) |
|  | installl(searchnt(pr[s][0]),searchter(c)); |
|  | } |
|  | } |
|  | for(i=0;i<nt;i++) |
|  | { |
|  | cout<<"Leading["<<NT[i]<<"]"<<"\t{"; |
|  | for(j=0;j<t;j++) |
|  | { |
|  | if(l[i][j]=='t') |
|  | cout<<T[j]<<","; |
|  | } |
|  | cout<<"}\n"; |
|  | } |
|  |  |
|  |  |
|  | top=0; |
|  | for(i=0;i<nt;i++) |
|  | { |
|  | for(j=0;j<n;j++) |
|  | { |
|  | if(NT[searchnt(pr[j][0])]==NT[i]) |
|  | { |
|  | if(searchter(pr[j][strlen(pr[j])-1])!=-1) |
|  | installt(searchnt(pr[j][0]),searchter(pr[j][strlen(pr[j])-1])); |
|  | else |
|  | { |
|  | for(k=(strlen(pr[j])-1);k>=3;k--) |
|  | { |
|  | if(searchnt(pr[j][k])==-1) |
|  | { |
|  | installt(searchnt(pr[j][0]),searchter(pr[j][k])); |
|  | break; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  | while(top!=0) |
|  | { |
|  | b=pop(); |
|  | c=pop(); |
|  | for(s=0;s<n;s++) |
|  | { |
|  | if(pr[s][3]==b) |
|  | installt(searchnt(pr[s][0]),searchter(c)); |
|  | } |
|  | } |
|  | for(i=0;i<nt;i++) |
|  | { |
|  | cout<<"Trailing["<<NT[i]<<"]"<<"\t{"; |
|  | for(j=0;j<t;j++) |
|  | { |
|  | if(tr[i][j]=='t') |
|  | cout<<T[j]<<","; |
|  | } |
|  | cout<<"}\n"; |
|  | } |
|  | getch(); |
|  | } |

**OUTPUT:**



## ****9. WAP to find operator precedence table.****

|  |
| --- |
|  |
|  | #include<string.h>  #include<stdio.h> |
|  | void main(){ |
|  |  |
|  | /\*OPERATOR PRECEDENCE PARSER\*/ |
|  | 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"); |
|  | scanf("%s",&ter); |
|  | 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("String 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 :**

