

EXPERIMENT 1

Implementation of Lexical Analyzer

Aim: Write a program in C/C++ to implement a lexical analyzer.

Algorithm:

1. Start
2. Get the input expression from the user.
3. Store the keywords and operators.
4. Perform analysis of the tokens based on the ASCII values.
5. ASCII Range TOKEN TYPE
 97-122 Keyword else identifier
 48-57 Constant else operator
 Greater than 12 Symbol
6. Print the token types.
7. Stop

Program (lexi.c): /* Lexical Analyzer */

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
#include<string.h>
using namespace std;
int main()
{
    char key[11][10]={"for","while","do","then","else","break","switch","case","if","continue"};
    char oper[13]={'+','-','*','/','%','&','<','>','=',';',':','!'};
    char a[20],b[20],c[20];
    int i,j,l,m,k,flag;
    printf("\n Enter the expression: ");
    gets(a);
    i=0;
    while(a[i])
    {
        flag=0;
        j=0;
```

```

l=0;
b[0]='\0';
if((toascii(a[i]>=97))&&(toascii(a[i]<=122)))
{
if((toascii(a[i+1]>=97))&&(toascii(a[i+1]<=122)))
{
while((toascii(a[i]>=97))&&(toascii(a[i]<=122)))
{
b[j]=a[i];
j++; i++;
}
b[j]='\0';
}
else
{
b[j]=a[i];
i++;
b[j+1]='\0';
}
for(k=0;k<=9;k++)
{
if(strcmp(b,key[k])==0)
{
flag=1;
break;
}
}
if(flag==1)
printf("\n %s is the keyword",b);
else
printf("\n %s is the identifier",b);
}
else if((toascii(a[i]>=48))&&(toascii(a[i]<=57)))
{
if((toascii(a[i+1]>=48))&&(toascii(a[i+1]<=57)))
{
while((toascii(a[i]>=48))&&(toascii(a[i]<=57)))
{
c[l]=a[i];
l++; i++;
}
}
else
{
c[l]=a[i];
i++;l++;
}
}

```

```

}
c[l]='\0';
printf("\n %s is the constant",c);
} //second ifelse
else
{
for(m=0;m<13;m++)
{
if(a[i]==oper[m])
{
printf("\n %c is the operator",a[i]);
break;
}
}
if(m>=13)
printf("\n %c is the symbol",a[i]);
i++;
} //last else
} //while
return 0;
}

```

OUTPUT:

```

1  #include<stdio.h>
2  #include<conio.h>
3  #include<ctype.h>

Enter the expression: if(b>5)continue

if is the keyword
( is the symbol
b is the identifier
> is the operator
5 is the constant
) is the symbol
continue is the keyword
PS C:\Users\rvais\Desktop> cd "c:\Users\rvais\Desktop\" ; if ($?) { g++ CD12.cpp -o CD12 } ; if ($?) { .\CD12 }

Enter the expression: while(b<20)break

while is the keyword
( is the symbol
b is the identifier
< is the operator
20 is the constant
) is the symbol
break is the keyword
PS C:\Users\rvais\Desktop>

```

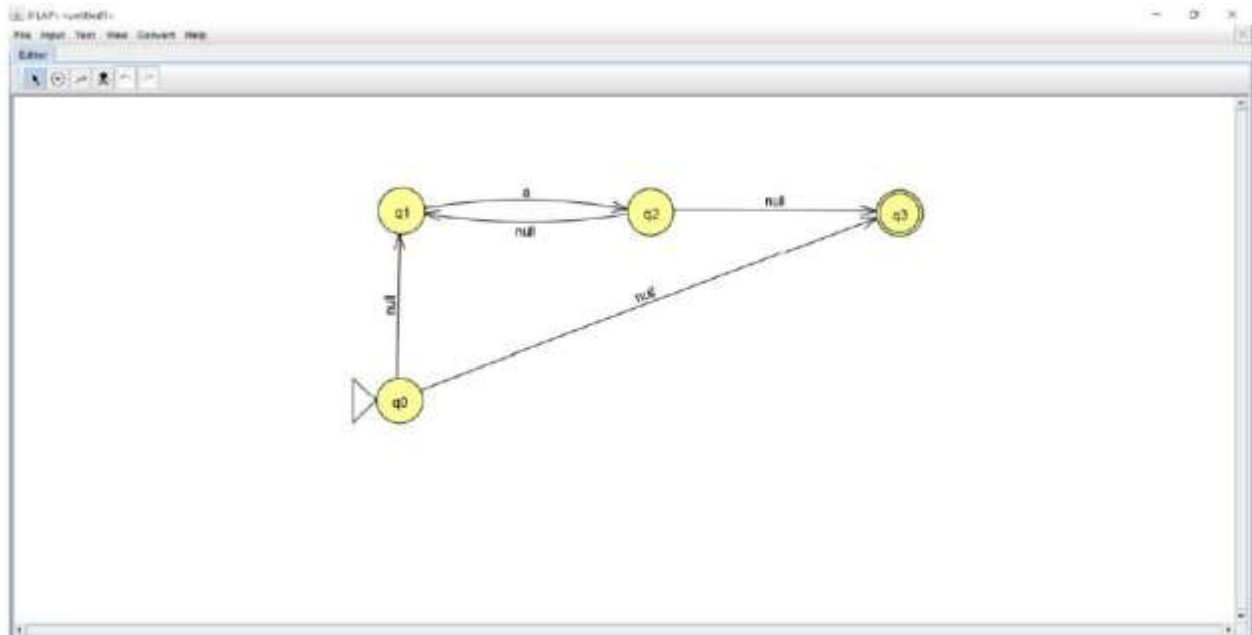
Result: The Program Executed successfully.

EXPERIMENT 2

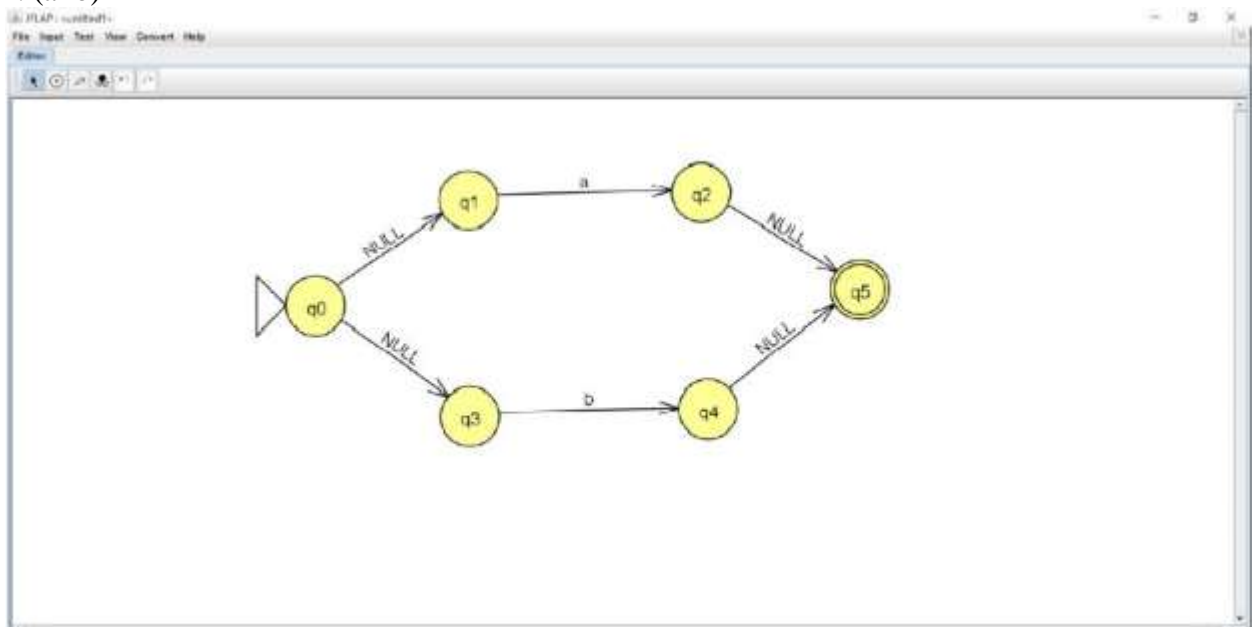
Conversion from Regular Expression to NFA

Aim: To convert the given Regular expression to NFA by using JFLAP.

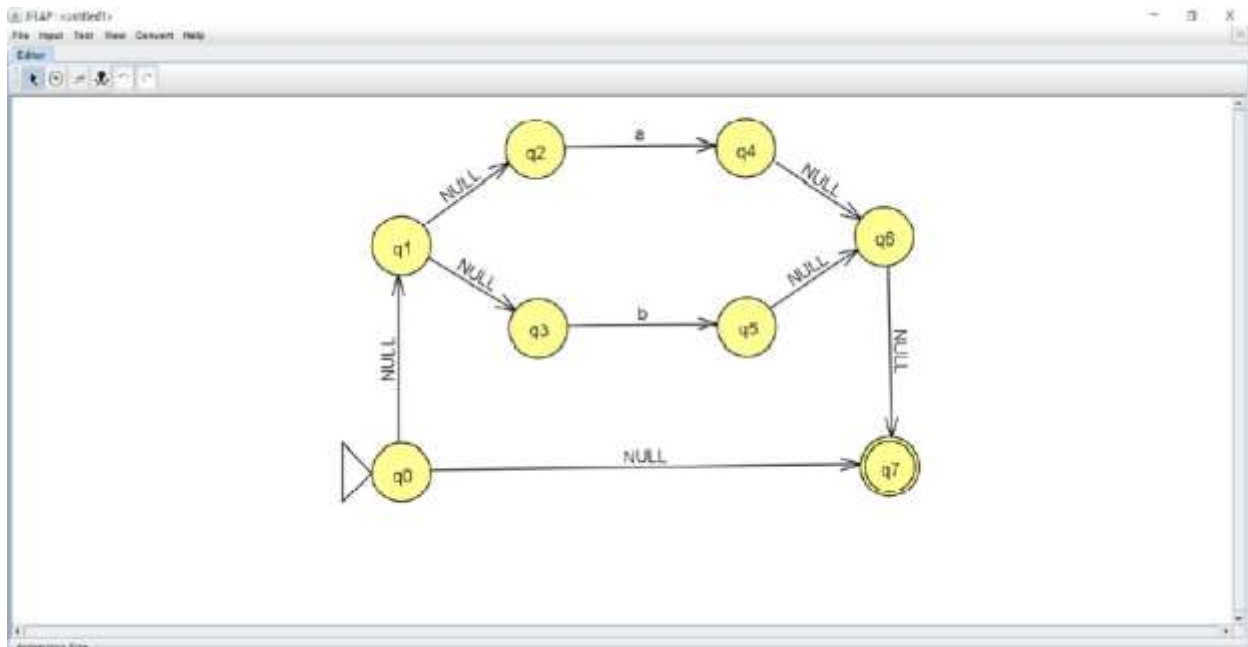
1. a^*



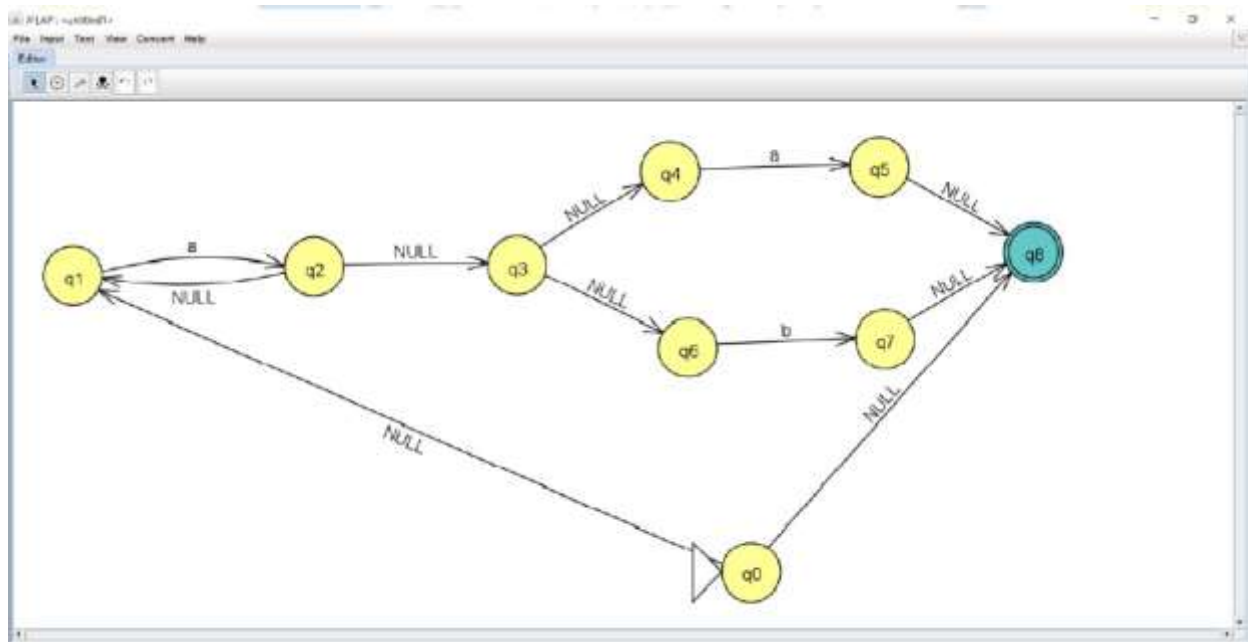
2. $(a+b)$



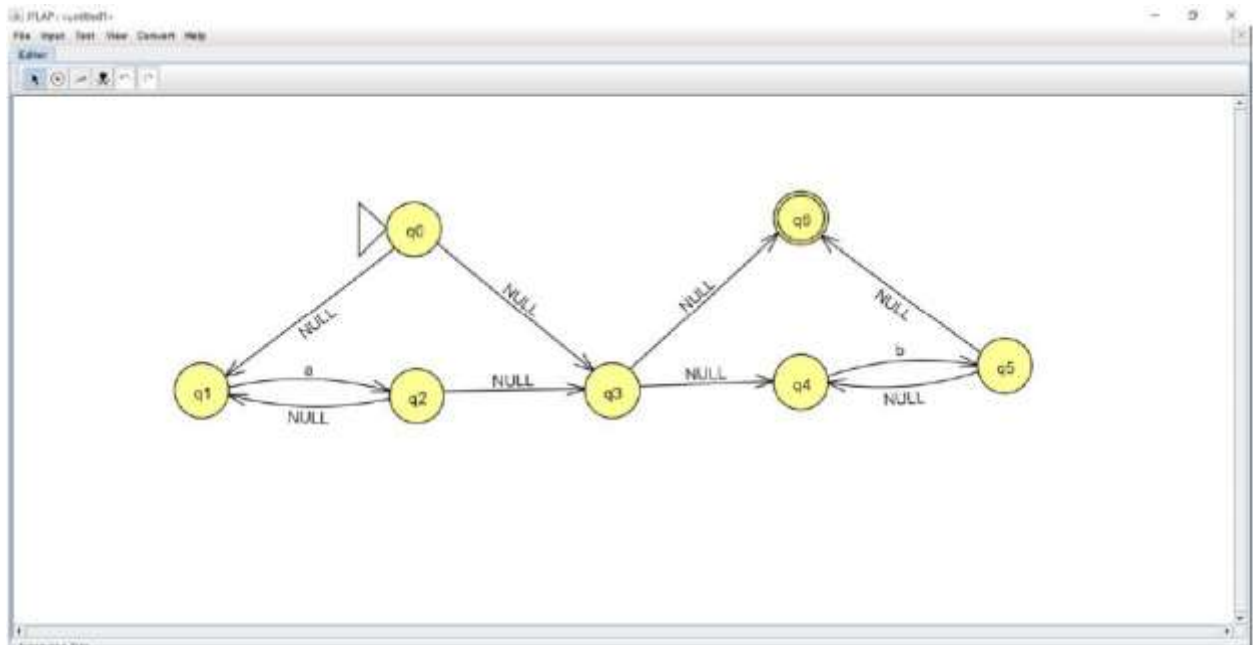
3. $(a+b)^*$



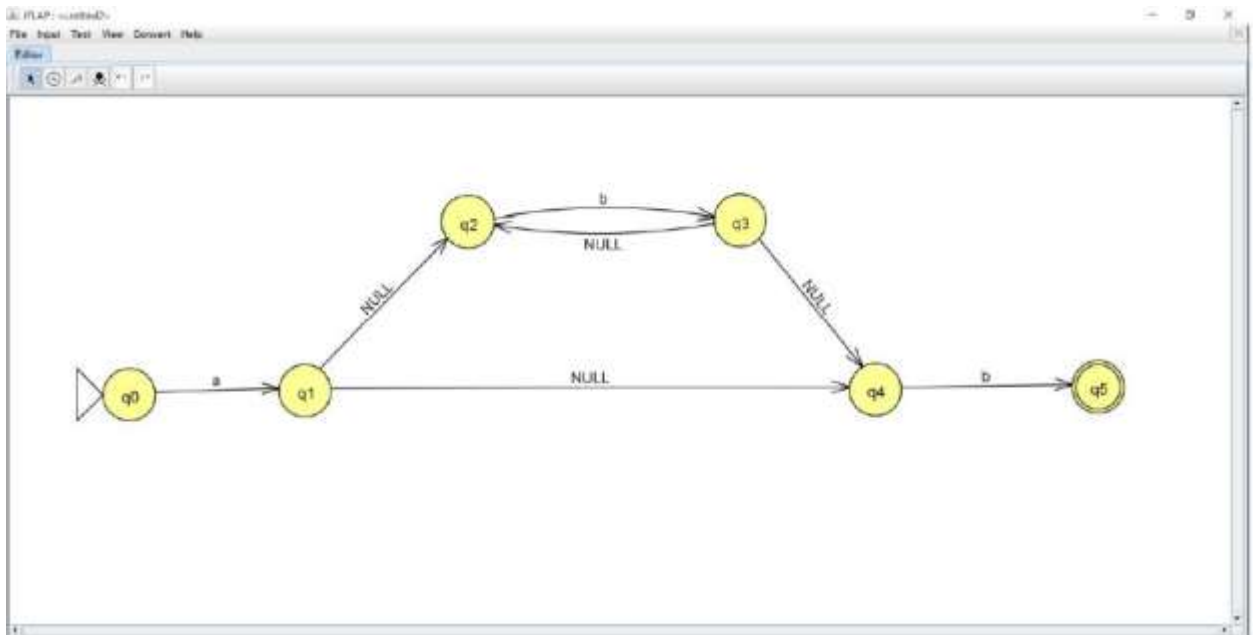
4. $a^*(a+b)$



5. a^*b^*



6. ab^*b



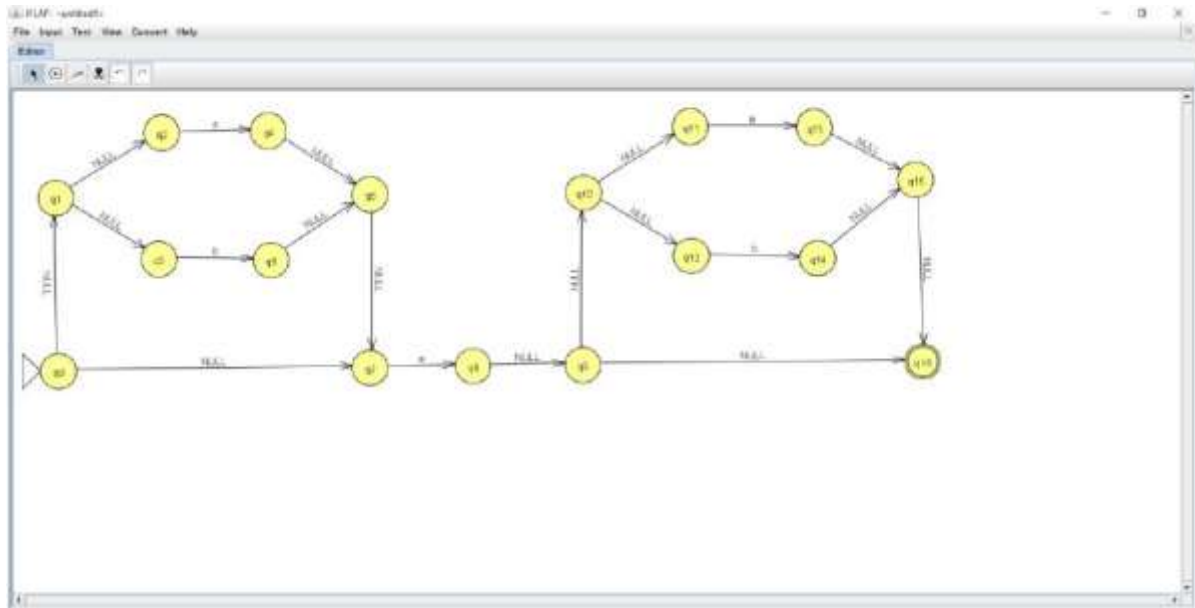
Result: We converted the given Regular expression to NFA.

Conversion from NFA to DFA

Aim: To convert the given Regular expression to DFA by using JFLAP.

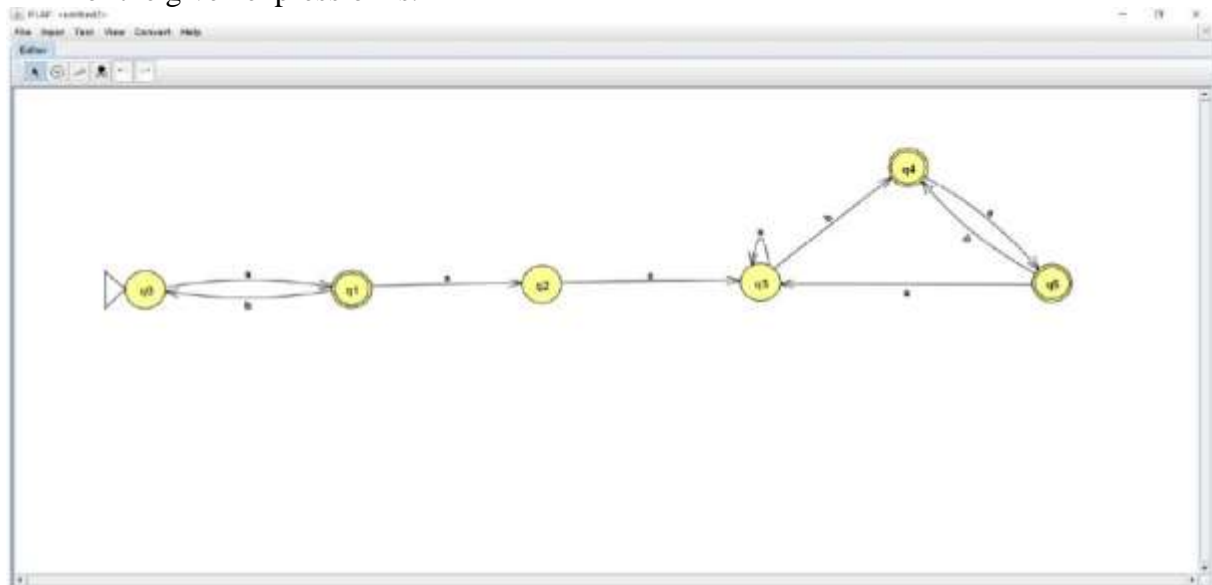
Ques: $(a+b) * a(a+b) *$

NFA for the given expression is:



OUTPUT: -

DFA for the given expression is:



Result: We converted the given Regular expression to DFA.

EXPERIMENT 4

Elimination of Ambiguity, Left Recursion and Left Factoring

Aim: To create a program which identifies the given grammar as left recursive or not and then perform elimination of ambiguity, elimination of left recursion and left factoring.

Program:

Elimination of left recursion:

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    int n, j, l, i, k;
    int length[10] = { };
    string d, a, b, flag;
    char c;
    cout<<"Enter Parent Non-Terminal: ";
    cin >> c;
    d.push_back(c);
    a += d + "'->";
    d += "->";
    b += d;
    cout<<"Enter productions: ";
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        cout<<"Enter Production ";
        cout<<i + 1<<" :";
        cin >> flag;
        length[i] = flag.size();
        d += flag;
        if (i != n - 1)
        {
            d += "|";
        }
    }
    cout<<"The Production Rule is: ";
    cout<<d<<endl;
    for (i = 0, k = 3; i < n; i++)
    {
        if (d[0] != d[k])
        {
            cout<<"Production: "<< i + 1;
            cout<<" does not have left recursion.";
```



```

cout<<endl;
if (d[k] == '#')
{
b.push_back(d[0]);
b += "\\";
}
else
{
for (j = k; j < k + length[i]; j++)
{
b.push_back(d[j]);
}
k = j + 1;
b.push_back(d[0]);
b += "\\|";
}
}
else
{
cout<<"Production: "<<i + 1 ;
cout<<" has left recursion";
cout<< endl;
if (d[k] != '#')
{
for (l = k + 1; l < k + length[i]; l++)
{
a.push_back(d[l]);
}
k = l + 1;
a.push_back(d[0]);
a += "\\|";
}
}
a += "#";
cout << b << endl;
cout << a << endl;
return 0;
}

```

Left factoring:

```

#include <iostream>
#include <string>
using namespace std;
int main()

```

```

{
int n,j,l,i,m;
int len[10] = { };
string a, b1, b2, flag;
char c;
cout << "Enter the Parent Non-Terminal : ";
cin >> c;
a.push_back(c);
b1 += a + "\\'->";
b2 += a + "\\\''->";
a += "->";
cout << "Enter total number of productions : ";
cin >> n;
for (i = 0; i < n; i++)
{
cout << "Enter the Production " << i + 1 << " : ";
cin >> flag;
len[i] = flag.size();
a += flag;
if (i != n - 1)
{
a += "|";
}
}
cout << "The Production Rule is : " << a << endl;
char x = a[3];
for (i = 0, m = 3; i < n; i++)
{
if (x != a[m])
{
while (a[m++] != '|');
}
else
{
if (a[m + 1] != '|')
{
b1 += "|" + a.substr(m + 1, len[i] - 1);
a.erase(m - 1, len[i] + 1);
}
else
{
b1 += "#";
a.insert(m + 1, 1, a[0]);
a.insert(m + 2, 1, "\");
m += 4;
}
}
}

```

```

    }
    }
    char y = b1[6];
    for (i = 0, m = 6; i < n - 1; i++)
    {
        if (y == b1[m])
        {
            if (b1[m + 1] != '|')
            {
                flag.clear();
                for (int s = m + 1; s < b1.length(); s++)
                {
                    flag.push_back(b1[s]);
                }
                b2 += "|" + flag;
                b1.erase(m - 1, flag.length() + 2);
            }
            else
            {
                b1.insert(m + 1, 1, b1[0]);
                b1.insert(m + 2, 2, "\\");
                b2 += "#";
                m += 5;
            }
        }
    }
    b2.erase(b2.size() - 1);
    cout << "After Left Factoring : " << endl;
    cout << a << endl;
    cout << b1 << endl;
    cout << b2 << endl;
    return 0;
}

```

OUTPUT:

Elimination of left recursion:

```
Enter the Parent Non-Terminal : S
Enter the number of productions : 2
Enter Production 1 : SOS1S
Enter Production 2 : 01
Production Rule : S->SOS1S|01
Production 1 has left recursion.
Production 2 does not have left recursion.
The Grammar after Eliminating the Left-Recursion is
S->01S'|
S'->OS1SS'|#
```

Left Factoring

```
Enter the Parent Non-Terminal : E
Enter total number of productions : 4
Enter the Production 1 : i
Enter the Production 2 : iE
Enter the Production 3 : (E)
Enter the Production 4 : iE+E
The Production Rule is : E->i|iE|(E)|iE+E
After Left Factoring :
E->iE'|(E)
E'->#|EE''
E''->#|+E
```

EXPERIMENT 5

FIRST AND FOLLOW computation

Aim: Write a program in C/C++ to find the FIRST and FOLLOW set for a given set of production rule of a grammar.

Algorithm:

Procedure First

1. Input the number of production N.
2. Input all the production rule PArray
3. Repeat steps a, b, c until process all input production rule i.e. PArray[N]
 - a. If $X_i \neq X_{i+1}$ then
 - i. Print Result array of X_i which contain FIRST(X_i)
 - b. If first element of X_i of PArray is Terminal or ϵ Then
 - i. Add Result = Result U first element
 - c. If first element of X_i of PArray is Non-Terminal Then
 - i. searchFirst(i, PArray, N)
4. End Loop
5. If N (last production) then
 - a. Print Result array of X_i which contain FIRST(X_i)
6. End

Procedure searchFirst(i, PArray, N)

1. Repeat steps Loop $j=i+1$ to N
 - a. If first element of X_j of PArray is Non-Terminal Then
 - i. searchFirst(j, of PArray, N)
 - b. If first element of X_j of PArray is Terminal or ϵ Then
 - i. Add Result = Result U first element
 - ii. Flag=0
2. End Loop

3. If Flag = 0 Then
 - a. Print Result array of Xj which contain FIRST(Xj)
4. End

Algorithm FOLLOW:

1. Declare the variables.
2. Enter the production rules for the grammar.
3. Calculate the FOLLOW set for each element call the user defined function follow().
4. If $x \rightarrow aBb$
 - a. If x is start symbol then $FOLLOW(x) = \{\$ \}$.
 - b. If b is NULL then $FOLLOW(B) = FOLLOW(x)$.
 - c. If b is not NULL then $FOLLOW(B) = FIRST(b)$.
5. END.

Program: // FIRST

```
#include<iostream>
#include<conio.h>
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
using namespace std;
void searchFirst(int n, int i, char pl[], char r[], char result[], int k)
{
    int j, flag;
    for(j=i+1; j<n; j++)
    {
        if(r[j]==pl[j])
        {
            if(isupper(r[j]))
            {
                searchFirst(n, j, pl, r, result, k);
            }
            if(islower(r[j]) || r[j]=='+' || r[j]=='*' || r[j]=='(' || r[j]==')')
            {
                result[k++]=r[j];
                result[k++]=';'; flag=0;
            }
        }
    }
}
```

```

    }
    }
    if(flag==0)
    {
    for(j=0;j<k-1;j++)cout<<result[j];
    }
    }
    int main()
    {
    char pr[10][10],pl[10],r[10],prev,result[10];
    int i,n,k,j;
    cout<<"\nHow many production rule : ";
    cin>>n;
    if(n==0) exit(0);
    for(i=0;i<n;i++)
    {
    cout<<"\nInput left part of production rules : ";
    cin>>pl[i];
    cout<<"\nInput right part of production rules : ";
    cin>>pr[i];
    r[i]=pr[i][0];
    }
    cout<<"\nProduction Rules are : \n";
    for(i=0;i<n;i++)
    {
    cout<<pl[i]<<"->"<<pr[i]<<"\n";//<<" "<<r[i]<<"\n";
    }
    cout<<"\n----O U T P U T---\n\n";
    prev=pl[0];k=0;
    for(i=0;i<n;i++)
    {
    if(prev!=pl[i])
    {
    cout<<"\nFIRST("<<prev<<")={ ";
    for(j=0;j<k-1;j++)cout<<result[j];
    cout<<"} ";
    k=0;prev=pl[i];
    //cout<<"\n3";
    }
    if(prev==pl[i])
    {
    if(islower(r[i]) || r[i]=='+' || r[i]=='*' || r[i]=='(' || r[i]==')')
    {
    result[k++]=r[i];
    result[k++]=',';
    }
    }
    }

```

```

if(isupper(r[i]))
{
cout<<"\nFIRST("<<prev<<")={ ";
searchFirst(n,i,pl,r,result,k);
cout<<"} ";
k=0;prev=pl[i+1];
}
}
}
if(i==n)
{
cout<<"\nFIRST("<<prev<<")={ ";
for(j=0;j<k-1;j++)cout<<result[j];
cout<<"} ";
k=0;prev=pl[i];
}
return 0;
}

```

OUTPUT: -

```

76     }
77     }
78     if(i==n)
79     {

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 2: Code

Production Rules are :
E->TX
X->+TX
X->e
T->FY
Y->*FY
Y->e
F->(E)
F->i

---O U T P U T---

FIRST(E)={(,i}
FIRST(X)={+,e}
FIRST(T)={(,i}
FIRST(Y)={*,e}
FIRST(F)={(,i}

PS C:\Users\rvais\Desktop>

Result: The Program Executed successfully.

Program: //FOLLOW


```

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

```

OUTPUT: -

```

36 void follow(char c)
37 {
38     if(a[0][0]==c)f[m++]='$';
39     for(i=0;i<n;i++)

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 2: Code

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

```

PS C:\Users\rvais\Desktop\VS Code Practice> cd "c:\Users\rvais\Desktop\" ; if ($?) { g++ CD12.cpp -o CD12 } ; if ($?) { .\CD12 }
Enter the no.of productions:3
Enter the productions(epsilon=$):
E=E+T
T=F
F=id
Enter the element whose FOLLOW is to be found:F
FOLLOW(F) = { $ + }
Do you want to continue(0/1)?1
Enter the element whose FOLLOW is to be found:E
FOLLOW(E) = { $ + }
Do you want to continue(0/1)?1
Enter the element whose FOLLOW is to be found:T
FOLLOW(T) = { $ + }
Do you want to continue(0/1)?0
PS C:\Users\rvais\Desktop>

```

Result: The Program Executed successfully.

EXPERIMENT 6

Predictive Parsing Table

Aim: Write a program in c for construction of predictive parser table.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
using namespace std;
char prol[7][10]={"S","A","A","B","B","C","C"};
char pror[7][10]={"A","Bb","Cd","aB","@","Cc","@"};
char prod[7][10]={"S->A","A->Bb","A->Cd","B->aB","B->@","C->Cc","C->@"}; char
first[7][10]={"abcd","ab","cd","a@","@","c@","@"}; char
follow[7][10]={"$","$","$","a$","b$","c$","d$"};
char table[5][6][10];
int numr(char c)
{
switch(c)
{
case 'S': return 0;
case 'A': return 1;
case 'B': return 2;
case 'C': return 3;
case 'a': return 0;
case 'b': return 1;
case 'c': return 2;
case 'd': return 3;
case '$': return 4;
}
return (2);
}
int main ()
{
int i,j,k;
for (i=0; i<5; i++)
for (j=0; j<6; j++)
strcpy(table[i][j]," ");
printf ("\nThe following is the predictive parsing table for the following grammar:\n");
for (i=0; i<7; i++)
printf ("%s\n",prod[i]);
printf ("\nPredictive parsing table is\n");
fflush (stdin);
for (i=0; i<7; i++)
{
```

```

k=strlen(first[i]);
for (j=0; j<10; j++)
if(first[i][j] !='@')
strcpy(table[numr(prol[i][0])+1][numr(first[i][j])+1],prod[i]);
}
for(i=0;i<7;i++)
{
if(strlen(pror[i])==1)
{
if(pror[i][0]=='@')
{
k=strlen(follow[i]);
for(j=0;j<k;j++)
strcpy(table[numr(prol[i][0])+1][numr(follow[i][j])+1],prod[i]);
}
}
}
strcpy(table[0][0]," ");
strcpy(table[0][1],"a");
strcpy(table[0][2],"b");
strcpy(table[0][3],"c");
strcpy(table[0][4],"d");
strcpy(table[0][5],"$");
strcpy(table[1][0],"S");
strcpy(table[2][0],"A");
strcpy(table[3][0],"B");
strcpy(table[4][0],"C");
printf("\n -----\\n");
for(i=0;i<5;i++)
for(j=0;j<6;j++){
printf("%-10s",table[i][j]);
if(j==5)
printf("\n -----\\n");
}
return 0;
}

```

OUTPUT:

```
75 return 0;
```

PROBLEMS OUTPUT **TERMINAL** DEBUG CONSOLE 2: Code

The following is the predictive parsing table for the following grammar:

S → A
A → Bb
A → Cd
B → aB
B → @
C → Cc
C → @

Predictive parsing table is

	a	b	c	d	\$
S	S → A	S → A	S → A	S → A	
A	A → Bb	A → Bb	A → Cd	A → Cd	
B	B → aB	B → @	B → @		B → @
C			C → @	C → @	C → @

Result: The Program Executed successfully.

EXPERIMENT 7

Shift Reduce Parsing

Aim: Write a program in C/C++ to implement the shift reduce parsing.

Algorithm:

1. Start the Process.
2. Symbols from the input are shifted onto stack until a handle appears on top of the stack.
3. The Symbols that are the handle on top of the stack are then replaces by the left-hand side of the production (reduced).
4. If this result in another handle on top of the stack, then another reduction is done, otherwise we go back to shifting.
5. This combination of shifting input symbols onto the stack and reducing productions when handles appear on the top of the stack continues until all of the input is consumed and the goal symbol is the only thing on the stack - the input is then accepted.
6. If we reach the end of the input and cannot reduce the stack to the goal symbol, the input is rejected.
7. Stop the process.

Program (srp.cpp):

```
#include<stdio.h>
#include<string.h>
int k=0,z=0,i=0,j=0,c=0;
char a[16],ac[20],stk[15],act[10];
void check();
int main()
{
puts("GRAMMAR is \n E->E+E \n E->E*E \n E->(E) \n E->id");
puts("Enter input string ");
gets(a);
c=strlen(a);
strcpy(act,"SHIFT->");
puts("STACK \t INPUT \tCOMMENT");
//puts("$ \t");
//puts(a);
printf("$ \t%s$\n",a);
for(k=0,i=0; j<c; k++,i++,j++)
{
if(a[j]=='i' && a[j+1]=='d')
{
```

```

stk[i]=a[j];
stk[i+1]=a[j+1];
stk[i+2]='\0';
a[j]=' ';
a[j+1]=' ';
//printf("$ \t%s$\n",a);
printf("\n$%s\t%s$\t%sid",stk,a,act);
check();
}
else
{
stk[i]=a[j];
stk[i+1]='\0';
a[j]=' ';
printf("\n$%s\t%s$\t%ssymbols",stk,a,act);
check();
}
}
}
void check()
{
strcpy(ac,"REDUCE TO E");
for(z=0; z<c; z++)
if(stk[z]=='i' && stk[z+1]=='d')
{
stk[z]='E';
stk[z+1]='\0';
printf("\n$%s\t%s$\t%s",stk,a,ac);
j++;
}
for(z=0; z<c; z++)
if(stk[z]=='E' && stk[z+1]=='+' && stk[z+2]=='E')
{
stk[z]='E';
stk[z+1]='\0';
stk[z+2]='\0';
printf("\n$%s\t%s$\t%s",stk,a,ac);
i=i-2;
}
for(z=0; z<c; z++)
if(stk[z]=='E' && stk[z+1]=='*' && stk[z+2]=='E')
{
stk[z]='E';
stk[z+1]='\0';
stk[z+1]='\0';
printf("\n$%s\t%s$\t%s",stk,a,ac);

```

```

i=i-2;
}
for(z=0; z<c; z++)
if(stk[z]=='(' && stk[z+1]=='E' && stk[z+2]==''))
{
stk[z]='E';
stk[z+1]='\0';
stk[z+1]='\0';
printf("\n%s\t%s\t%s",stk,a,ac);
i=i-2;
}
}

```

OUTPUT:

```

69      i=i-2;
70    }
71    for(z=0; z<c; z++)

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 2: Code

GRAMMAR is
E->E+E
E->E*E
E->(E)
E->id

Enter input string
id+id*id

STACK	INPUT	COMMENT
\$	id+id*id\$	
\$id	+id*id\$	SHIFT->id
\$E	+id*id\$	REDUCE TO E
\$E+	id*id\$	SHIFT->symbols
\$E+id	*id\$	SHIFT->id
\$E+E	*id\$	REDUCE TO E
\$E	*id\$	REDUCE TO E
\$E+	id\$	SHIFT->symbols
\$E*id	\$	SHIFT->id
\$E*E	\$	REDUCE TO E
\$E	\$	REDUCE TO E

PS C:\Users\rvais\Desktop>

Code for another Grammar: -

```

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int z = 0, i = 0, j = 0, c = 0;
char a[16], ac[20], stk[15], act[10];
void check()
{
strcpy(ac,"REDUCE TO E -> ");
for(z = 0; z < c; z++)
{

```



```

if(stk[z] == '4')
{
printf("%s4", ac);
stk[z] = 'E';
stk[z + 1] = '\0';
printf("\n%s\t%s\t", stk, a);
}
}
for(z = 0; z < c - 2; z++)
{
if(stk[z] == '2' && stk[z + 1] == 'E' &&
stk[z + 2] == '2')
{
printf("%s2E2", ac);
stk[z] = 'E';
stk[z + 1] = '\0';
stk[z + 2] = '\0';
printf("\n%s\t%s\t", stk, a);
i = i - 2;
}
}
for(z=0; z<c-2; z++)
{
if(stk[z] == '3' && stk[z + 1] == 'E' &&
stk[z + 2] == '3')
{
printf("%s3E3", ac);
stk[z]='E';
stk[z + 1]='\0';
stk[z + 1]='\0';
printf("\n%s\t%s\t", stk, a);
i = i - 2;
}
}
return; //return to main
}
int main()
{
printf("GRAMMAR is -\nE->2E2 \nE->3E3 \nE->4\n");
strcpy(a,"32423");
c=strlen(a);
strcpy(act,"SHIFT");
printf("\nstack \t input \t action");
printf("\n$\t%s$\t", a);
for(i = 0; j < c; i++, j++)
{

```

```

printf("%s", act);
stk[i] = a[j];
stk[i + 1] = '\0';
a[j]=' ';
printf("\n%s\t%s\t", stk, a);
check();
}
check();
if(stk[0] == 'E' && stk[1] == '\0')
printf("Accept\n");
else //else reject
printf("Reject\n");
}

```

OUTPUT:

```

GRAMMAR is -
E->2E2
E->3E3
E->4

stack   input   action
$       32423$  SHIFT
$3      2423$   SHIFT
$32     423$    SHIFT
$324    23$     REDUCE TO E -> 4
$32E    23$     SHIFT
$32E2   3$      REDUCE TO E -> 2E2
$3E     3$      SHIFT
$3E3    $       REDUCE TO E -> 3E3
$E      $       Accept

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

```

Result: The Program Executed successfully.

EXPERIMENT 8

Computation of leading and trailing

Aim: Write a program for finding the leading and trailing.

Program:

```
#include<iostream>
#include<string.h>
using namespace std;
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]);
}
}
int main()
{
int i,s,k,j,n;
char pr[30][30],b,c;
cout<<"Enter the no of productions:\n";
cin>>n;
cout<<"Enter the productions one by one\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";
}}

```

OUTPUT:

```
162 while(top!=0)
163 {
164 b=pop();
165 c=pop();
166 for(s=0;s<n;s++)
167 {
```

PROBLEMS OUTPUT **TERMINAL** DEBUG CONSOLE 2: Code + -

```
Enter the no of productions:
6
Enter the productions one by one
E->E+T
E->T
T->T*F
T->F
F->(E)
F->i
Leading[E]    {+,*,(,i,}
Leading[T]    {*,(,i,}
Leading[F]    {(,i,}
Trailing[E]  {+,*,),i,}
Trailing[T]  {*,),i,}
Trailing[F]  {),i,}
PS C:\Users\rvais\Desktop>
```

Result: The Program Executed successfully.

EXPERIMENT 5

Computation of LR (0) items

Aim: Write a program in C/C++ to implement LR(0) items.

Algorithm:

1. Start.
2. Create structure for production with LHS and RHS.
3. Open file and read input from file.
4. Build state 0 from extra grammar Law $S' \rightarrow S \$$ that is all start symbol of grammar and one Dot (.) before S symbol.
5. If Dot symbol is before a non-terminal, add grammar laws that this non-terminal is in Left Hand Side of that Law and set Dot in before of first part of Right Hand Side.
6. If state exists (a state with this Laws and same Dot position), use that instead.
7. Now find set of terminals and non-terminals in which Dot exist in before.
8. If step 7 Set is non-empty go to 9, else go to 10.
9. For each terminal/non-terminal in set step 7 create new state by using all grammar law that Dot position is before of that terminal/non-terminal in reference state by increasing Dot point to next part in Right Hand Side of that laws.
10. Go to step 5.
11. End of state building.
12. Display the output.
13. End.

Program:

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
char prod[20][20],listofvar[26]="ABCDEFGHIJKLMNOPQRSTUVWXYZR";
int novar=1,i=0,j=0,k=0,n=0,m=0,arr[30];
int noitem=0;
struct Grammar
{
    char lhs;
    char rhs[8];
}g[20],item[20],clos[20][10];
int isvariable(char variable)
{
    for(int i=0;i<novar;i++)
        if(g[i].lhs==variable)
            return i+1;
    return 0;
}
void findclosure(int z, char a)
```



```

{
int n=0,i=0,j=0,k=0,l=0;
for(i=0;i<arr[z];i++)
{
for(j=0;j<strlen(clos[z][i].rhs);j++)
{
if(clos[z][i].rhs[j]=='.' && clos[z][i].rhs[j+1]==a)
{
clos[noitem][n].lhs=clos[z][i].lhs;
strcpy(clos[noitem][n].rhs,clos[z][i].rhs);
char temp=clos[noitem][n].rhs[j];
clos[noitem][n].rhs[j]=clos[noitem][n].rhs[j+1];
clos[noitem][n].rhs[j+1]=temp;
n=n+1;
}
}
}
for(i=0;i<n;i++)
{
for(j=0;j<strlen(clos[noitem][i].rhs);j++)
{
if(clos[noitem][i].rhs[j]=='.' && isvariable(clos[noitem][i].rhs[j+1])>0)
{
for(k=0;k<novar;k++)
{
if(clos[noitem][i].rhs[j+1]==clos[0][k].lhs)
{
for(l=0;l<n;l++)
if(clos[noitem][l].lhs==clos[0][k].lhs &&
strcmp(clos[noitem][l].rhs,clos[0][k].rhs)==0)
break;
if(l==n)
{
clos[noitem][n].lhs=clos[0][k].lhs;
strcpy(clos[noitem][n].rhs,clos[0][k].rhs);
n=n+1;
}
}
}
}
}
}
arr[noitem]=n;
int flag=0;
for(i=0;i<noitem;i++)
{

```

```

if(arr[i]==n)
{
for(j=0;j<arr[i];j++)
{
int c=0;
for(k=0;k<arr[i];k++)
if(clos[noitem][k].lhs==clos[i][k].lhs &&
strcmp(clos[noitem][k].rhs,clos[i][k].rhs)==0)
c=c+1;
if(c==arr[i])
{
flag=1;
goto exit;
}
}
}
exit::;
if(flag==0)
arr[noitem++]=n;
}
void main()
{
clrscr();
cout<<"ENTER THE PRODUCTIONS OF THE GRAMMAR(0 TO END) :\n";
do
{
cin>>prod[i++];
}while(strcmp(prod[i-1],"0")!=0);
for(n=0;n<i-1;n++)
{
m=0;
j=novar;
g[novar++].lhs=prod[n][0];
for(k=3;k<strlen(prod[n]);k++)
{
if(prod[n][k] != '|')
g[j].rhs[m++]=prod[n][k];
if(prod[n][k]=='|')
{
g[j].rhs[m]='\0';
m=0;
j=novar;
g[novar++].lhs=prod[n][0];
}
}
}

```

```

}
for(i=0;i<26;i++)
if(!isvariable(listofvar[i]))
break;
g[0].lhs=listofvar[i];
char temp[2]={ g[1].lhs,'\0' };
strcat(g[0].rhs,temp);
cout<<"\n\n augmented grammar \n";
for(i=0;i<noavar;i++)
cout<<endl<<g[i].lhs<<"->"<<g[i].rhs<<" ";
getch();
for(i=0;i<noavar;i++)
{
clos[noitem][i].lhs=g[i].lhs;
strcpy(clos[noitem][i].rhs,g[i].rhs);
if(strcmp(clos[noitem][i].rhs,"ε")==0)
strcpy(clos[noitem][i].rhs,".");
else
{
for(int j=strlen(clos[noitem][i].rhs)+1;j>=0;j--)
clos[noitem][i].rhs[j]=clos[noitem][i].rhs[j-1];
clos[noitem][i].rhs[0]='.';
}
}
arr[noitem++]=noavar;
for(int z=0;z<noitem;z++)
{
char list[10];
int l=0;
for(j=0;j<arr[z];j++)
{
for(k=0;k<strlen(clos[z][j].rhs)-1;k++)
{
if(clos[z][j].rhs[k]=='.')
{
for(m=0;m<l;m++)
if(list[m]==clos[z][j].rhs[k+1])
break;
if(m==l)
list[l++]=clos[z][j].rhs[k+1];
}
}
}
for(int x=0;x<l;x++)
findclosure(z,list[x]);
}

```

```

cout<<"\n THE SET OF ITEMS ARE \n\n";
for(z=0;z<noitem;z++)
{
cout<<"\n I"<<z<<"\n\n";
for(j=0;j<arr[z];j++)
cout<<clos[z][j].lhs<<"->"<<clos[z][j].rhs<<"\n";
getch();
}
getch();
}

```

OUTPUT: -

Result: The Program Executed successfully.

Program: //FOLLOW

```

#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>
using namespace std;
int n,m=0,p,i=0,j=0;
char a[10][10],f[10];
void follow(char c);
void first(char c);
int main()
{
int i,z;
char c,ch;
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",&z,&ch);
}
while(z==1);
}
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]);
}
}
}
}

```

OUTPUT: -

ENTER THE PRODUCTIONS OF THE GRAMMAR(0 TO END) :

E->E+T

E->T

T->T*F

T->F

F->(E)

F->i

0

augumented grammar

A->E

E->E+T

E->T

T->T*F

T->F

F->(E)

F->i

THE SET OF ITEMS ARE

I0

A->.E

E->.E+T

E->.T

T->.T*F

T->.F

F->.(E)

F->.i

I1

A->E.

E->E.+T

I2

E->T.

T->T.*F

I3

T->F.

I4

F->(E)

E->.E+T

E->.T

T->.T*F

T->.F

F->.(E)

F->.i

I5

F->i.

I6

```
E->E+.T  
T-> .T*F  
T-> .F  
F-> . (E)  
F-> .i
```

I7

```
T->T*.F  
F-> . (E)  
F-> .i
```

I8

```
F-> (E.)  
E->E.+T
```

I9

```
E->E+T.  
T->T.*F
```

I10

```
T->T*F.
```

I11

```
F-> (E) .
```

Result: The Program Executed successfully.

EXPERIMENT 10

Intermediate code generation – Postfix, Prefix

Aim: Write a program in C/C++ or Java to generate Intermediate Code (Prefix and Postfix Expression) from given syntax tree.

Program: //Prefix

```
#define SIZE 50 /* Size of Stack */
#include<string.h>
#include<stdio.h>
#include <ctype.h>
using namespace std;
char s[SIZE];
int top=-1; /* Global declarations */
push(char elem)
{ /* Function for PUSH operation */
s[++top]=elem;
}
char pop()
{ /* Function for POP operation */
return(s[top--]);
}
int pr(char elem)
{ /* Function for precedence */
switch(elem)
{
case '#': return 0;
case ')': return 1;
case '+':
case '-': return 2;
case '*':
case '/': return 3;
}
}
int main()
{ /* Main Program */
char infx[50],prfx[50],ch,elem;
int i=0,k=0;
printf("\n\nRead the Infix Expression : ");
scanf("%s",infx);
push('#');
strrev(infx);
while( (ch=infx[i++]) != '\0')
{
if( ch == ')') push(ch);
```



```

else
if(isalnum(ch)) prfx[k++]=ch;
else
if( ch == '(')
{
while( s[top] != ')')
prfx[k++]=pop();
elem=pop(); /* Remove ) */
}
else
{ /* Operator */
while( pr(s[top]) >= pr(ch) )
prfx[k++]=pop();
push(ch);
}
}
while( s[top] != '#' ) /* Pop from stack till empty */
prfx[k++]=pop();
prfx[k]='\0'; /* Make prfx as valid string */
strrev(prfx);
strrev(infx);
printf("\n\nGiven Infix Expn: %s Prefix Expn: %s\n",infx,prfx);
return 0;
}

```

OUTPUT:

```

1  #define SIZE 50          /* Size of Stack */
2  #include<string.h>
3  #include<stdio.h>
4  #include <ctype.h>
5  using namespace std;
6  char s[SIZE];
7  int top=-1;             /* Global declarations */
8
9  int push(char elem)
10 {
11     s[++top]=elem;      /* Function for PUSH operation */
12 }

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

2: Code

```

PS C:\Users\rvais\Desktop\VS Code Practice> cd "C:\Users\rvais\Desktop\" ; if ($?) { g++ CD12.cpp -o CD12 } ; if ($?) { .\CD12 }

Read the Infix Expression : a+b

Given Infix Expn: a+b Prefix Expn: +ab
PS C:\Users\rvais\Desktop>

```

Result: The Program Executed successfully

Program: // Postfix

```

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

```

```

while(( x =pop() ) != '(' )
printf("%c:",x);
}
else
{
//check greater priority operator.
while(priority(stack[top]) >= priority(*e) )
printf("%c", pop());
push(*e);
}
e++;
}
while(top != -1)
{
printf("%c",pop());
}
return 0;
}

```

OUTPUT:

The screenshot shows a code editor with the following C code (lines 55-65):

```

55 push(*e);
56 }
57 e++;
58 }
59 while(top != -1)
60 {
61 printf("%c",pop());
62 }
63 return 0;
64 }
65

```

Below the code editor is a terminal window titled "Input". The terminal shows the following output:

```

Enter the expression:a+b-c
ab+c-

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

```

Result: The Program Executed successfully

EXPERIMENT 11

Intermediate code generation – Quadruple, Triple, Indirect triple

Aim: Write a program in C/C++ to implement intermediate code generation - Quadruple, Triple, Indirect triple.

Algorithm:

The algorithm takes a sequence of three-address statements as input. For each three address statements of the form $a := b \text{ op } c$ perform the various actions. These are as follows:

1. Invoke a function `getreg` to find out the location L where the result of computation $b \text{ op } c$ should be stored.
2. Consult the address description for y to determine y' . If the value of y currently in memory and register both then prefer the register y' . If the value of y is not already in L then generate the instruction `MOV y' , L` to place a copy of y in L .
3. Generate the instruction `OP z' , L` where z' is used to show the current location of z . if z is in both then prefer a register to a memory location. Update the address descriptor of x to indicate that x is in location L . If x is in L then update its descriptor and remove x from all other descriptors.
4. If the current value of y or z have no next uses or not live on exit from the block or in register then alter the register descriptor to indicate that after execution of $x := y \text{ op } z$ those register will no longer contain y or z .

Program:

```
#include<stdio.h>
#include<ctype.h>
#include<stdlib.h>
#include<string.h>
void small();
void dove(int i);
int p[5]={0,1,2,3,4},c=1,i,k,l,m,pi;
char sw[5]={'=','-','+','/','*'},j[20],a[5],b[5],ch[2];
void main()
{
    printf("Enter the expression : ");
    scanf("%s",j);
    printf("The Intermediate code is :\n");
    small();
}
void dove(int i)
{
    a[0]=b[0]='\0';
    if(!isdigit(j[i+2])&&!isdigit(j[i-2]))
    {
```

```

        a[0]=j[i-1];
        b[0]=j[i+1];
    }
    if(isdigit(j[i+2]))
    {
        a[0]=j[i-1];
        b[0]='t';
        b[1]=j[i+2];
    }
    if(isdigit(j[i-2]))
    {
        b[0]=j[i+1];
        a[0]='t';
        a[1]=j[i-2];
        b[1]='\0';
    }
    if(isdigit(j[i+2]) &&isdigit(j[i-2]))
    {
        a[0]='t';
        b[0]='t';
        a[1]=j[i-2];
        b[1]=j[i+2];
        sprintf(ch,"%d",c);
        j[i+2]=j[i-2]=ch[0];
    }
    if(j[i]=='*')
        printf("t%d=%s*%s\n",c,a,b);
    if(j[i]=='/')
        printf("t%d=%s/%s\n",c,a,b);
    if(j[i]=='+')
        printf("t%d=%s+%s\n",c,a,b);if(j[i]=='-')
        printf("t%d=%s-%s\n",c,a,b);
    if(j[i]=='=')
        printf("%c=t%d",j[i-1],--c);
    sprintf(ch,"%d",c);
    j[i]=ch[0];
    c++;
    small();
}
void small()
{
    pi=0;l=0;
    for(i=0;i<strlen(j);i++)
    {
        for(m=0;m<5;m++)
            if(j[i]==sw[m])

```

```

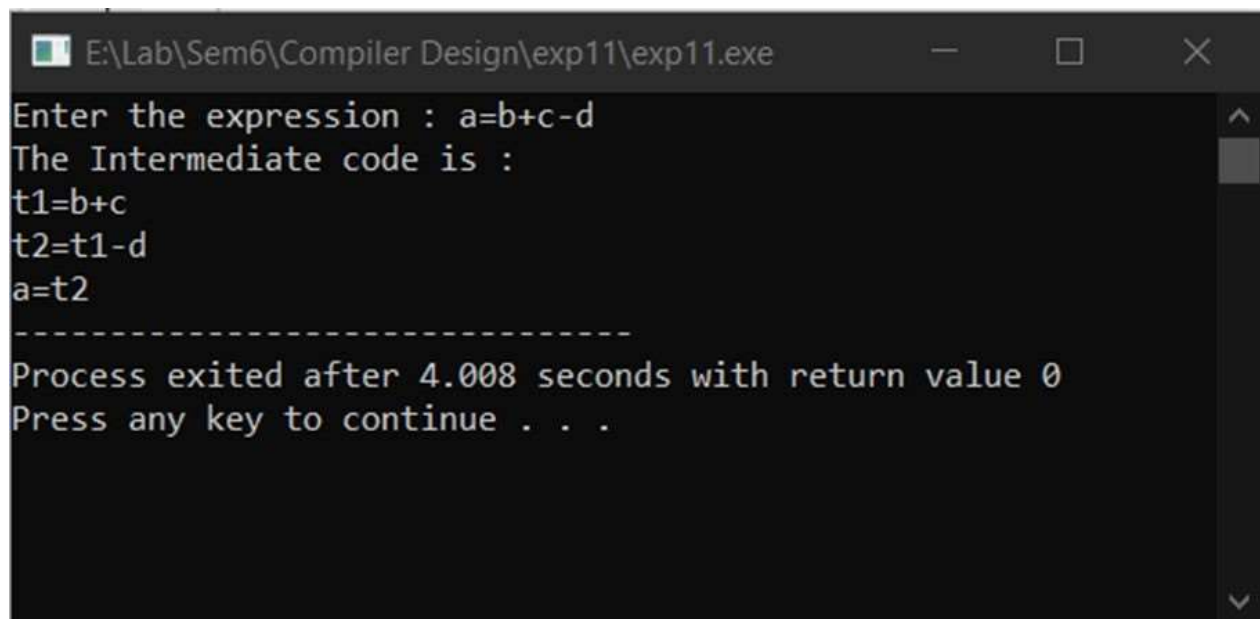
    if(pi<=p[m])
    {
        pi=p[m];
        l=1;
        k=i;
    }
}
if(l==1)
dove(k);
else
exit(0);
}

```

INPUT:

a=b+c-d

OUTPUT: -



```

E:\Lab\Sem6\Compiler Design\exp11\exp11.exe
Enter the expression : a=b+c-d
The Intermediate code is :
t1=b+c
t2=t1-d
a=t2
-----
Process exited after 4.008 seconds with return value 0
Press any key to continue . . .

```

Result: A program to implement intermediate code generation - Quadruple, Triple, Indirect triple has been compiled and run successfully.

EXPERIMENT 12

Implementation of Code Generator

Aim: To write a C program to implement Simple Code Generator.

Algorithm:

Input: Set of three address code sequence.

Output: Assembly code sequence for three address codes (opd1=opd2, op, opd3).

- 1- Start
- 2- Get address code sequence.
- 3- Determine current location of 3 using address (for 1st operand).
- 4- If current location not already exist generate move (B,O).
- 5- Update address of A(for 2nd operand).
- 6- If current value of B and () is null,exist.
- 7- If they generate operator () A,3 ADPR.
- 8- Store the move instruction in memory 9-
- 9-Stop.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>
#include<graphics.h>
typedef struct
{
char var[10];
int alive;
}regist;
regist preg[10];
void substring(char exp[],int st,int end)
{
int i,j=0;
char dup[10]="";
for(i=st;i<end;i++)
dup[j++]=exp[i];
dup[j]='\0';
strcpy(exp,dup);
}
int getregister(char var[])
{
int i;
for(i=0;i<10;i++)
{
if(preg[i].alive==0)
```

```

strcpy(preg[i].var,var);
break;
}
}
return(i);
}
void getvar(char exp[],char v[])
{
int i,j=0;
char var[10]="";
for(i=0;exp[i]!='\0';i++)
if(isalpha(exp[i]))
var[j++]=exp[i];
else
break;
strcpy(v,var);
}
void main()
{
char basic[10][10],var[10][10],fstr[10],op;
int i,j,k,reg,vc,flag=0;
clrscr();
printf("\nEnter the Three Address
Code:\n"); for(i=0;;i++)
{
gets(basic[i]);
if(strcmp(basic[i],"exit")==0)
break;
}
printf("\nThe Equivalent Assembly Code is:\n");
for(j=0;j<i;j++)
{
getvar(basic[j],var[vc++]);
strcpy(fstr,var[vc-1]);
substring(basic[j],strlen(var[vc-
1])+1,strlen(basic[j])); getvar(basic[j],var[vc++]);
reg=getregister(var[vc-1]);
if(preg[reg].alive==0)
{
printf("\nMov R%d,%s",reg,var[vc-
1]); preg[reg].alive=1;
}
op=basic[j][strlen(var[vc-1])];
substring(basic[j],strlen(var[vc-
1])+1,strlen(basic[j])); getvar(basic[j],var[vc++]);
switch(op)

```



```

{
case '+': printf("\nAdd"); break;
case '-': printf("\nSub"); break;
case '*': printf("\nMul"); break;
case '/': printf("\nDiv"); break;
}
flag=1;
for(k=0;k<=reg;k++)
{
if(strcmp(preg[k].var,var[vc-1])==0)
{
printf("R%d, R%d",k,reg);
preg[k].alive=0;
flag=0;
break;
}
}
if(flag)
{
printf(" %s,R%d",var[vc-1],reg);
printf("\nMov %s,R%d",fstr,reg);
}
strcpy(preg[reg].var,var[vc-3]);
getch();
}
}

```

Output:

Enter the Three Address Code:

a=b+c

c=a*c

exit

The Equivalent Assembly Code is:

Mov R0,b

Add c,R0

Mov a,R0

Mov R1,a

Mul c,R1

Mov c,R1

Result: The Program Executed successfully.

EXPERIMENT 13

Implementation of DAG

Aim: Write a c or c++ or java to implement DAG for input expression.

Program:

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string exp;
    cout<<"Enter the expression:-";
    cin>>exp;
    int j=0,k=0;
    char q;
    for(int i=exp.length()-1;i>1;i--)
    {
        if(islower(exp[i]) || (exp[i]>=48 && exp[i]<=57))
        {
            cout<<j<<"-"<<exp[i]<<endl;
            j++;
        }
    }
    for(int i=exp.length()-1;i>1;i--)
    {
        if(!(islower(exp[i]) || (exp[i]>=48 && exp[i]<=57)))
        {
            cout<<j<<"-"<<exp[i]<<k<<k+1<<endl;
            j++;
            k+=2;
        }
    }
    cout<<j<<"-"<<exp[0]<<endl;
    j++;
    cout<<j<<"-"<<exp[1]<<j-1<<j-2<<endl;
    return 0;
}
```

OUTPUT:

```
1 #include<iostream>
2 #include<string>
3 using namespace std;
4 int main()
5 {
6     string exp;
7     cout<<"Enter the expression: ";
8     cin>>exp;
9     int i=0;
10    while(i<exp.length())
11    {
12        if(exp[i]=='+' || exp[i]=='-')
13        {
14            int a=0;
15            int b=0;
16            int c=0;
17            int d=0;
18            int e=0;
19            int f=0;
20            int g=0;
21            int h=0;
22            int i=0;
23            int j=0;
24            int k=0;
25            int l=0;
26            int m=0;
27            int n=0;
28            int o=0;
29            int p=0;
30            int q=0;
31            int r=0;
32            int s=0;
33            int t=0;
34            int u=0;
35            int v=0;
36            int w=0;
37            int x=0;
38            int y=0;
39            int z=0;
40            int A=0;
41            int B=0;
42            int C=0;
43            int D=0;
44            int E=0;
45            int F=0;
46            int G=0;
47            int H=0;
48            int I=0;
49            int J=0;
50            int K=0;
51            int L=0;
52            int M=0;
53            int N=0;
54            int O=0;
55            int P=0;
56            int Q=0;
57            int R=0;
58            int S=0;
59            int T=0;
60            int U=0;
61            int V=0;
62            int W=0;
63            int X=0;
64            int Y=0;
65            int Z=0;
66            int a=0;
67            int b=0;
68            int c=0;
69            int d=0;
70            int e=0;
71            int f=0;
72            int g=0;
73            int h=0;
74            int i=0;
75            int j=0;
76            int k=0;
77            int l=0;
78            int m=0;
79            int n=0;
80            int o=0;
81            int p=0;
82            int q=0;
83            int r=0;
84            int s=0;
85            int t=0;
86            int u=0;
87            int v=0;
88            int w=0;
89            int x=0;
90            int y=0;
91            int z=0;
92            int A=0;
93            int B=0;
94            int C=0;
95            int D=0;
96            int E=0;
97            int F=0;
98            int G=0;
99            int H=0;
100           int I=0;
101           int J=0;
102           int K=0;
103           int L=0;
104           int M=0;
105           int N=0;
106           int O=0;
107           int P=0;
108           int Q=0;
109           int R=0;
110           int S=0;
111           int T=0;
112           int U=0;
113           int V=0;
114           int W=0;
115           int X=0;
116           int Y=0;
117           int Z=0;
118           int a=0;
119           int b=0;
120           int c=0;
121           int d=0;
122           int e=0;
123           int f=0;
124           int g=0;
125           int h=0;
126           int i=0;
127           int j=0;
128           int k=0;
129           int l=0;
130           int m=0;
131           int n=0;
132           int o=0;
133           int p=0;
134           int q=0;
135           int r=0;
136           int s=0;
137           int t=0;
138           int u=0;
139           int v=0;
140           int w=0;
141           int x=0;
142           int y=0;
143           int z=0;
144           int A=0;
145           int B=0;
146           int C=0;
147           int D=0;
148           int E=0;
149           int F=0;
150           int G=0;
151           int H=0;
152           int I=0;
153           int J=0;
154           int K=0;
155           int L=0;
156           int M=0;
157           int N=0;
158           int O=0;
159           int P=0;
160           int Q=0;
161           int R=0;
162           int S=0;
163           int T=0;
164           int U=0;
165           int V=0;
166           int W=0;
167           int X=0;
168           int Y=0;
169           int Z=0;
170           int a=0;
171           int b=0;
172           int c=0;
173           int d=0;
174           int e=0;
175           int f=0;
176           int g=0;
177           int h=0;
178           int i=0;
179           int j=0;
180           int k=0;
181           int l=0;
182           int m=0;
183           int n=0;
184           int o=0;
185           int p=0;
186           int q=0;
187           int r=0;
188           int s=0;
189           int t=0;
190           int u=0;
191           int v=0;
192           int w=0;
193           int x=0;
194           int y=0;
195           int z=0;
196           int A=0;
197           int B=0;
198           int C=0;
199           int D=0;
200           int E=0;
201           int F=0;
202           int G=0;
203           int H=0;
204           int I=0;
205           int J=0;
206           int K=0;
207           int L=0;
208           int M=0;
209           int N=0;
210           int O=0;
211           int P=0;
212           int Q=0;
213           int R=0;
214           int S=0;
215           int T=0;
216           int U=0;
217           int V=0;
218           int W=0;
219           int X=0;
220           int Y=0;
221           int Z=0;
222           int a=0;
223           int b=0;
224           int c=0;
225           int d=0;
226           int e=0;
227           int f=0;
228           int g=0;
229           int h=0;
230           int i=0;
231           int j=0;
232           int k=0;
233           int l=0;
234           int m=0;
235           int n=0;
236           int o=0;
237           int p=0;
238           int q=0;
239           int r=0;
240           int s=0;
241           int t=0;
242           int u=0;
243           int v=0;
244           int w=0;
245           int x=0;
246           int y=0;
247           int z=0;
248           int A=0;
249           int B=0;
250           int C=0;
251           int D=0;
252           int E=0;
253           int F=0;
254           int G=0;
255           int H=0;
256           int I=0;
257           int J=0;
258           int K=0;
259           int L=0;
260           int M=0;
261           int N=0;
262           int O=0;
263           int P=0;
264           int Q=0;
265           int R=0;
266           int S=0;
267           int T=0;
268           int U=0;
269           int V=0;
270           int W=0;
271           int X=0;
272           int Y=0;
273           int Z=0;
274           int a=0;
275           int b=0;
276           int c=0;
277           int d=0;
278           int e=0;
279           int f=0;
280           int g=0;
281           int h=0;
282           int i=0;
283           int j=0;
284           int k=0;
285           int l=0;
286           int m=0;
287           int n=0;
288           int o=0;
289           int p=0;
290           int q=0;
291           int r=0;
292           int s=0;
293           int t=0;
294           int u=0;
295           int v=0;
296           int w=0;
297           int x=0;
298           int y=0;
299           int z=0;
300           int A=0;
301           int B=0;
302           int C=0;
303           int D=0;
304           int E=0;
305           int F=0;
306           int G=0;
307           int H=0;
308           int I=0;
309           int J=0;
310           int K=0;
311           int L=0;
312           int M=0;
313           int N=0;
314           int O=0;
315           int P=0;
316           int Q=0;
317           int R=0;
318           int S=0;
319           int T=0;
320           int U=0;
321           int V=0;
322           int W=0;
323           int X=0;
324           int Y=0;
325           int Z=0;
326           int a=0;
327           int b=0;
328           int c=0;
329           int d=0;
330           int e=0;
331           int f=0;
332           int g=0;
333           int h=0;
334           int i=0;
335           int j=0;
336           int k=0;
337           int l=0;
338           int m=0;
339           int n=0;
340           int o=0;
341           int p=0;
342           int q=0;
343           int r=0;
344           int s=0;
345           int t=0;
346           int u=0;
347           int v=0;
348           int w=0;
349           int x=0;
350           int y=0;
351           int z=0;
352           int A=0;
353           int B=0;
354           int C=0;
355           int D=0;
356           int E=0;
357           int F=0;
358           int G=0;
359           int H=0;
360           int I=0;
361           int J=0;
362           int K=0;
363           int L=0;
364           int M=0;
365           int N=0;
366           int O=0;
367           int P=0;
368           int Q=0;
369           int R=0;
370           int S=0;
371           int T=0;
372           int U=0;
373           int V=0;
374           int W=0;
375           int X=0;
376           int Y=0;
377           int Z=0;
378           int a=0;
379           int b=0;
380           int c=0;
381           int d=0;
382           int e=0;
383           int f=0;
384           int g=0;
385           int h=0;
386           int i=0;
387           int j=0;
388           int k=0;
389           int l=0;
390           int m=0;
391           int n=0;
392           int o=0;
393           int p=0;
394           int q=0;
395           int r=0;
396           int s=0;
397           int t=0;
398           int u=0;
399           int v=0;
400           int w=0;
401           int x=0;
402           int y=0;
403           int z=0;
404           int A=0;
405           int B=0;
406           int C=0;
407           int D=0;
408           int E=0;
409           int F=0;
410           int G=0;
411           int H=0;
412           int I=0;
413           int J=0;
414           int K=0;
415           int L=0;
416           int M=0;
417           int N=0;
418           int O=0;
419           int P=0;
420           int Q=0;
421           int R=0;
422           int S=0;
423           int T=0;
424           int U=0;
425           int V=0;
426           int W=0;
427           int X=0;
428           int Y=0;
429           int Z=0;
430           int a=0;
431           int b=0;
432           int c=0;
433           int d=0;
434           int e=0;
435           int f=0;
436           int g=0;
437           int h=0;
438           int i=0;
439           int j=0;
440           int k=0;
441           int l=0;
442           int m=0;
443           int n=0;
444           int o=0;
445           int p=0;
446           int q=0;
447           int r=0;
448           int s=0;
449           int t=0;
450           int u=0;
451           int v=0;
452           int w=0;
453           int x=0;
454           int y=0;
455           int z=0;
456           int A=0;
457           int B=0;
458           int C=0;
459           int D=0;
460           int E=0;
461           int F=0;
462           int G=0;
463           int H=0;
464           int I=0;
465           int J=0;
466           int K=0;
467           int L=0;
468           int M=0;
469           int N=0;
470           int O=0;
471           int P=0;
472           int Q=0;
473           int R=0;
474           int S=0;
475           int T=0;
476           int U=0;
477           int V=0;
478           int W=0;
479           int X=0;
480           int Y=0;
481           int Z=0;
482           int a=0;
483           int b=0;
484           int c=0;
485           int d=0;
486           int e=0;
487           int f=0;
488           int g=0;
489           int h=0;
490           int i=0;
491           int j=0;
492           int k=0;
493           int l=0;
494           int m=0;
495           int n=0;
496           int o=0;
497           int p=0;
498           int q=0;
499           int r=0;
500           int s=0;
501           int t=0;
502           int u=0;
503           int v=0;
504           int w=0;
505           int x=0;
506           int y=0;
507           int z=0;
508           int A=0;
509           int B=0;
510           int C=0;
511           int D=0;
512           int E=0;
513           int F=0;
514           int G=0;
515           int H=0;
516           int I=0;
517           int J=0;
518           int K=0;
519           int L=0;
520           int M=0;
521           int N=0;
522           int O=0;
523           int P=0;
524           int Q=0;
525           int R=0;
526           int S=0;
527           int T=0;
528           int U=0;
529           int V=0;
530           int W=0;
531           int X=0;
532           int Y=0;
533           int Z=0;
534           int a=0;
535           int b=0;
536           int c=0;
537           int d=0;
538           int e=0;
539           int f=0;
540           int g=0;
541           int h=0;
542           int i=0;
543           int j=0;
544           int k=0;
545           int l=0;
546           int m=0;
547           int n=0;
548           int o=0;
549           int p=0;
550           int q=0;
551           int r=0;
552           int s=0;
553           int t=0;
554           int u=0;
555           int v=0;
556           int w=0;
557           int x=0;
558           int y=0;
559           int z=0;
560           int A=0;
561           int B=0;
562           int C=0;
563           int D=0;
564           int E=0;
565           int F=0;
566           int G=0;
567           int H=0;
568           int I=0;
569           int J=0;
570           int K=0;
571           int L=0;
572           int M=0;
573           int N=0;
574           int O=0;
575           int P=0;
576           int Q=0;
577           int R=0;
578           int S=0;
579           int T=0;
580           int U=0;
581           int V=0;
582           int W=0;
583           int X=0;
584           int Y=0;
585           int Z=0;
586           int a=0;
587           int b=0;
588           int c=0;
589           int d=0;
590           int e=0;
591           int f=0;
592           int g=0;
593           int h=0;
594           int i=0;
595           int j=0;
596           int k=0;
597           int l=0;
598           int m=0;
599           int n=0;
600           int o=0;
601           int p=0;
602           int q=0;
603           int r=0;
604           int s=0;
605           int t=0;
606           int u=0;
607           int v=0;
608           int w=0;
609           int x=0;
610           int y=0;
611           int z=0;
612           int A=0;
613           int B=0;
614           int C=0;
615           int D=0;
616           int E=0;
617           int F=0;
618           int G=0;
619           int H=0;
620           int I=0;
621           int J=0;
622           int K=0;
623           int L=0;
624           int M=0;
625           int N=0;
626           int O=0;
627           int P=0;
628           int Q=0;
629           int R=0;
630           int S=0;
631           int T=0;
632           int U=0;
633           int V=0;
634           int W=0;
635           int X=0;
636           int Y=0;
637           int Z=0;
638           int a=0;
639           int b=0;
640           int c=0;
641           int d=0;
642           int e=0;
643           int f=0;
644           int g=0;
645           int h=0;
646           int i=0;
647           int j=0;
648           int k=0;
649           int l=0;
650           int m=0;
651           int n=0;
652           int o=0;
653           int p=0;
654           int q=0;
655           int r=0;
656           int s=0;
657           int t=0;
658           int u=0;
659           int v=0;
660           int w=0;
661           int x=0;
662           int y=0;
663           int z=0;
664           int A=0;
665           int B=0;
666           int C=0;
667           int D=0;
668           int E=0;
669           int F=0;
670           int G=0;
671           int H=0;
672           int I=0;
673           int J=0;
674           int K=0;
675           int L=0;
676           int M=0;
677           int N=0;
678           int O=0;
679           int P=0;
680           int Q=0;
681           int R=0;
682           int S=0;
683           int T=0;
684           int U=0;
685           int V=0;
686           int W=0;
687           int X=0;
688           int Y=0;
689           int Z=0;
690           int a=0;
691           int b=0;
692           int c=0;
693           int d=0;
694           int e=0;
695           int f=0;
696           int g=0;
697           int h=0;
698           int i=0;
699           int j=0;
700           int k=0;
701           int l=0;
702           int m=0;
703           int n=0;
704           int o=0;
705           int p=0;
706           int q=0;
707           int r=0;
708           int s=0;
709           int t=0;
710           int u=0;
711           int v=0;
712           int w=0;
713           int x=0;
714           int y=0;
715           int z=0;
716           int A=0;
717           int B=0;
718           int C=0;
719           int D=0;
720           int E=0;
721           int F=0;
722           int G=0;
723           int H=0;
724           int I=0;
725           int J=0;
726           int K=0;
727           int L=0;
728           int M=0;
729           int N=0;
730           int O=0;
731           int P=0;
732           int Q=0;
733           int R=0;
734           int S=0;
735           int T=0;
736           int U=0;
737           int V=0;
738           int W=0;
739           int X=0;
740           int Y=0;
741           int Z=0;
742           int a=0;
743           int b=0;
744           int c=0;
745           int d=0;
746           int e=0;
747           int f=0;
748           int g=0;
749           int h=0;
750           int i=0;
751           int j=0;
752           int k=0;
753           int l=0;
754           int m=0;
755           int n=0;
756           int o=0;
757           int p=0;
758           int q=0;
759           int r=0;
760           int s=0;
761           int t=0;
762           int u=0;
763           int v=0;
764           int w=0;
765           int x=0;
766           int y=0;
767           int z=0;
768           int A=0;
769           int B=0;
770           int C=0;
771           int D=0;
772           int E=0;
773           int F=0;
774           int G=0;
775           int H=0;
776           int I=0;
777           int J=0;
778           int K=0;
779           int L=0;
780           int M=0;
781           int N=0;
782           int O=0;
783           int P=0;
784           int Q=0;
785           int R=0;
786           int S=0;
787           int T=0;
788           int U=0;
789           int V=0;
790           int W=0;
791           int X=0;
792           int Y=0;
793           int Z=0;
794           int a=0;
795           int b=0;
796           int c=0;
797           int d=0;
798           int e=0;
799           int f=0;
800           int g=0;
801           int h=0;
802           int i=0;
803           int j=0;
804           int k=0;
805           int l=0;
806           int m=0;
807           int n=0;
808           int o=0;
809           int p=0;
810           int q=0;
811           int r=0;
812           int s=0;
813           int t=0;
814           int u=0;
815           int v=0;
816           int w=0;
817           int x=0;
818           int y=0;
819           int z=0;
820           int A=0;
821           int B=0;
822           int C=0;
823           int D=0;
824           int E=0;
825           int F=0;
826           int G=0;
827           int H=0;
828           int I=0;
829           int J=0;
830           int K=0;
831           int L=0;
832           int M=0;
833           int N=0;
834           int O=0;
835           int P=0;
836           int Q=0;
837           int R=0;
838           int S=0;
839           int T=0;
840           int U=0;
841           int V=0;
842           int W=0;
843           int X=0;
844           int Y=0;
845           int Z=0;
846           int a=0;
847           int b=0;
848           int c=0;
849           int d=0;
850           int e=0;
851           int f=0;
852           int g=0;
853           int h=0;
854           int i=0;
855           int j=0;
856           int k=0;
857           int l=0;
858           int m=0;
859           int n=0;
860           int o=0;
861           int p=0;
862           int q=0;
863           int r=0;
864           int s=0;
865           int t=0;
866           int u=0;
867           int v=0;
868           int w=0;
869           int x=0;
870           int y=0;
871           int z=0;
872           int A=0;
873           int B=0;
874           int C=0;
875           int D=0;
876           int E=0;
877           int F=0;
878           int G=0;
879           int H=0;
880           int I=0;
881           int J=0;
882           int K=0;
883           int L=0;
884           int M=0;
885           int N=0;
886           int O=0;
887           int P=0;
888           int Q=0;
889           int R=0;
890           int S=0;
891           int T=0;
892           int U=0;
893           int V=0;
894           int W=0;
895           int X=0;
896           int Y=0;
897           int Z=0;
898           int a=0;
899           int b=0;
900           int c=0;
901           int d=0;
902           int e=0;
903           int f=0;
904           int g=0;
905           int h=0;
906           int i=0;
907           int j=0;
908           int k=0;
909           int l=0;
910           int m=0;
911           int n=0;
912           int o=0;
913           int p=0;
914           int q=0;
915           int r=0;
916           int s=0;
917           int t=0;
918           int u=0;
919           int v=0;
920           int w=0;
921           int x=0;
922           int y=0;
923           int z=0;
924           int A=0;
925           int B=0;
926           int C=0;
927           int D=0;
928           int E=0;
929           int F=0;
930           int G=0;
931           int H=0;
932           int I=0;
933           int J=0;
934           int K=0;
935           int L=0;
936           int M=0;
937           int N=0;
938           int O=0;
939           int P=0;
940           int Q=0;
941           int R=0;
942           int S=0;
943           int T=0;
944           int U=0;
945           int V=0;
946           int W=0;
947           int X=0;
948           int Y=0;
949           int Z=0;
950           int a=0;
951           int b=0;
952           int c=0;
953           int d=0;
954           int e=0;
955           int f=0;
956           int g=0;
957           int h=0;
958           int i=0;
959           int j=0;
960           int k=0;
961           int l=0;
962           int m=0;
963           int n=0;
964           int o=0;
965           int p=0;
966           int q=0;
967           int r=0;
968           int s=0;
969           int t=0;
970           int u=0;
971           int v=0;
972           int w=0;
973           int x=0;
974           int y=0;
975           int z=0;
976           int A=0;
977           int B=0;
978           int C=0;
979           int D=0;
980           int E=0;
981           int F=0;
982           int G=0;
983           int H=0;
984           int I=0;
985           int J=0;
986           int K=0;
987           int L=0;
988           int M=0;
989           int N=0;
990           int O=0;
991           int P=0;
992           int Q=0;
993           int R=0;
994           int S=0;
995           int T=0;
996           int U=0;
997           int V=0;
998           int W=0;
999           int X=0;
1000          int Y=0;
1001          int Z=0;
1002          int a=0;
1003          int b=0;
1004          int c=0;
1005          int d=0;
1006          int e=0;
1007          int f=0;
1008          int g=0;
1009          int h=0;
1010          int i=0;
1011          int j=0;
1012          int k=0;
1013          int l=0;
1014          int m=0;
1015          int n=0;
1016          int o=0;
1017          int p=0;
1018          int q=0;
1019          int r=0;
1020          int s=0;
1021          int t=0;
1022          int u=0;
1023          int v=0;
1024          int w=0;
1025          int x=0;
1026          int y=0;
1027          int z=0;
1028          int A=0;
1029          int B=0;
1030          int C=0;
1031          int D=0;
1032          int E=0;
1033          int F=0;
1034          int G=0;
1035          int H=0;
1036          int I=0;
1037          int J=0;
1038          int K=0;
1039          int L=0;
1040          int M=0;
1041          int N=0;
1042          int O=0;
1043          int P=0;
1044          int Q=0;
1045          int R=0;
1046          int S=0;
1047          int T=0;
1048          int U=0;
1049          int V=0;
1050          int W=0;
1051          int X=0;
1052          int Y=0;
1053          int Z=0;
1054          int a=0;
1055          int b=0;
1056          int c=0;
1057          int d=0;
1058          int e=0;
1059          int f=0;
1060          int g=0;
1061          int h=0;
1062          int i=0;
1063          int j=0;
1064          int k=0;
1065          int l=0;
1066          int m=0;
1067          int n=0;
1068          int o=0;
1069          int p=0;
1070          int q=0;
1071          int r=0;
1072          int s=0;
1073          int t=0;
1074          int u=0;
1075          int v=0;
1076          int w=0;
1077          int x=0;
1078          int y=0;
1079          int z=0;
1080          int A=0;
1081          int B=0;
1082          int C=0;
1083          int D=0;
1084          int E=0;
1085          int F=0;
1086          int G=0;
1087          int H=0;
1088          int I=0;
1089          int J=0;
1090          int K=0;
1091          int L=0;
1092          int M=0;
1093          int N=0;
1094          int O=0;
1095          int P=0;
1096          int Q=0;
1097          int R=0;
1098          int S=0;
1099          int T=0;
1100          int U=0;
1101          int V=0;
1102          int W=0;
1103          int X=0;
1104          int Y=0;
1105          int Z=0;
1106          int a=0;
1107          int b=0;
1108          int c=0;
1109          int d=0;
1110          int e=0;
1111          int f=0;
1112          int g=0;
1113          int h=0;
1114          int i=0;
1115          int j=0;
1116          int k=0;
1117          int l=0;
1118          int m=0;
1119          int n=0;
1120          int o=0;
1121          int p=0;
1122          int q=0;
1123          int r=0;
1124          int s=0;
1125          int t=0;
1126          int u=0;
1127          int v=0;
1128          int w=0;
1129          int x=0;
1130          int y=0;
1131          int z=0;
1132          int A=0;
1133          int B=0;
1134          int C=0;
1135          int D=0;
1136          int E=0;
1137          int F=0;
1138          int G=0;
1139          int H=0;
1140          int I=0;
1141          int J=0;
1142          int K=0;
1143          int L=0;
1144          int M=0;
1145          int N=0;
1146          int O=0;
1147          int P=0;
1148          int Q=0;
1149          int R=0;
1150          int S=0;
1151          int T=0;
1152          int U=0;
1153          int V=0;
1154          int W=0;
1155          int X=0;
1156          int Y=0;
1157          int Z=0;
1158          int a=0;
1159          int b=0;
1160          int c=0;
1161          int d=0;
1162          int e=0;
1163          int f=0;
1164          int g=0;
1165          int h=0;
1166          int i=0;
1167          int j=0;
1168          int k=0;
1169          int l=0;
1170          int m=0;
1171          int n=0;
1172          int o=0;
1173          int p=0;
1174          int q=0;
1175          int r=0;
1176          int s=0;
1177          int t=0;
1178          int u=0;
1179          int v=0;
1180          int w=0;
1181          int x=0;
1182          int y=0;
1183          int z=0;
1184          int A=0;
1185          int B=0;
1186          int C=0;
1187          int D=0;
1188          int E=0;
1189          int F=0;
1190          int G=0;
1191          int H=0;
1192          int I=0;
1193          int J=0;
1194          int K=0;
1195          int L=0;
1196          int M=0;
1197          int N=0;
1198          int O=0;
1199          int P=0;
1200          int Q=0;
1201          int R=0;
1202          int S=0;
1203          int T=0;
1204          int U=0;
1205          int V=0;
1206          int W=0;
1207          int X=0;
1208          int Y=0;
1209          int Z=0;
1210          int a=0;
1211          int b=0;
1212          int c=0;
1213          int d=0;
1214          int e=0;
1215          int f=0;
1216          int g=0;
1217          int h=0;
1218          int i=0;
1219          int j=0;
1220          int k=0;
1221          int l=0;
1222          int m=0;
1223          int n=0;
1224          int o=0;

```

VALUE ADDED-1

Implementation of Global Data Flow Analysis

Aim: Write a program in c to implement Global Data Flow Analysis.

Program:

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
void input();
void output();
void change(int p,int q,char *res);
void constant();
void expression();
struct expr
{
char op[2],op1[5],op2[5],res[5];
int flag;
}arr[10];
int n;
int main()
{
int ch=0;
input();
constant();
expression();
output();
}
void input()
{
int i;
printf("\n\nEnter the maximum number of expressions:");
scanf("%d",&n);
printf("\nEnter the input : \n");
for(i=0;i<n;i++)
{
scanf("%s",arr[i].op);
scanf("%s",arr[i].op1);
scanf("%s",arr[i].op2);
scanf("%s",arr[i].res);
arr[i].flag=0;
}
}
void constant()
{
int i;
int op1,op2,res;
char op,res1[5];
for(i=0;i<n;i++)
{
if(isdigit(arr[i].op1[0]) && isdigit(arr[i].op2[0]))
{
op1=atoi(arr[i].op1);
op2=atoi(arr[i].op2);
op=arr[i].op[0];
```

```

switch(op)
{
case '+':
res=op1+op2;
break;
case '-':
res=op1-op2;
break;
case '*':
res=op1*op2;
break;
case '/':
res=op1/op2;
break;
}
sprintf(res1,"%d",res);
arr[i].flag=1;
change(i,i,res1);
}
}

void expression()
{
int i,j;
for(i=0;i<n;i++)
{
for(j=i+1;j<n;j++)
{
if(strcmp(arr[i].op,arr[j].op)==0)
{
if(strcmp(arr[i].op,"+")==0||strcmp(arr[i].op,"*")==0)
{
if(strcmp(arr[i].op1,arr[j].op1)==0&&strcmp(arr[i].op2,arr[j].op2)==0 ||
strcmp(arr[i].op1,arr[j].op2)==0&&strcmp(arr[i].op2,arr[j].op1)==0)
{
arr[j].flag=1;
change(i,j,NULL);
}
}
else
{
if(strcmp(arr[i].op1,arr[j].op1)==0&&strcmp(arr[i].op2,arr[j].op2)==0)
{
arr[j].flag=1;
change(i,j,NULL);
}
}
}
}
}

void output()
{
int i=0;
printf("\nOptimized code is : ");
for(i=0;i<n;i++)
{
if(!arr[i].flag)
{
printf("\n%s %s %s %s\n",arr[i].op,arr[i].op1,arr[i].op2,arr[i].res);
}
}
}

```

```

}
}
void change(int p,int q,char *res)
{
int i;
for(i=q+1;i<n;i++)
{
if(strcmp(arr[q].res,arr[i].op1)==0)
if(res == NULL)
strcpy(arr[i].op1,arr[p].res);
else
strcpy(arr[i].op1,res);
else if(strcmp(arr[q].res,arr[i].op2)==0)
if(res == NULL)
strcpy(arr[i].op2,arr[p].res);
else
strcpy(arr[i].op2,res);
}
}
}

```

OUTPUT:

Enter the maximum number of expressions: 5

Enter the input :

```

+ 4 2 t1
+ a t1 t2
- b a t3
+ a 6 t4
+ t3 t2 t4

```

Optimized code is :

```

+ a 6 t2
- b a t3
+ t3 t2 t4

```

Result: The Program Executed successfully.

VALUE ADDED-2

Implement any one storage allocation strategies (heap, stack, static)

Aim: Write a program in c to implement Stack storage allocation strategies.

Algorithm:

Step1: Initially check whether the stack is empty
Step2: Insert an element into the stack using push operation
Step3: Insert more elements onto the stack until stack becomes full
Step4: Delete an element from the stack using pop operation
Step5: Display the elements in the stack
Step6: Top the stack element will be displayed

Program:

```
//implementation of heap allocation storage strategies//
#include<stdio.h>
#include<stdlib.h>
#define TRUE 1
#define FALSE 0
typedef struct Heap
{
    int data;
    struct Heap *next;
}
node;
node *create();
void main()
{
    int choice,val;
    char ans;
    node *head;
    void display(node *);
    node *search(node *,int);
    node *insert(node *);
    void dele(node **);
    head=NULL;
    do
    {
        printf("\nprogram to perform various operations on heap using dynamic memory management");
        printf("\n1.create");
        printf("\n2.display");
        printf("\n3.insert an element in a list");
        printf("\n4.delete an element from list");
        printf("\n5.quit");
        printf("\nenter your chioce(1-5)");
```



```

scanf("%d",&choice);
switch(choice)
{
case 1:head=create();
break;
case 2:display(head);
break;
case 3:head=insert(head);
break;
case 4:dele(&head);
break;
case 5:exit(0);
default:
printf("invalid choice,try again");
}
}
while(choice!=5);
}
node* create()
{
node *temp,*New,*head;
int val,flag;
char ans='y';
node *get_node();
temp=NULL;
flag=TRUE;
do
{
printf("\n enter the element:");
scanf("%d",&val);
New=get_node();
if(New==NULL)
printf("\nmemory is not allocated");
New->data=val;
if(flag==TRUE)
{
head=New;
temp=head;
flag=FALSE;
}
else
{
temp->next=New;
temp=New;
}
}
printf("\ndo you want to enter more elements?(y/n)");

```

```

}
while(ans=='y');
printf("\nthe list is created\n");
return head;
}
node *get_node()
{
node *temp;
temp=(node*)malloc(sizeof(node));
temp->next=NULL;
return temp;
}
void display(node *head)
{
node *temp;
temp=head;
if(temp==NULL)
{
printf("\nthe list is empty\n");
return;
}
while(temp!=NULL)
{
printf("%d->",temp->data);
temp=temp->next;
}
printf("NULL");
}
node *search(node *head,int key)
{
node *temp;
int found;
temp=head;
if(temp==NULL)
{
printf("the linked list is empty\n");
return NULL;
}
found=FALSE;
while(temp!=NULL && found==FALSE)
{
if(temp->data!=key)
temp=temp->next;
else
found=TRUE;
}
}

```

```

if(found==TRUE)
{
printf("\nthe element is present in the list\n");
return temp;
}
else
{
printf("the element is not present in the list\n");
return NULL;
}
}
node *insert(node *head)
{
int choice;
node *insert_head(node *);
void insert_after(node *);
void insert_last(node *);
printf("n1.insert a node as a head node");
printf("n2.insert a node as a head node");
printf("n3.insert a node at intermediate position in t6he list");
printf("\nenter your choice for insertion of node:");
scanf("%d",&choice);
switch(choice)
{
case 1:head=insert_head(head);
break;
case 2:insert_last(head);
break;
case 3:insert_after(head);
break;
}
return head;
}
node *insert_head(node *head)
{
node *New,*temp;
New=get_node();
printf("\nEnter the element which you want to insert");
scanf("%d",&New->data);
if(head==NULL)
head=New;
else
{
temp=head;
New->next=temp;
head=New;
}
}

```

```

    }
    return head;
}
void insert_last(node *head)
{
    node *New,*temp;
    New=get_node();
    printf("\nenter the element which you want to insert");
    scanf("%d",&New->data);
    if(head==NULL)
        head=New;
    else
    {
        temp=head;
        while(temp->next!=NULL)
            temp=temp->next;
        temp->next=New;
        New->next=NULL;
    }
}
void insert_after(node *head)
{
    int key;
    node *New,*temp;
    New=get_node();
    printf("\nenter the elements which you want to insert");
    scanf("%d",&New->data);
    if(head==NULL)
    {
        head=New;
    }
    else
    {
        printf("\nenter the element which you want to insert the node");
        scanf("%d",&key);
        temp=head;
        do
        {
            if(temp->data==key)
            {
                New->next=temp->next;
                temp->next=New;
                return;
            }
            else
                temp=temp->next;
        }
    }
}

```

```

    }
    while(temp!=NULL);
    }
    }
    node *get_prev(node *head,int val)
    {
        node *temp,*prev;
        int flag;
        temp=head;
        if(temp==NULL)
            return NULL;
        flag=FALSE;
        prev=NULL;
        while(temp!=NULL && ! flag)
        {
            if(temp->data!=val)
            {
                prev=temp;
                temp=temp->next;
            }
            else
                flag=TRUE;
        }
        if(flag)
            return prev;
        else
            return NULL;
    }
    void dele(node **head)
    {
        node *temp,*prev;
        int key;
        temp=*head;
        if(temp==NULL)
        {
            printf("\nthe list is empty\n");
            return;
        }
        printf("\nenter the element you want to delete:");
        scanf("%d",&key);
        temp=search(*head,key);
        if(temp!=NULL)
        {
            prev=get_prev(*head,key);
            if(prev!=NULL)
            {

```

```

prev->next=temp->next;
free(temp);
}
else
{
*head=temp->next;
free(temp);
}
printf("\nthe element is deleted\n");

}
}

```

Output:

program to perform various operations on heap using dynamic memory management

```

1.create
2.display
3.insert an element in a list
4.delete an element from list
5.quit
enter your chioce(1-5)2

```

the list is empty

program to perform various operations on heap using dynamic memory management

```

1.create
2.display
3.insert an element in a list
4.delete an element from list
5.quit
enter your chioce(1-5)5

```

Result: The Program Executed successfully.