



# NCR Campus, Modinagar

(A Constituent SRM IST, Chennai, T.N.)

Delhi-Meerut Road, Sikari Kalan, Modinagar – 201204, GHAZIABAD (U.P.)

## Compiler Design Lab (18CSC304J)

### Lab Record

(Jan – May 2022)

Name	Hardik Gupta
Reg No.	RA1911026030027
Degree / Branch	B-Tech, CSE with AI and ML
Semester / Section	6 <sup>th</sup> Semester, A
Course Code	18CSC304J
Course Title	Compiler Design
Faculty	Ms. Bharti Vidhury

**SRM IST, DELHI-NCR CAMPUS, MODINAGAR**

*Department Of Computer Science and Engineering*



Registration Number

R	A	1	9	1	1	0	2	6	0	3	0	0	2	7
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

### BONAFIDE CERTIFICATE

It is to be certified that the bonafide practical record submitted by **Hardik Gupta** of 6<sup>th</sup> semester for Bachelor of Technology degree in the Department of Computer Science and Engineering, Delhi-NCR Campus, SRM IST has been done for the course **Compiler Design Lab (18CSC304J)** during the academic semester session Jan 2022 – May 2022.

-----  
**Dr. R. P. Mahapatra**

Head of the Department  
Computer Science & Engg.

-----  
**Ms. Bharti Vidhury**

Assistant Professor  
Computer Science & Engg.

*Submitted for the University Examination held on* \_\_\_\_\_

Examiner 1

Examiner 2

## **INDEX**

<b>S.N.</b>	<b>Name of Experiment</b>	<b>Page No.</b>	<b>Date of Exp.</b>	<b>Signature</b>
<b>1</b>	Implementation of Lexical Analyzer	4-6	25-01-2022	
<b>2</b>	Regular Expression to NFA	7-9	1-01-2022	
<b>3</b>	RE to NFA to DFA	10-10	8-02-2022	
<b>4</b>	Computation of FIRST in a grammar	11-14	15-02-2022	
<b>5</b>	Computation of FOLLOW in a grammar	15-17	23-02-2022	
<b>6</b>	Computation of Predictive Parsing	18-20	1-03-2022	
<b>7</b>	Computation of Shift Reduce Parsing	21-26	08-03-2022	
<b>8</b>	Program for finding the leading and trailing.	27-31	15-03-2022	
<b>9</b>	Implementation of 3-Address Code using Quadruple	32-33	22-03-2022	
<b>10</b>	Intermediate Code Generation	34-37	30-03-2022	
<b>11</b>	Intermediate code generation-Postfix Expression	38-40	12-04-2022	
<b>12</b>	Intermediate code generation-Prefix Expression	41-43	19-04-2022	
<b>13</b>	Construction of DAG	44-45	26-04-2022	
<b>14</b>	Recursive Descent Parsing	46-47	3-05-2022	

## 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", "co
ntinue" };
    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:

```

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

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>

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>

```

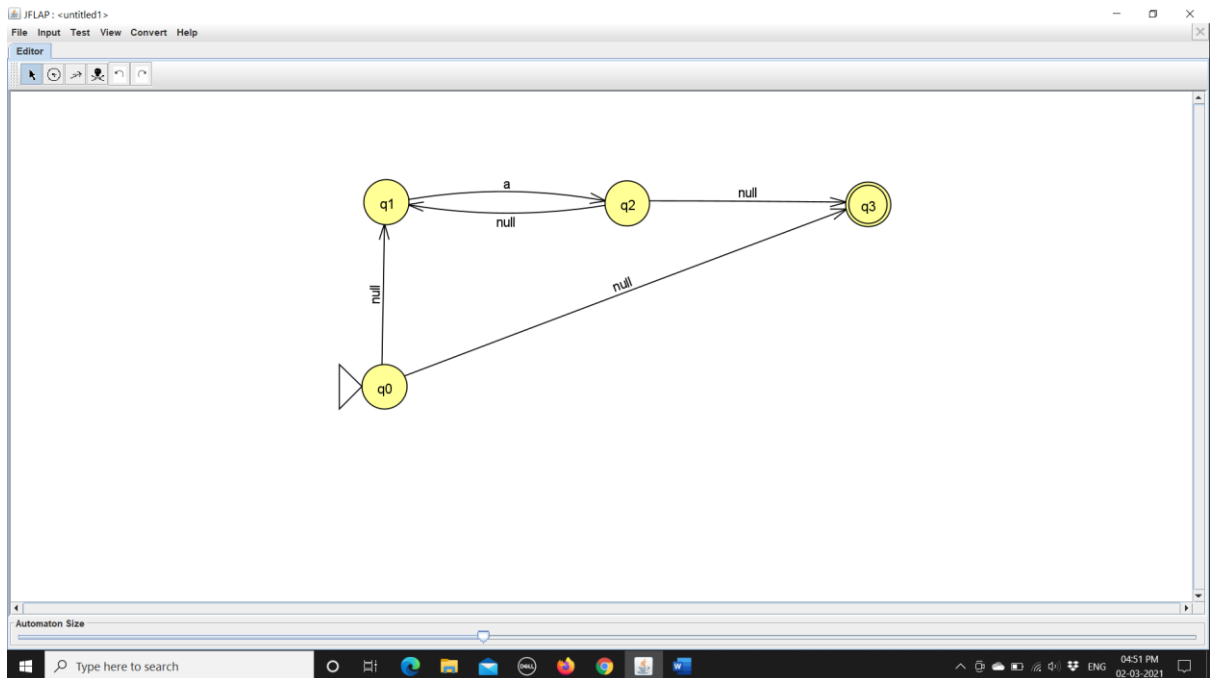
**Result:** The Program Executed successfully.

## EXPERIMENT 2

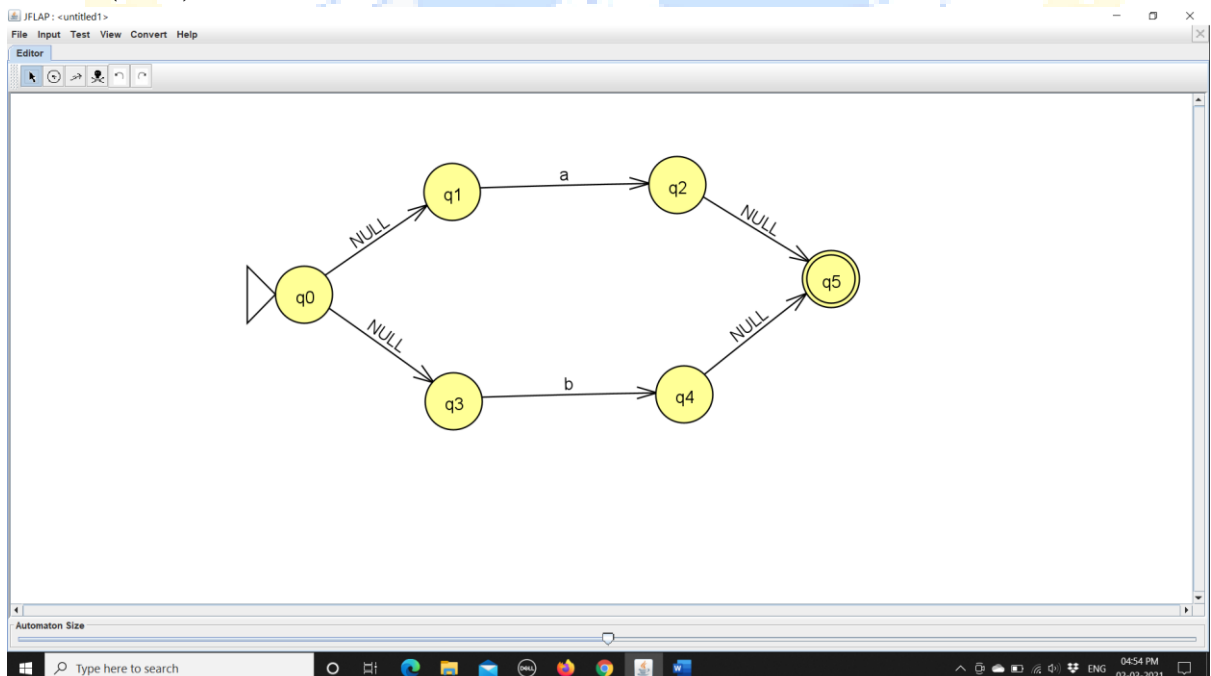
### Regular Expression to NFA

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

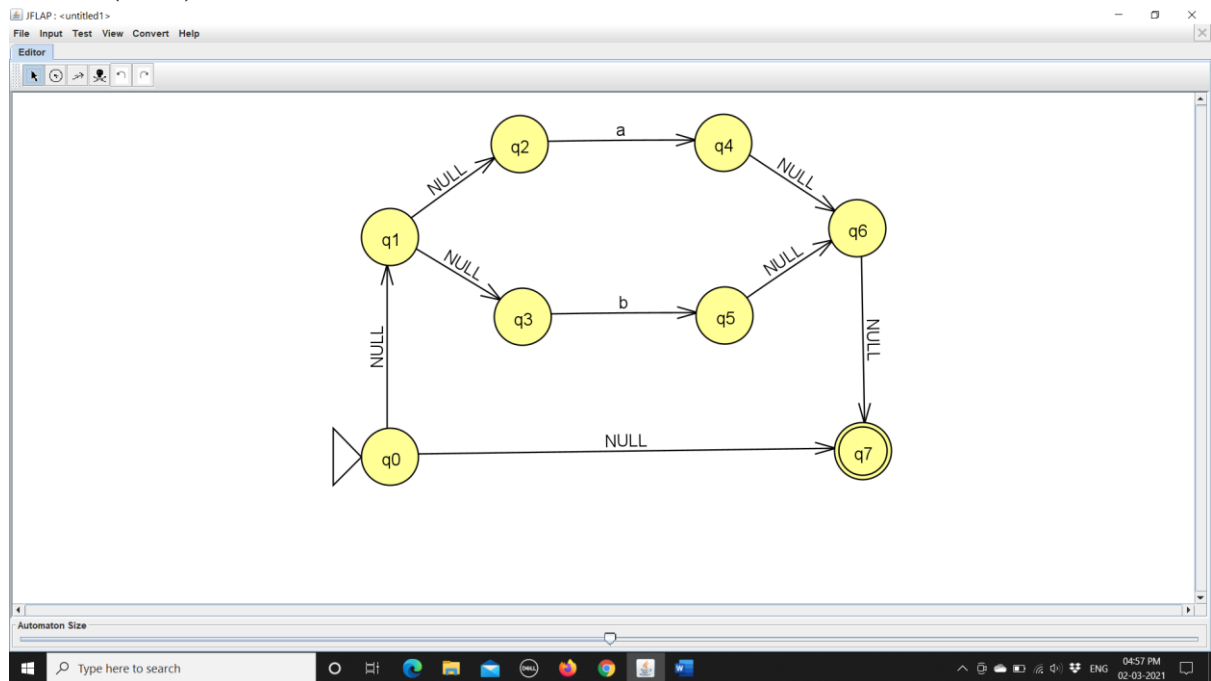
1.  $a^*$



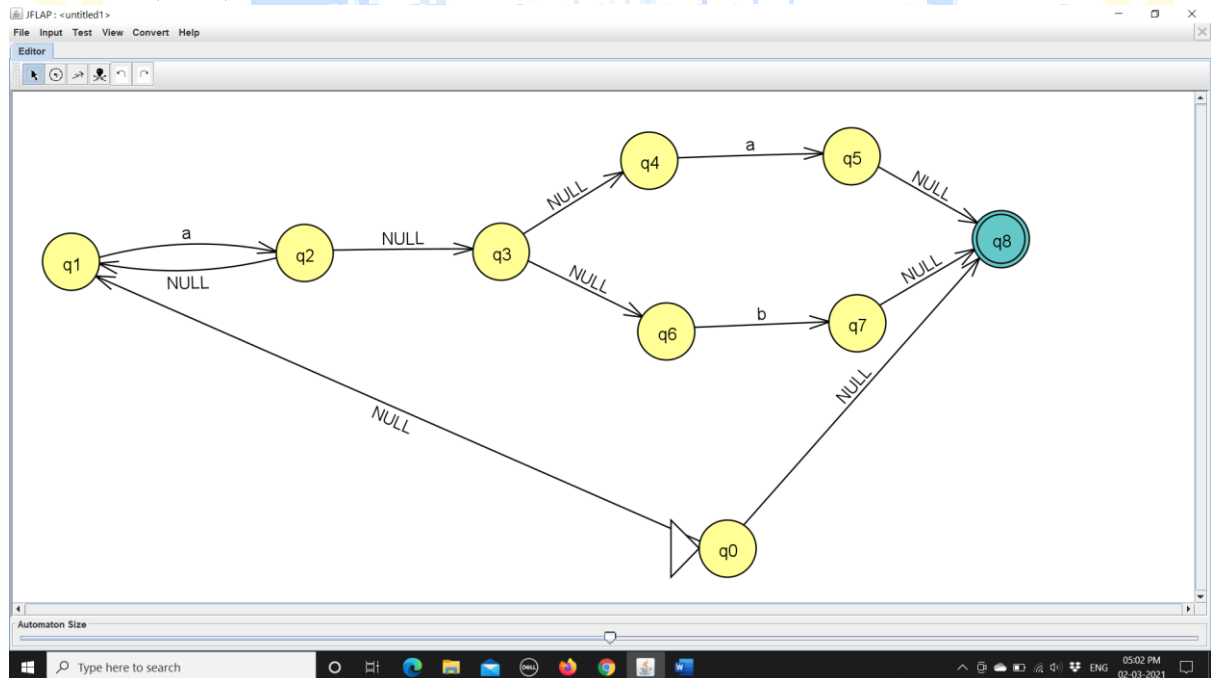
2.  $(a+b)$



### 3. $(a+b)^*$

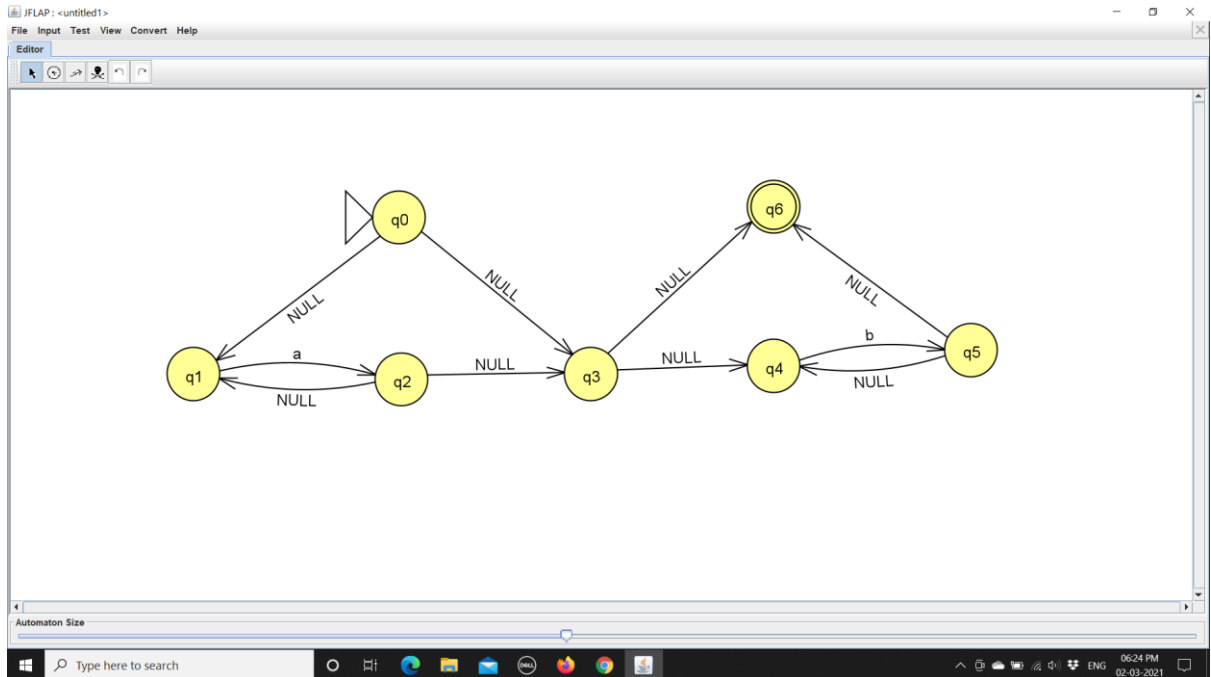


### 4. $a^*(a+b)$

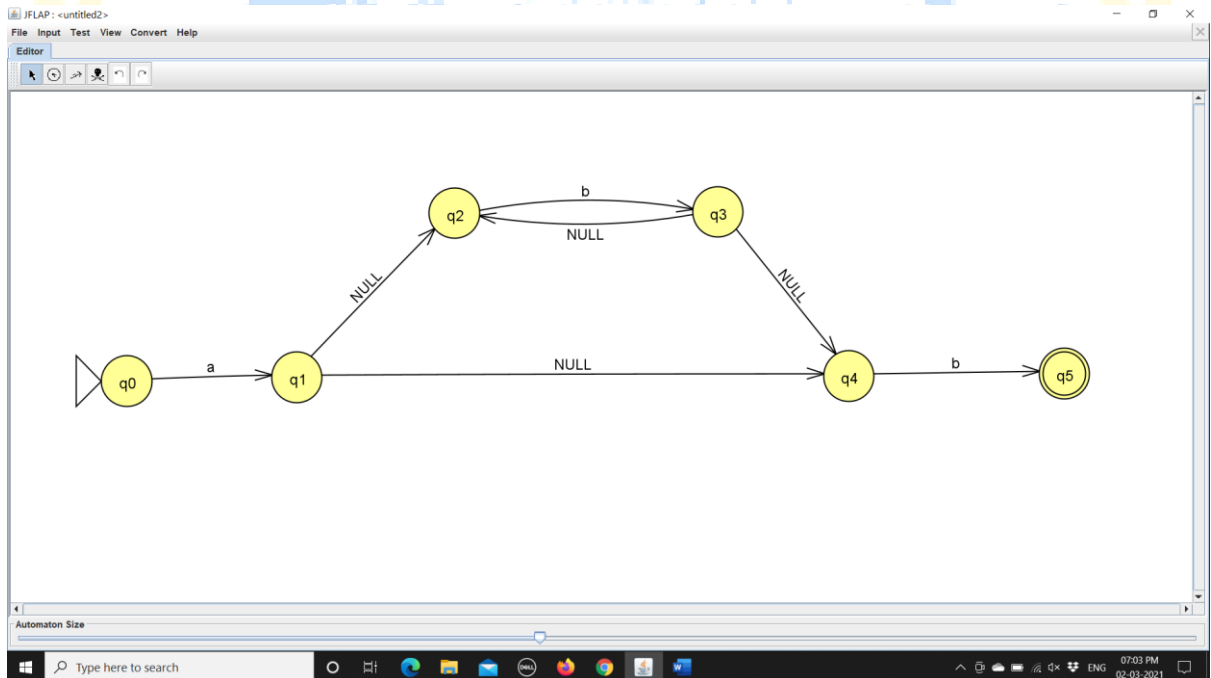




5.  $a^*b^*$



6.  $ab^*b$



**Result:** We converted the given Regular expression to NFA.

## Regular Expression to NFA to DFA

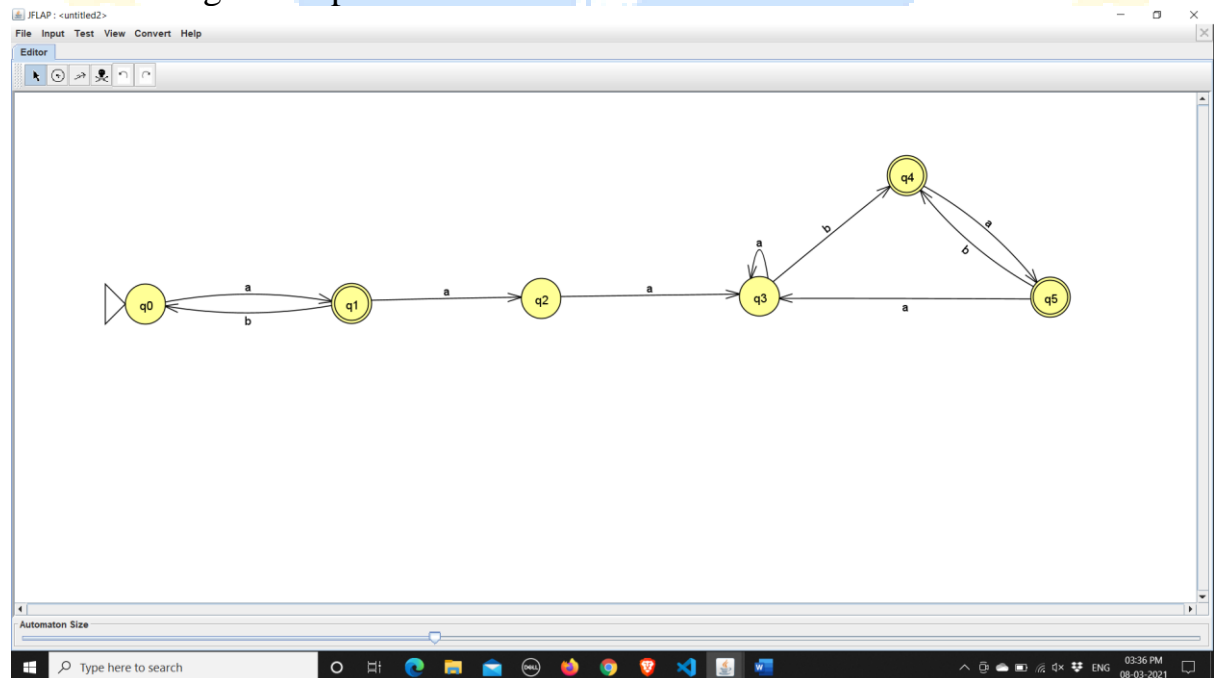
**Ques:**  $(a+b) * a(a+b) *$

The screenshot displays the FLAP (Finite Language Automata Package) software interface. The main window shows a state transition diagram with 17 states, labeled q0 through q16. State q0 is the start state, indicated by a double circle, and state q16 is the final state, indicated by a double circle. The transitions are as follows:

- q0 to q1: NULL
- q0 to q7: NULL
- q1 to q2: NULL
- q1 to q3: NULL
- q2 to q4: a
- q3 to q5: b
- q4 to q6: NULL
- q5 to q6: NULL
- q6 to q7: NULL
- q7 to q8: a
- q8 to q9: NULL
- q9 to q10: NULL
- q9 to q16: NULL
- q10 to q11: NULL
- q10 to q12: NULL
- q11 to q13: a
- q12 to q14: b
- q13 to q15: NULL
- q14 to q15: NULL
- q15 to q16: NULL

The interface includes a menu bar (File, Input, Test, View, Convert, Help), an Editor toolbar with icons for file operations and editing, and a status bar at the bottom showing 'Automation Size'.

DFA for the given expression is:



10

## **EXPERIMENT 4**

### **Computation of FIRST in a grammar**

**Aim:** Write a program in C/C++ to find the FIRST 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  $\epsilon$  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  $\epsilon$  Then
    - i. Add Result = Result U first element
    - ii. Flag=0
2. End Loop
3. If Flag = 0 Then
  - a. Print Result array of  $X_j$  which contain FIRST( $X_j$ )
4. End

#### **Program:**

```
#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[i]==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";
    }
}

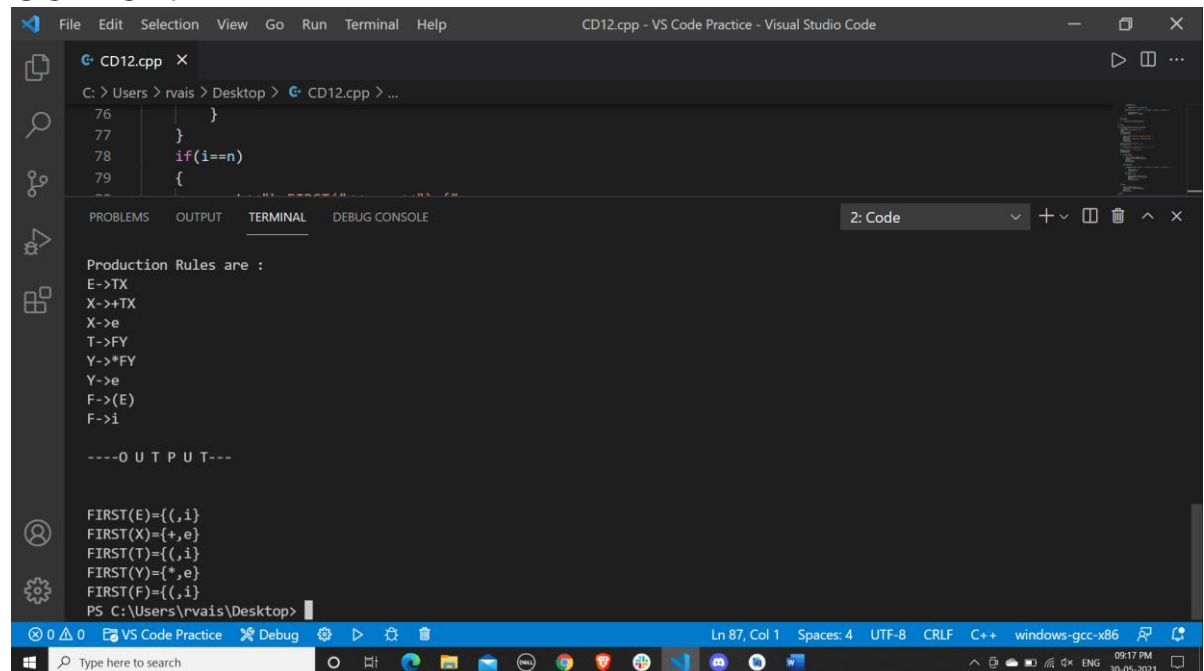
```

```

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: -



```
File Edit Selection View Go Run Terminal Help
CD12.cpp - VS Code Practice - Visual Studio Code

C: > Users \rvais > Desktop > CD12.cpp > ...

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

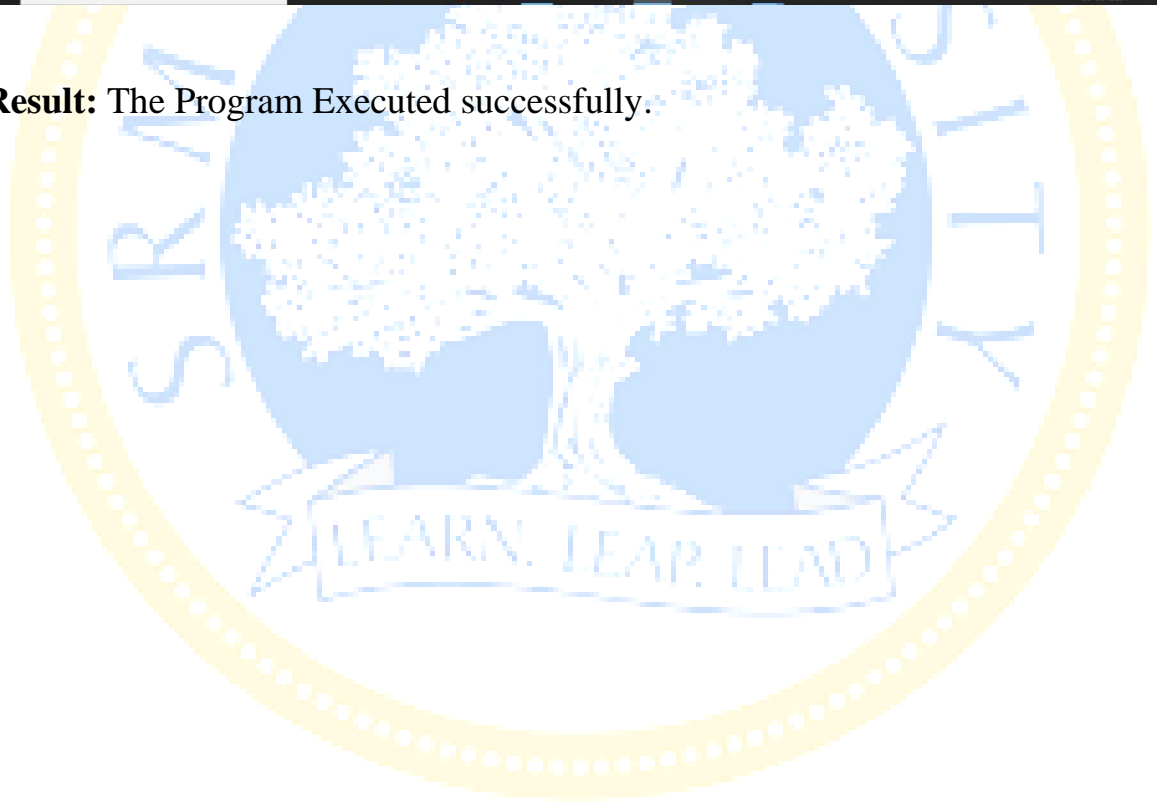
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
2: Code

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.



## EXPERIMENT 5

### Computation of FOLLOW in a grammar

**Aim:** Write a program in C/C++ to find a FOLLOW set from a given set of production rule.

**Algorithm:**

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  $FOLLOW(B) = FIRST(b)$ .
- END.

**Program:**

```
#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:");
```

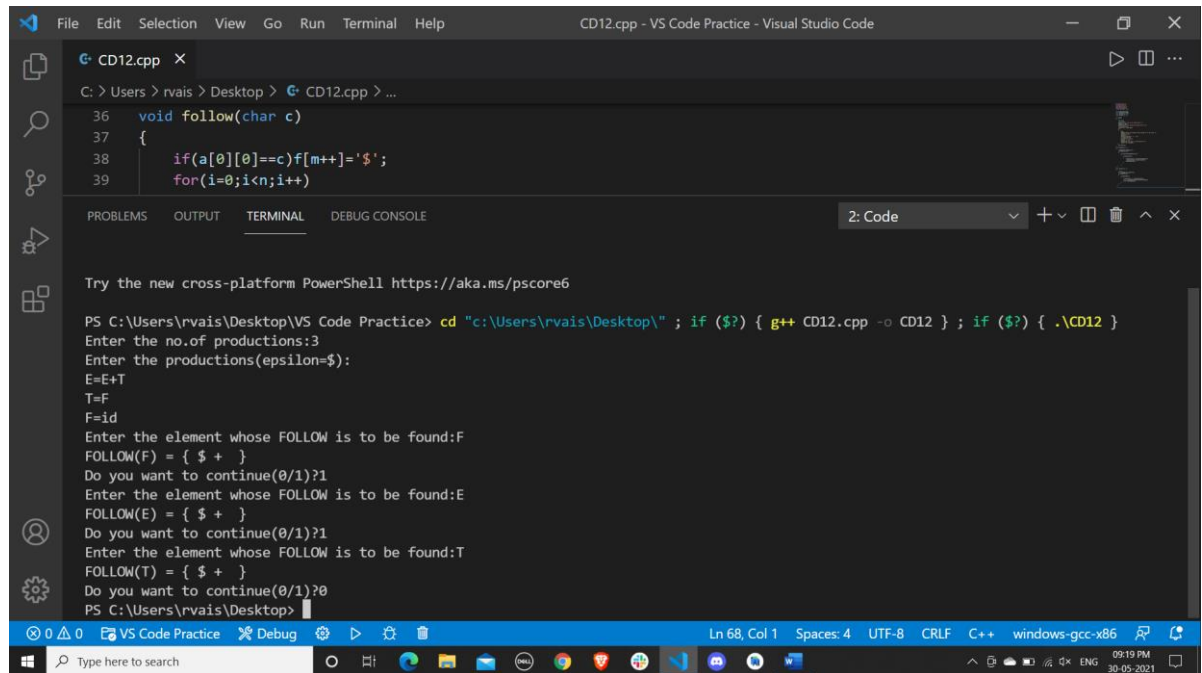
```

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: -



```
CD12.cpp x
C: > Users \rvais > Desktop > CD12.cpp > ...

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

### Computation of Predictive Parsing

**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++)
```

```

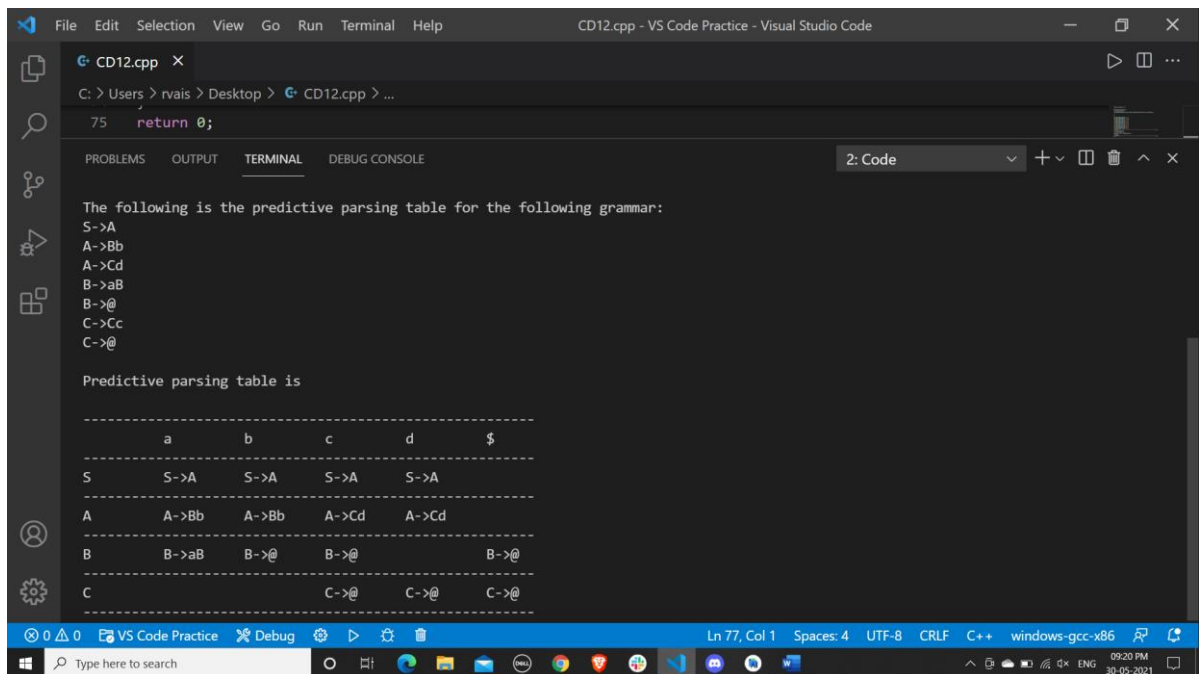
printf ("%s\n",prod[i]);
printf ("\nPredictive parsing table is\n");
fflush (stdin);
for (i=0; i<7; i++)
{
k=strlen(first[i]);
for (j=0; j<10; j++)
if(first[i][j] !='@')
strcpy(table[numr(prol[i][0])+1][numr(first[i][j])+1],prod[i]);
}

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:



```
File Edit Selection View Go Run Terminal Help
CD12.cpp - VS Code Practice - Visual Studio Code
C:\Users\rvais\Desktop> G++ CD12.cpp > ...
75 return 0;
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 2: Code

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

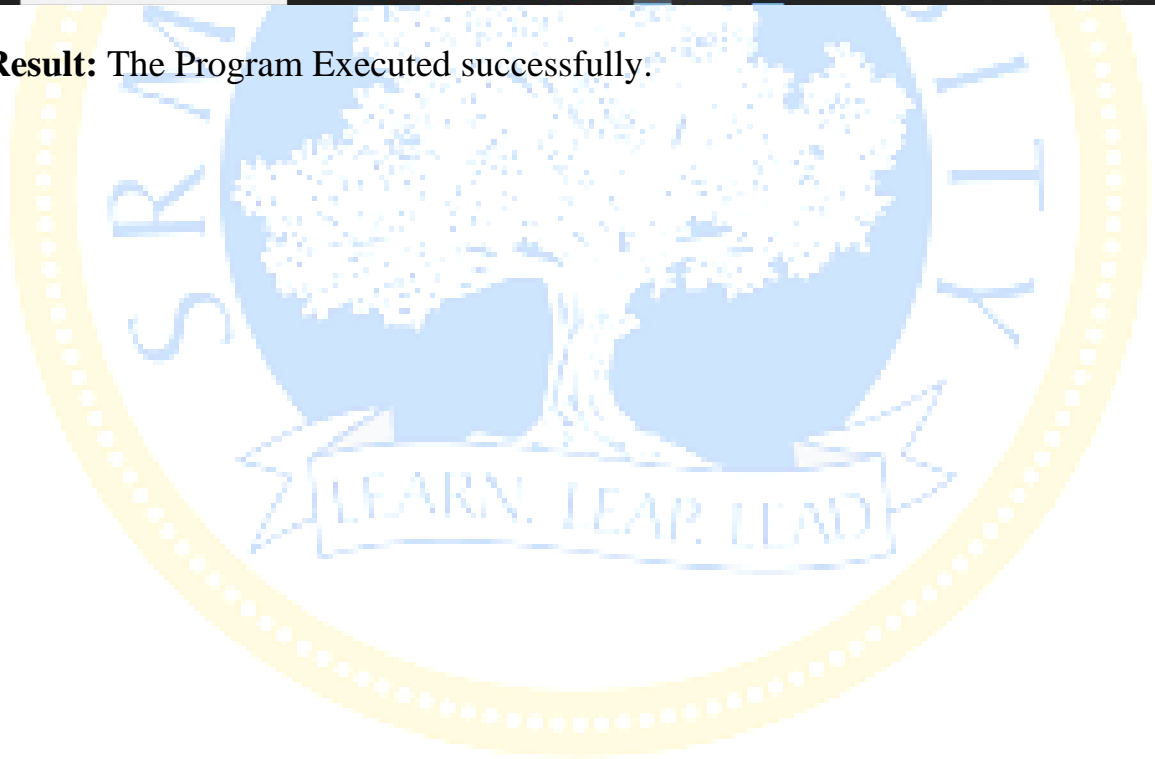
S->A  
A->Bb  
A->Cd  
B->aB  
B->@  
C->Cc  
C->@

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->@

VS Code Practice Debug Ln 77, Col 1 Spaces: 4 UTF-8 CRLF C++ windows-gcc-x86 09:20 PM 30-05-2021

**Result:** The Program Executed successfully.



## **EXPERIMENT 7**

### **Computation of 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$\t%symbols",stk,a,act);
        check();
    }
}
}
void check()
{
    strcpy(ac,"REDUCE TO E");
    for(z=0; z<c; z++)
        if(stk[z]=='i' && stk[z+1]=='d')
        {
            stk[z]='E';
            stk[z+1]='\0';
            printf("\n$%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:

```

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 ->E2 \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';

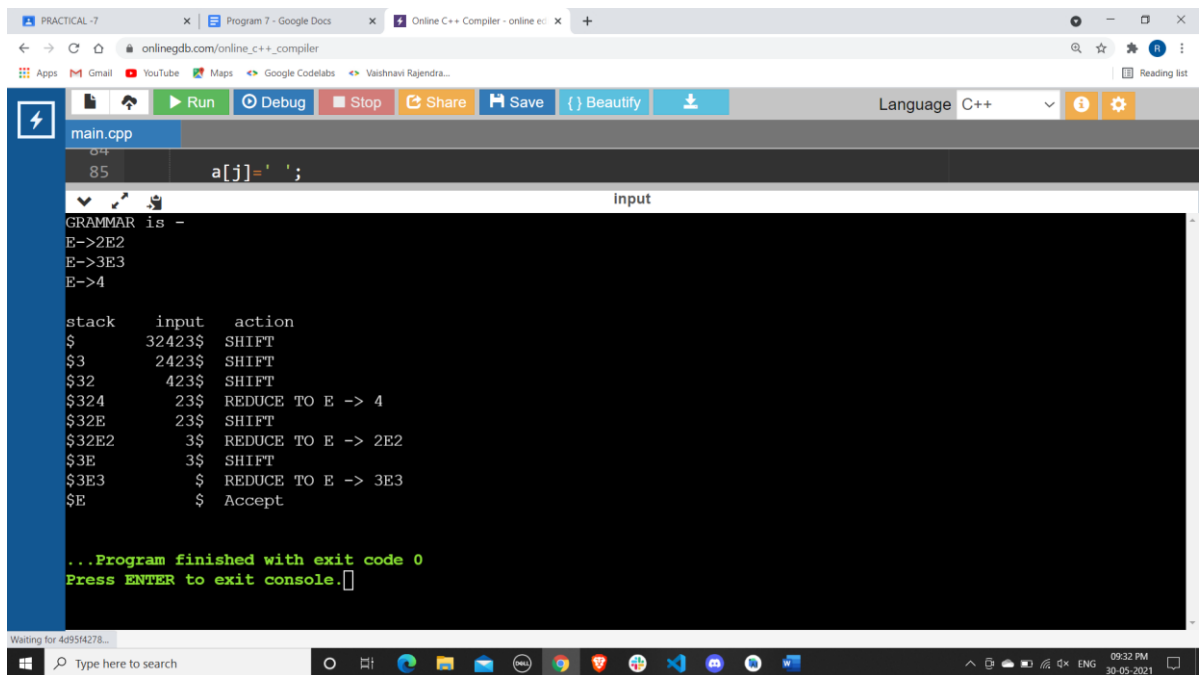
        a[j] = ' ';

        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:



```
main.cpp
85      a[j]=' ';

input
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])==NT[i])
                    {
                        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();
}

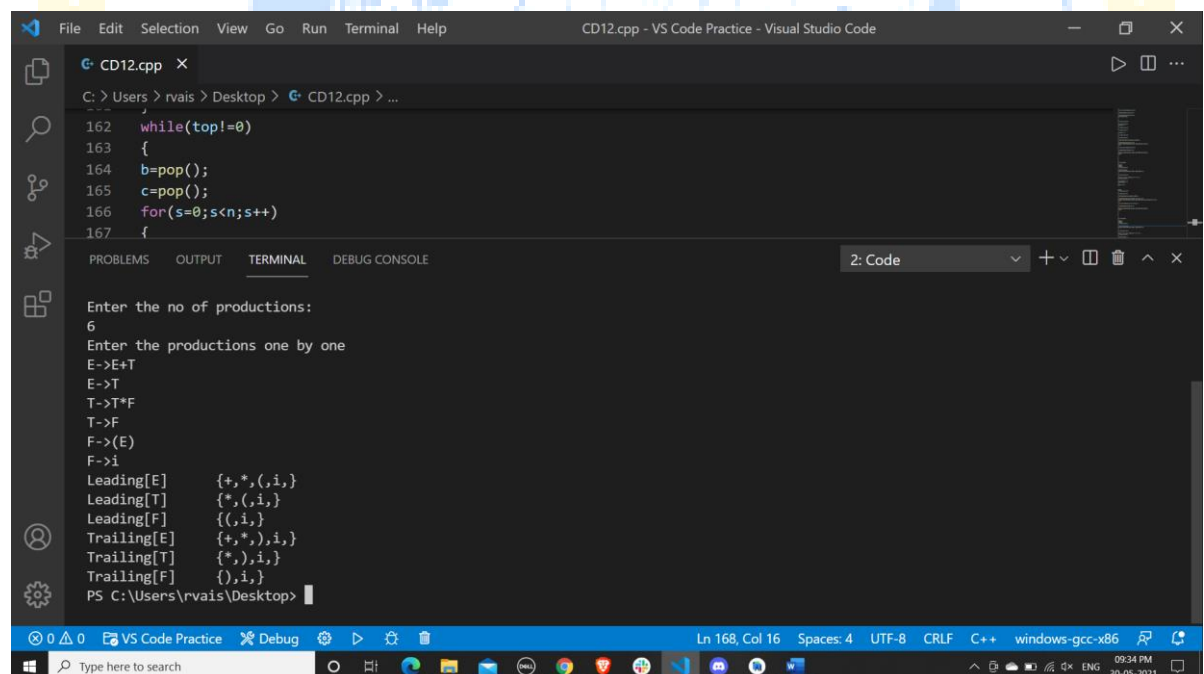
```

```

        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:**



```

C:\Users\rvais\Desktop> G++ CD12.cpp > ...
162 while(top!=0)
163 {
164     b=pop();
165     c=pop();
166     for(s=0;s<n;s++)
167     {
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 9

### Implementation of 3-Address Code using Quadruple

**Aim:** Write a program to implement 3-Address Code using Quadruple.

**Program:**

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

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

```
int main()
```

$$\{$$

```
char line[20];
```

```
int s[20];
```

```
int t=1;
```

```
int i=0;
```

```
printf("Enter String ");
```

```
gets(line);
```

```
for(i=0;i<20;i++)
```

```
s[i]=0;
```

```
printf("op\ta1\ta2\tres\n");
```

```
for(i=2;line[i]!='\0';i++)
```

{

```
if(line[i]=='/' || line[i]=='*')
```

{

```
printf("\n");
```

```
if(s[i]==0)
```

{

```
if(s[i+1]==0)
```

{

```
printf(":=\t%c\t\t t%d\n",line[i+1],t);
```

```
s[i+1]=t++;
```

}

```
printf("%c\t",line[i]);
```

```
(s[i-1]==0)?printf("%c\t",line[i-1]):printf("t%d\t",s[i-1]);
```

```
printf("t%d \t t%d",s[i+1],t);
```

```
s[i-1]=s[i+1]=t++;
```

```
s[i]=1;
```

}

}

}

```
for(i=2;line[i]!='\0';i++)
```

 $\{$





## **EXPERIMENT 10**

### **Intermediate Code Generation**

**Aim:** Write a program in C/C++ to generate intermediate code from a given syntax tree statement.

**Algorithm:**

1. Start the process.
2. Input an expression EXP from user.
3. Process the expression from right hand side to left hand side.
4. FLAG:=0; TOP = -1;
5. IF EXP = '=' then
  - i. IF EXP(index - 1) = 0 then
    1. PRINT EXP element from index to (index - 1) and POP STACK[TOP]. Terminate
  - Else
    - i. PRINT Wrong Expression
- [EndIF]
- IF an operator is found and FLAG = 0 then
  - i. TOP:= TOP + 1
  - ii. add to STACK[TOP].
  - iii. FLAG:=1
- Else
  - i. pop twice the STACK and result add to the newID(identifier) and PRINT.
  - ii. TOP:=TOP-2. Save newID to STACK[TOP]
  - iii. FLAG:=0
- [EndIF]
6. IF an operand is found then
  - i. TOP:=TOP+1
  - ii. move to STACK [TOP]
  - iii. IF TOP > 1 then
    1. pop twice the STACK and result add to the newID(identifier) and PRINT.
    2. TOP:=TOP-2. Save newID to STACK[TOP]
    3. FLAG:=0
- [End]
7. End the process

**Program (icgen.cpp):**

```
/* Intermediate Code Generator */
// Here consideration is any input expression
// only contain digits at the end
#include<iostream>
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>
using namespace std;

int main()
{
    char g,exp[20],stack[20];
    int m=0,i,top=-1,flag=0,len,j;
    cout<<"\nInput an expression : ";
    gets(exp);
    cout<<"\nIntermediate code generator\n";
    len=strlen(exp);

    //If expression contain digits
    if(isdigit(exp[len-1]))
    {
        cout<<"T = inttoreal(";
        i=len-1;
        while(isdigit(exp[i]))
        {
            i--;
        }
        for(j=i+1;j<len;j++)
        {
            cout<<exp[j];
        }
        cout<<".0)\n";
        exp[i+1]='T';len=i+2;
    }
    else //If expression having no digit
    {
        cout<<"T = "<<exp[len-1]<<"\n";
        exp[len-1]='T';
    }
    for(i=len-1;i>=0;i--)
    {
```

```

if(exp[i]=='=')
{
    if((i-1)==0)
    {
        // If expression contains unary operator in RHS near = operator
        if(isalpha(stack[top]))
        {
            cout<<exp[i-1]<<" "<<exp[i]<<" "<<stack[top];
        }
        else
        {
            cout<<exp[i-1]<<" "<<exp[i]<<" "<<stack[top]<<stack[top-1];
        }
        break;
    }
    else
    {
        cout<<"\nWrong Expression !!!";
        break;
    }
}
if(exp[i]=='+'||exp[i]=='/'||exp[i]=='*'||exp[i]=='-'||exp[i]=='%')
{
    if(flag==0)
    {
        flag=1;top=top+1;
        stack[top]=exp[i];
    }
    else
    {
        g=char('A' + m);m++;
        cout<<g<<" = "<<stack[top]<<stack[top-1]<<"\n";
        stack[top-1]=g;
        stack[top]=exp[i];
        flag=0;
    }
}
}
else
{
    top=top+1;
    stack[top]=exp[i];
    if(top>1)
    {

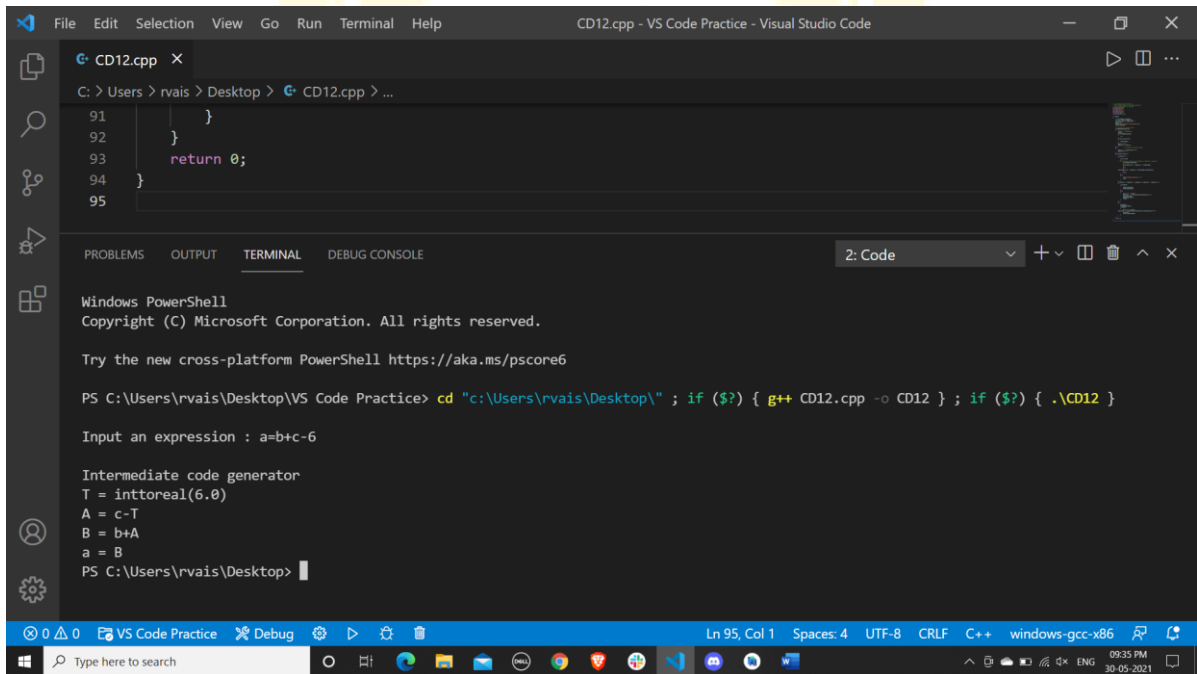
```

```

        g=char('A' + m);m++;
        cout<<g<<" = "<<stack[top]<<stack[top-1]<<stack[top-2]<<"\n";
        top=top-2;
        stack[top]=g;flag=0;
    }
}
}
return 0;
}

```

## OUTPUT:



```

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

Input an expression : a=b+c-6

Intermediate code generator
T = inttoreal(6.0)
A = c-T
B = b+A
a = B
PS C:\Users\rvais\Desktop>

```

**Result:** The Program Executed successfully

## **EXPERIMENT 11**

### **Intermediate code generation - Postfix expression**

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

**Program:**

```
#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.

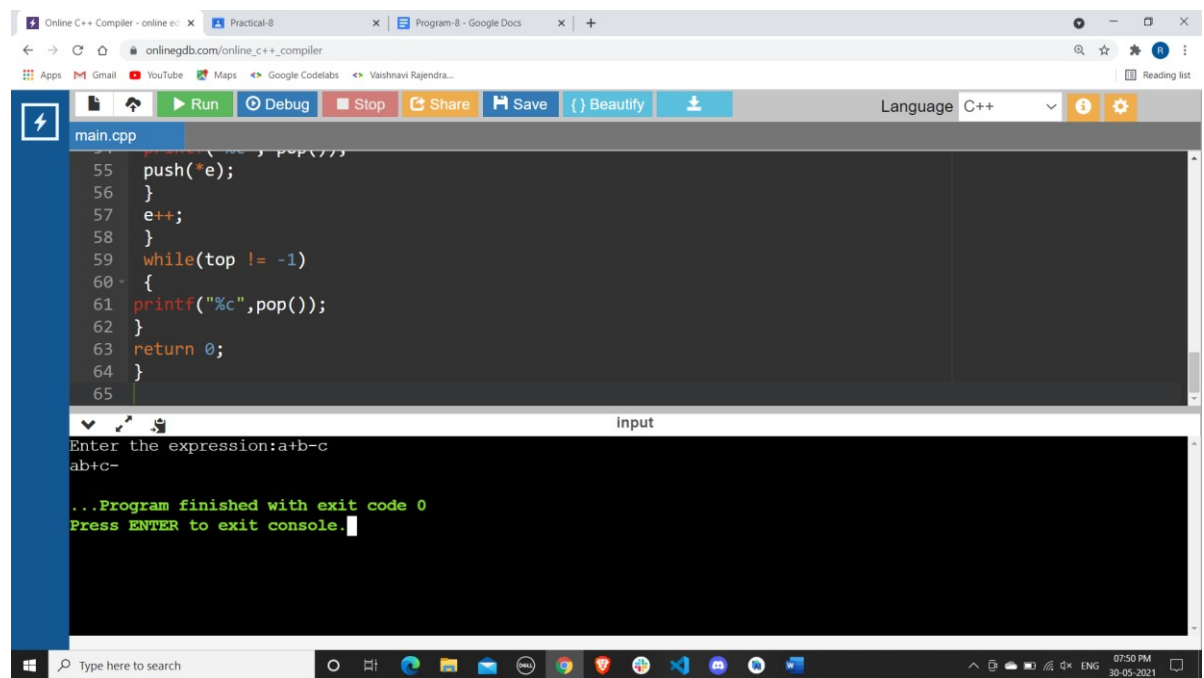
            while(priority(stack[top]) >= priority(*e) )
                printf("%c", pop());
            push(*e);
        }
        e++;
    }
    while(top != -1)
    {
        printf("%c",pop());
    }

    return 0;

}

```

## OUTPUT:



The screenshot displays a web browser window with the URL `onlinegdb.com/online_c++_compiler`. The browser tabs include 'Online C++ Compiler - online c++', 'Practical-8', and 'Program-8 - Google Docs'. The compiler interface features a top toolbar with buttons for Run, Debug, Stop, Share, Save, and Beautify, along with a language dropdown set to C++. The main editor shows a C++ program in `main.cpp` with line numbers 55 to 65. The code implements a stack using an array `e` and a `top` pointer. It includes a `pop()` function and a `main` function that pushes the expression 'a+b-c' onto the stack and then prints the top element. Below the editor, the 'Input' section shows the expression 'a+b-c' entered. The output console at the bottom displays the message '...Program finished with exit code 0' and 'Press ENTER to exit console.', indicating successful execution. The Windows taskbar at the bottom shows the time as 07:50 PM on 30-05-2021.

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

Input

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

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

**Result:** The Program Executed successfully



## **EXPERIMENT 12**

### **Intermediate code generation - Prefix Expression**

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

**Program:**

```
#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 infix[50],prfx[50],ch,elem;
    int i=0,k=0;
    printf("\n\nRead the Infix Expression : ");
    scanf("%s",infix);
    push('#');
    strrev(infix);

    while( (ch=infix[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);
                }
            }

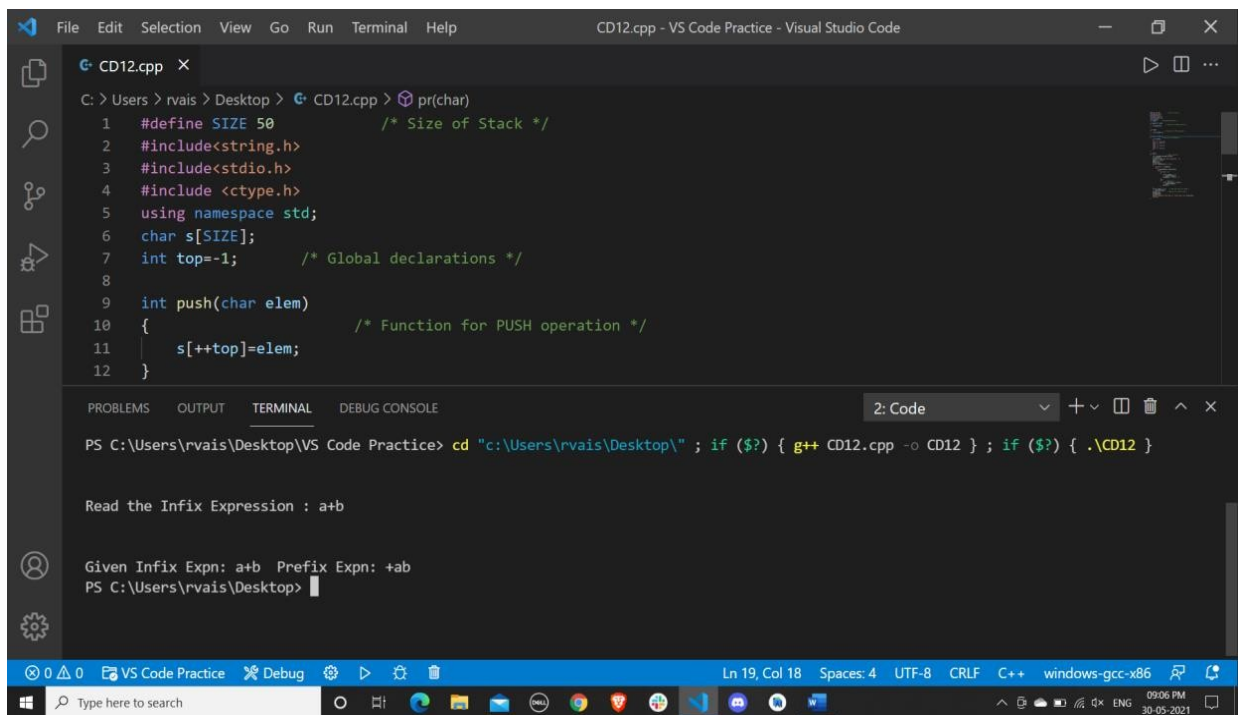
    while( s[top] != '#') /* Pop from stack till empty */
        prfx[k++]=pop();
    prfx[k]='\0'; /* Make prfx as valid string */

    strrev(prfx);
    strrev(infix);
    printf("\n\nGiven Infix Expn: %s Prefix Expn: %s\n",infix,prfx);

    return 0;
}

```

## OUTPUT:



```
CD12.cpp X
C:\Users\rvais\Desktop> cd CD12.cpp & pr(char)
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.

## **EXPERIMENT 13**

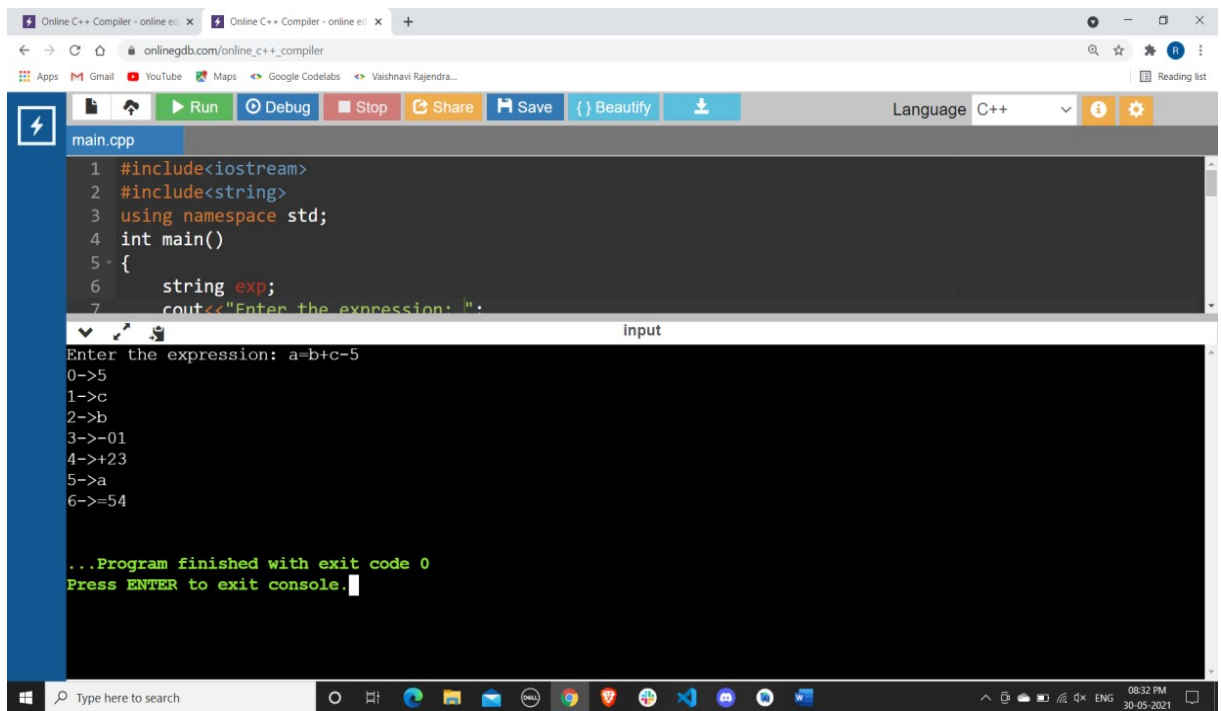
### **Construction of DAG**

**Aim:** Write a c or c++ or java to Construct 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:



The screenshot shows a web browser window with the URL `onlinegdb.com/online_c++_compiler`. The browser has several tabs open, including "Online C++ Compiler - online e...", "Online C++ Compiler - online e...", and "Vaishnavi Rajendra...". The browser's address bar shows the URL. The browser's toolbar includes buttons for "Run", "Debug", "Stop", "Share", "Save", "Beautify", and "Download". The language is set to "C++".

The code editor shows the following C++ code in `main.cpp`:

```
1 #include<iostream>
2 #include<string>
3 using namespace std;
4 int main()
5 {
6     string exp;
7     cout<<"Enter the expression: ";
```

The input field shows the following input:

```
Enter the expression: a=b+c-5
0->5
1->c
2->b
3->-01
4->+23
5->a
6->=54
```

The output field shows the following output:

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

**Result:** The Program Executed successfully.

## **EXPERIMENT 14**

### **Recursive Descent Parsing**

**Aim:** Write a program in C/ C++ or Java to implement Recursive Descent Parsing.

**Program:**

```
#include<iostream>
#include<map>
#include<vector>
using namespace std;
int main()
{
    int flag = 0;
    map<char,vector<string> >rules;
    string exp,test;
    rules['S'].push_back("aAc");
    rules['A'].push_back("cd");
    rules['A'].push_back("d");
    cout<<"Enter the string: ";
    cin>>exp;
    string start="aAc";
    if(start[0]!=exp[0])
        cout<<"Not Accepted";
    else
    {
        cout<<"S"<<endl<<start<<endl;
        string a= (rules['A'])[0];          string b=(rules['A'])[1];
        string t;
        t=start[0]+a+start[2];
        cout<<t<<endl; if(t==exp)
        {
            flag = 1;
            cout<<"Accepted";
        }
        else
        {
            cout<<start<<endl;
            t=start[0]+b+start[2];
            cout<<t<<endl;
        }
    }
}
```

```

        if(t==exp)
        {
            flag = 1;
            cout<<"Accepted";
        }
    }
    if(flag == 0) cout<<"Not accepted";
    return 0;
}

```

## OUTPUT:

The screenshot shows an online C++ compiler interface. The code in the editor is as follows:

```

19 //Function to check if a string is a palindrome or not
20 //Function to check if a string is a palindrome or not
21 string a= (rules['A'])[0];    string b=(rules['A'])[1];
22 string t;
23 t=start[0]+a+start[2];
24 cout<<t<<endl;
25 if(t==exp)
26 {

```

The input entered is "adc". The output shown is:

```

Enter the string: adc
S
aAc
acdc
aAc
adc
Accepted
...Program finished with exit code 0
Press ENTER to exit console.

```

**Result:** The Program Executed successfully