**COMPILER CONSTRUCTION (CS-310)**

**SUBMITTED BY**

**MUNIB-UL-HASSAN**

**ROLL NO #**

**CS19-037**

****

**SUBMITTED TO**

**SIR RAHEEL**

**DEPARTMENT OF COMPUTER SCIENCE**

**SIR SYED UNIVERSITY OF ENGINEERING AND TECHNIOLOGY**

**LAB # 01**

# Task 01:

Make a program which recognizes the key strokes as you press different letters. Example: if you press letter A then it displays the output as “Letter A is pressed “After pressing the 10 letters, it counts the # number of occurrences of the letters.

# CODING:

using System;

using System.Collections.Generic; using System.Linq;

using System.Text;

using System.Threading.Tasks; namespace std

{

public class C\_freq

{

public int i;

public int[] freq = new int[256]; public void cal\_freq(char[] str1, int n)

{

for (i = 0; i<n ; i++)

{

freq[str1[i]]++;

}

}

public void display(char[] str1, int n)

{

for (int i = 0; i < 256; i++)

{

if (freq[i] != 0)

{

Console.WriteLine("The frequency of " + (char)i + " is " + freq[i] + "\n");

}

}

}

}

class Program

{

static void Main(string[] args)

{

char[] str1 = new char[10]; char ch;

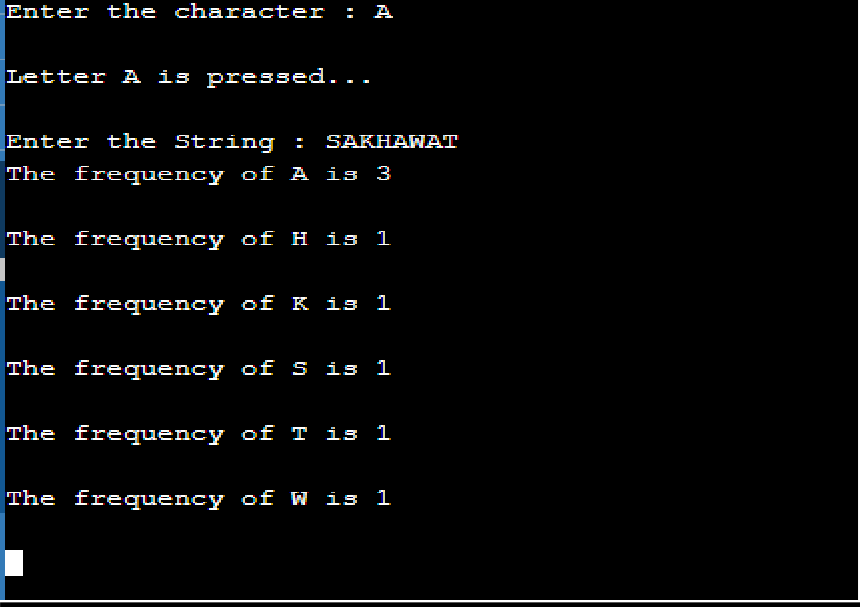
string str2; int n;

Console.Write("Enter the character : "); ch = Console.ReadLine()[0];

Console.WriteLine("\nLetter " + ch + " is pressed...\n"); Console.Write("Enter the String : ");

str2 = Console.ReadLine(); str1 = str2.ToCharArray(); n = str1.Length;

C\_freq f = new C\_freq(); f.cal\_freq(str1,n); f.display(str1,n); Console.ReadLine();}}}



OUTPUT:

**Task 01:**

**LAB # 02**

* Saving & opening a given text in a file.
* Searching for a given string in a file
* Replace the searched string with the given string

# CODING:

using System; using System.IO;

namespace CompilerLab2

{ class Program

{

static void Main(string[] args)

{

string path = @"D:\LAB\Dexter.txt";

using (StreamWriter sw = File.CreateText(path))

{ sw.WriteLine("Hello Dexter !"); sw.WriteLine("How Are You"); sw.WriteLine("Where Are You From"); sw.Close();

}

Console.WriteLine("Kindly check the file"); Console.ReadKey();

string[] lines = File.ReadAllLines(path); for (int i = 0; i < lines.Length; i++)

{

if (lines[i] == "Hello Dexter !")

{

lines[i] = "Hello David !"; }

}

using (StreamWriter sw = File.CreateText(path))

{

for (int i = 0; i < lines.Length; i++)

{

sw.WriteLine(lines[i]);

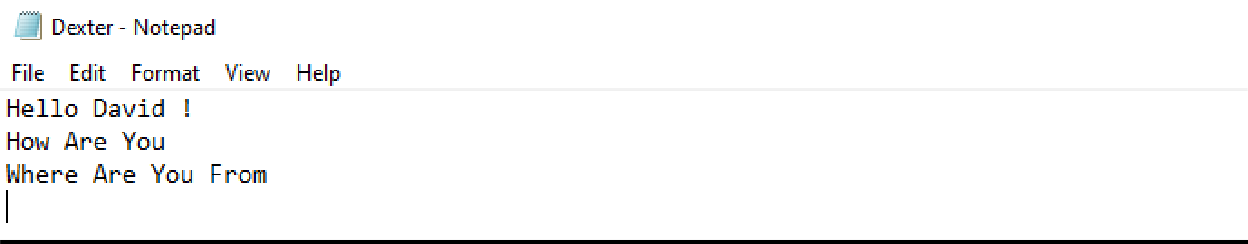
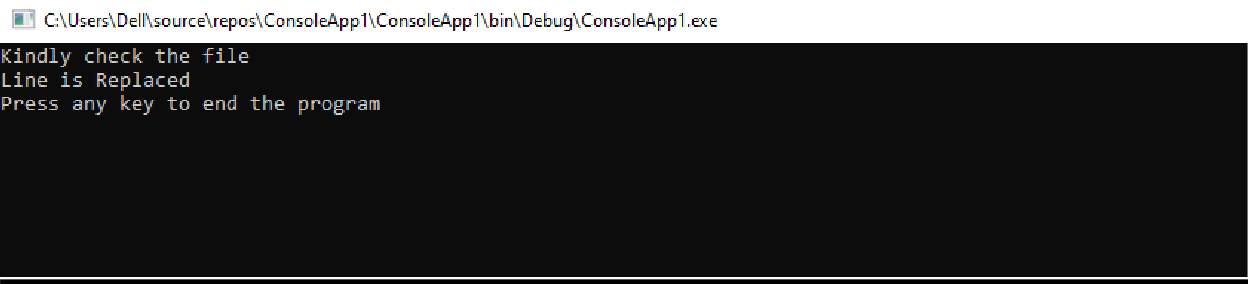
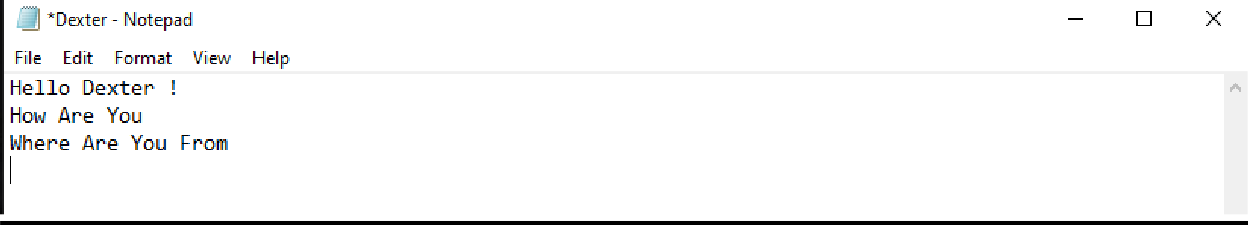
}

sw.Close();

}

Console.WriteLine("Line is Replaced"); Console.WriteLine("Press any key to end the program"); Console.ReadKey(); } }}

# OUTPUT:



**LAB # 03**

**Task 01:**

**Write a lexical analyzer in any language for the given tokens.**

**CODING:**

using System;

using System.Collections.Generic; using System.Linq;

using System.Text;

namespace ConsoleApplication4

{

class Program

{

static void Main(string[] args)

{

string input, c; do

{

Console.WriteLine(" Enter any R.E"); input = Console.ReadLine();

if (input == "ws")

{

Console.WriteLine(" Atribute is -"); Console.WriteLine(" Token is -");

}

else if (input == "if")

{

Console.WriteLine(" Attribute is -"); Console.WriteLine(" Token is if");

}

else if (input == "then")

{

Console.WriteLine(" Attribute is -"); Console.WriteLine(" Token is then");

}

else if (input == "else")

{

Console.WriteLine(" Attribute is -"); Console.WriteLine(" Token is else");

}

else if (input == "num")

{

Console.WriteLine(" Attribute is pointer to table entry"); Console.WriteLine(" Token is num");

}

else if (input == "id")

{

Console.WriteLine(" Attribute is pointer to table entry"); Console.WriteLine(" Token is id");

}

else if (input == "<")

{

Console.WriteLine(" Attribute is LT"); Console.WriteLine(" Token is relop");

}

else if (input == "<=")

{

Console.WriteLine(" Attribute is LE"); Console.WriteLine(" Token is relop");

}

else if (input == "=")

{

Console.WriteLine(" Attribute is EQ"); Console.WriteLine(" Token is relop");

}

else if (input == "<>")

{

Console.WriteLine(" Attribute is NE"); Console.WriteLine(" Token is relop");

}

else if (input == ">")

{

Console.WriteLine(" Attribute is GT"); Console.WriteLine(" Token is relop");

}

else if (input == ">=")

{

Console.WriteLine(" Attribute is GE"); Console.WriteLine(" Token is relop");

}

else

{

Console.WriteLine(" Expression is invalid \n");

}

Console.WriteLine(" Do you want to continue y/n : "); c = Console.ReadLine();

}

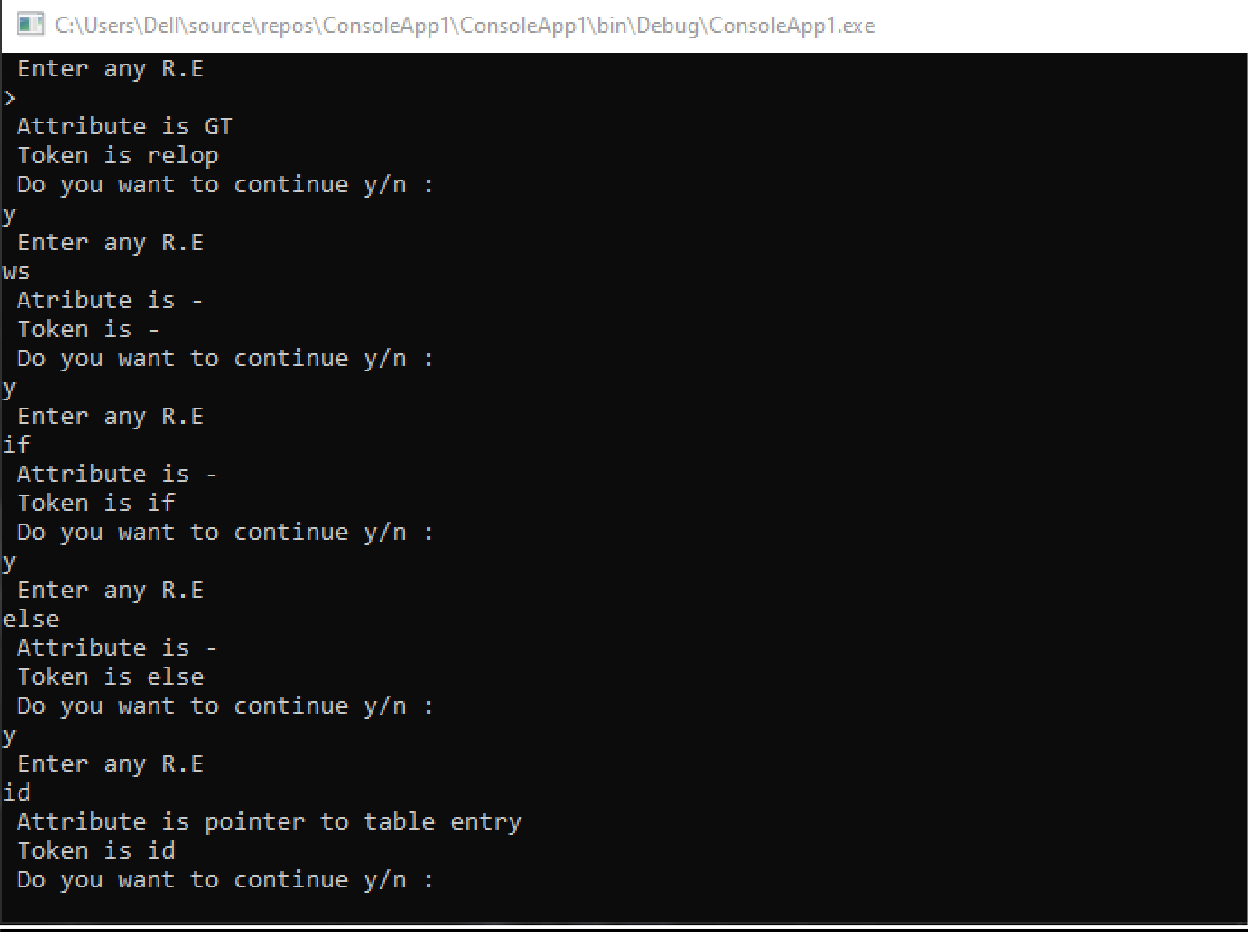
while (c != "no"); Console.ReadLine();

}

}

}

# OUTPUT:



**LAB # 04**

**Task 01:**

**Input Buffering Technique- I CODING:**

using System;

using System.Collections.Generic; using System.Linq;

using System.Text;

namespace ConsoleApplication4

{

class Program

{

static void Main(string[] args)

{

string[] token = { "do", "int", "float", "double","string", "char"}; string input;

Console.WriteLine("Enter character: "); input = Console.ReadLine();

for (int i = 0; i <= input.Length - 1; i++)

{

Char[] buffer;

if (input == token[i])

{

for (int j = 0; j <= input.Length - 1; j++)

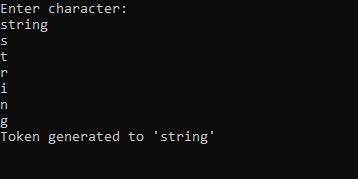
{

Console.WriteLine(buffer[j]);

}

Console.WriteLine("Token generated to '" + token[i] + "'");

}

}

Console.ReadLine();

}

}

# OUTPUT:

**LAB # 05 (INPUT BUFFERING)**

**CODING:**

using System;

using System.Collections.Generic; using System.Linq;

using System.Text;

namespace buffertechniquetwo

{

class Program

{

static void Main(string[] args)

{

string a;

int len, div, x = 0, y = 5, z = 0, i = 5; int choice = 0;

while (true)

{

Console.WriteLine("Input the string:"); a = Console.ReadLine();

len = a.Length; div = len / 5;

Console.WriteLine("\n"); string[] sub = new string[15]; while (div > z)

{

sub[z] = a.Substring(x, y); Console.WriteLine(sub[z]); x = x + i;

z++;

}

Console.WriteLine("\nPress 1 to exit!"); Console.WriteLine("Press 2 to continue!"); choice = int.Parse(Console.ReadLine());

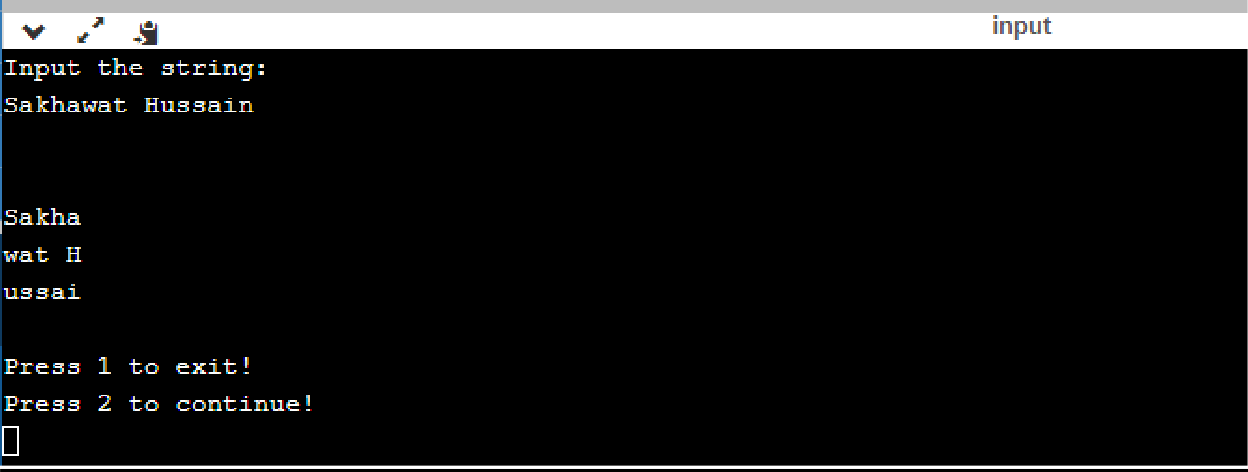
if (choice == 1)

{

Environment.Exit(2);

}}}}}

# OUTPUT:



**LAB # 06**

**Task 01:**

(a | b) (ba | ab) \*

Construct the transition diagram for the above regular expression and implement it in any conventional programming language.

# CODING:

using System;

using System.Collections.Generic; using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab\_Ccstr

{

class Program

{

static void Main(string[] args)

{

string a, ch; int i = 1, len;

bool flag = false;

do

{

Console.Write("\nEnter a String : "); a = Console.ReadLine();

char[] x = a.ToCharArray(); len = x.Length;

if (x[0] == 'a' || x[0] == 'b')

{

if (len == 1 && len % 2 != 0)

{

Console.WriteLine("String is Correct"); flag = true;

}

while (i < len && len % 2 != 0)

{

if (x[i] == 'a' && x[i + 1] == 'b' || x[i] == 'b' && x[i + 1] == 'a')

{

i += 2;

if (i >= len)

{

Console.WriteLine("String is Correct"); flag = true;

}

}

}

}

if (flag == false)

{

Console.WriteLine("String is not Correct!");

}

flag = false;

Console.WriteLine("\nEnter Yes IF You Want Try Again Or No To Quit : "); ch = Console.ReadLine().ToUpper();

}

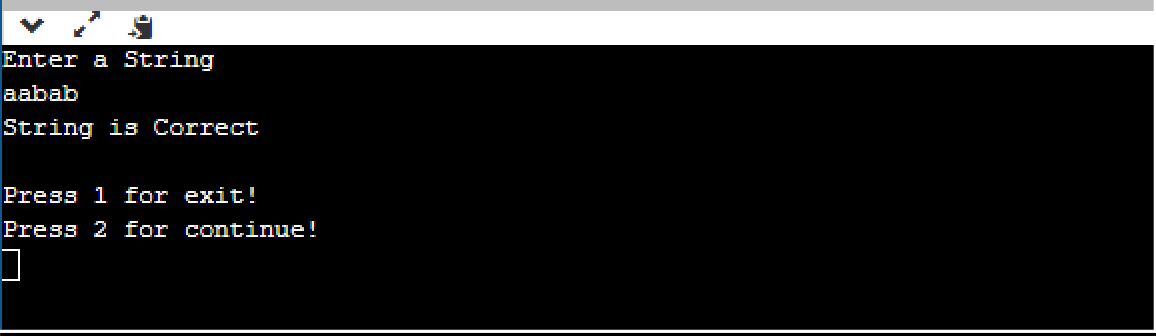
while (ch == "YES"); Console.ReadLine();

}

}

}

# OUTPUT:



**LAB # 07**

**Task 01:**

RECOGNITION OF TOKENS :

# CODING:

using System;

using System.Collections.Generic; using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab\_Ccstr

{

class Program

{

static void Main(string[] args)

{

string a, ch; do

{

Console.Write("\nEnter the string : "); a = Console.ReadLine();

char[] exp = a.ToCharArray(); int len = a.Length;

if (exp[0] == '<')

{

if (len == 2)

{

if (exp[1] == '=')

{

LE\nState: 2", a);

}

Console.WriteLine("Expression: {0} \nToken: relop \nAttribute Value:

else if (exp[1] == '>')

{

Console.WriteLine("Expression: {0} \nToken: relop \nAttribute Value: NE\nState: 3", a);

}

}

if (len == 1)

Console.WriteLine("Expression: {0} \nToken: relop \nAttribute Value: LT\nState: 4", a);

}

else if (exp[0] == '=')

{

5", a);

Console.WriteLine("Expression: {0} \nToken: relop \nAttribute Value: EQ\nState:

}

else if (exp[0] == '>')

{

if (len == 2)

{

if (exp[1] == '=')

{

Console.WriteLine("Expression: {0} \nToken: relop \nAttribute Value: GE\nState: 7", a);

}

}

if (len == 1)

Console.WriteLine("Expression: {0} \nToken: relop \nAttribute Value: GT\nState: 8", a);

}

else

{

Console.WriteLine("Invalid...!!!");

}

Console.WriteLine("\nEnter Yes IF You Want Try Again Or No To Quit : "); ch = Console.ReadLine().ToUpper();

}

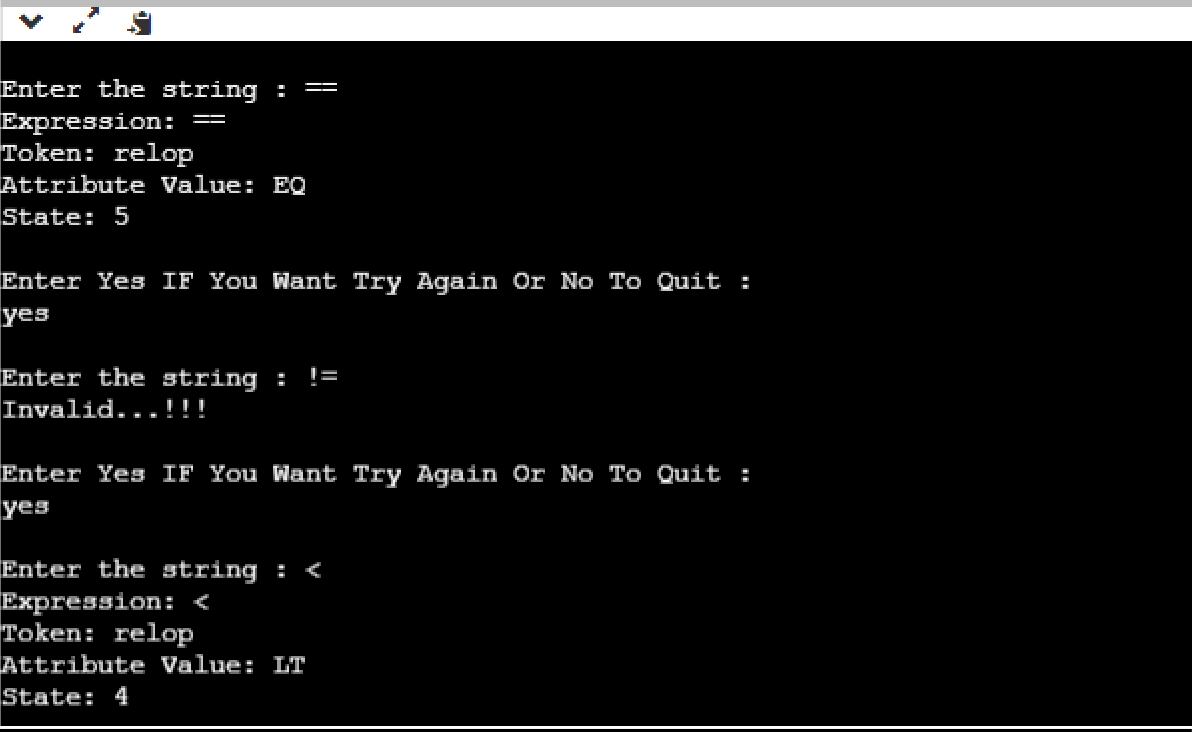
while (ch == "YES"); Console.ReadLine();

}

}

}

# OUTPUT:



**LAB # 08**

**Task 01:**

RECOGNITION OF IDENTIFIERS :

# CODING:

using System;

using System.Collections.Generic; using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab\_Ccstr

{

class Program

{

static void Main(string[] args)

{

int len;

string exp, ch;

char[] digit = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9' };

do

{

Console.Write("Enter A String : "); exp = Console.ReadLine();

char[] exp1 = exp.ToCharArray(); len = exp1.Length;

if (exp1[0] >= digit[0] && exp1[0] <= digit[9])

{

Console.WriteLine("Invalid Identifier...!!!");

}

else

{

Installid()", exp);

}

Console.WriteLine("\nString: {0} \nToken: GetToken() \nAttribute Value:

Console.WriteLine("\nEnter Yes IF You Want Try Again Or No To Quit : "); ch = Console.ReadLine().ToUpper();

}

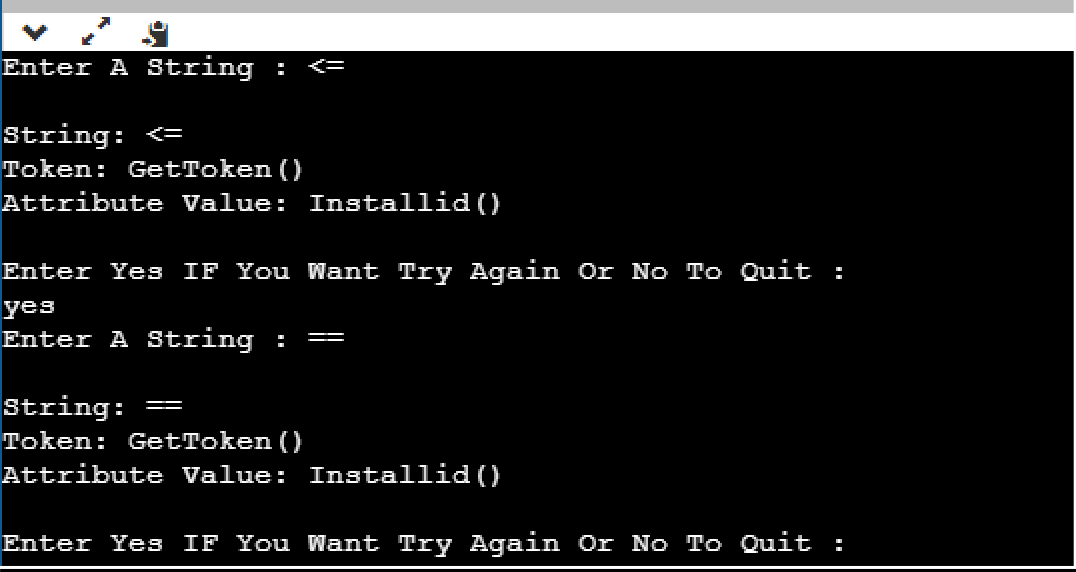
while (ch == "YES"); Console.ReadLine();

}

}

}

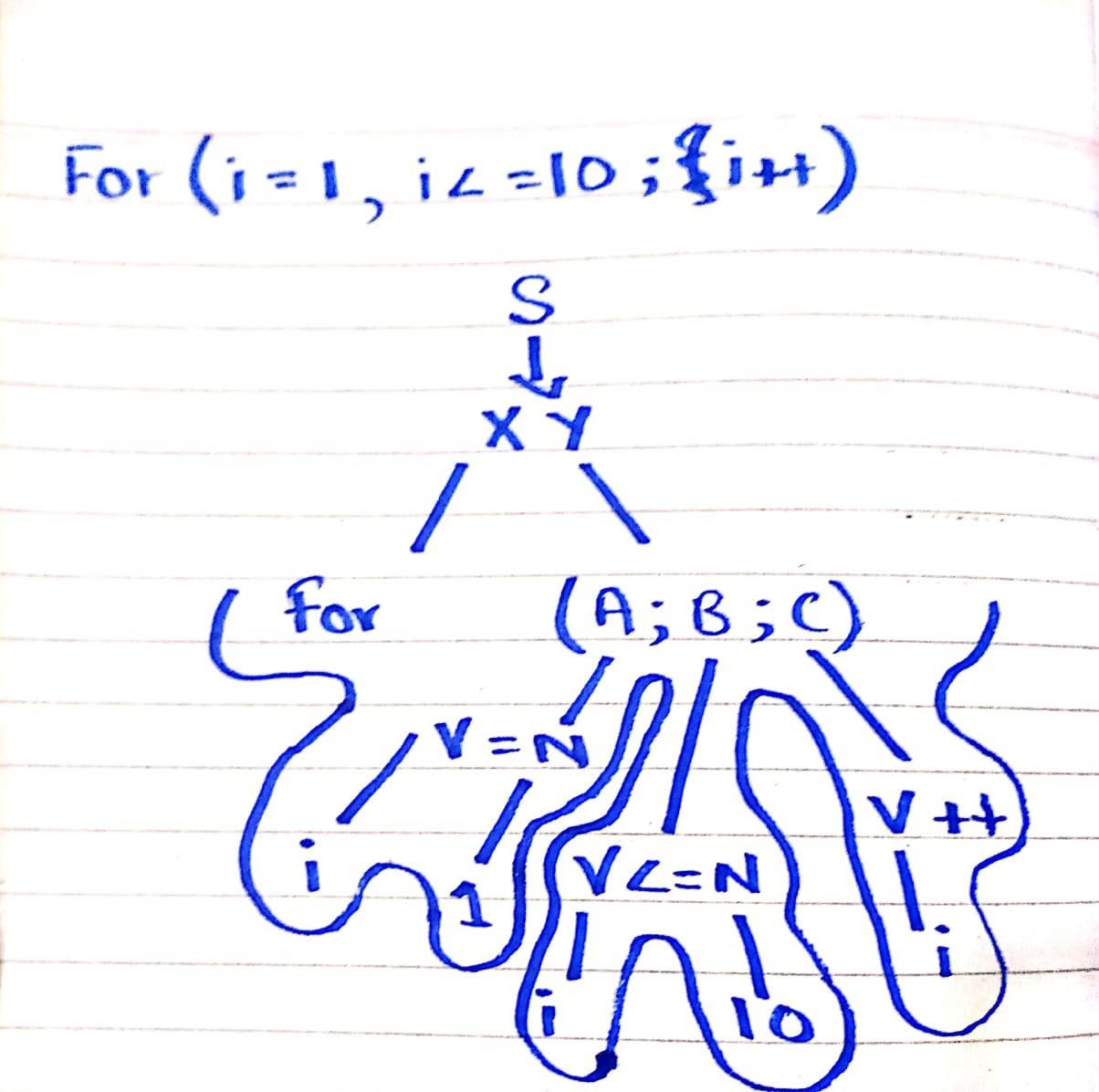
# OUTPUT:



**Task 01:**

**LAB # 09**

**OUTPUT:**



**LAB # 10**

**Task 01:**

LEXICAL ANALYZER:

# CODING:

#include<iostream> #include<fstream> #include<stdlib.h> #include<string.h> #include<ctype.h>

using namespace std;

int isKeyword(char buffer[]){

char keywords[32][10] = {"auto","break","case","char","const","continue","default",

"do","double","else","enum","extern","float","for","goto",

"if","int","long","register","return","short","signed",

"sizeof","static","struct","switch","typedef","union",

"unsigned","void","volatile","while"}; int i, flag = 0;

for(i = 0; i < 32; ++i){ if(strcmp(keywords[i], buffer) == 0){ flag = 1;

break;

}

}

return flag;

}

int main(){

char ch, buffer[15], operators[] = "+-\*/%="; ifstream fin("program.txt");

int i,j=0; if(!fin.is\_open()){

cout<<"error while opening the file\n"; exit(0);

}

while(!fin.eof()){ ch = fin.get();

for(i = 0; i < 6; ++i){ if(ch == operators[i])

cout<<ch<<" is operator\n";

}

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

}

else if((ch == ' ' || ch == '\n') && (j != 0)){ buffer[j] = '\0';

j = 0;

if(isKeyword(buffer) == 1) cout<<buffer<<" is keyword\n"; else

cout<<buffer<<" is indentifier\n";

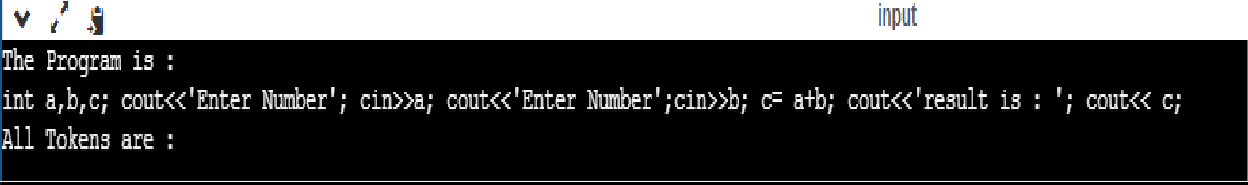
}

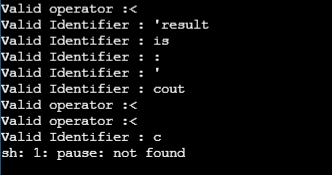
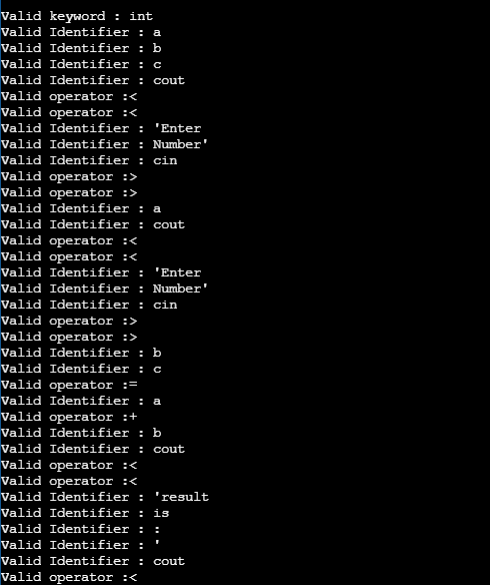
}

fin.close(); return 0;

}

# OUTPUT:





**Task 01:**

**LAB # 11**

LEFT RECURSION AND LEFT FACTORING:

# CODING:

#include<iostream> #include<stdio.h> #include<conio.h> #include<string>

using namespace std; int main()

{ string ip,op1,op2,temp; int sizes[10] = {};

char c; int n,j,l;

cout<<"Enter the Parent Non-Terminal : "; cin>>c;

ip.push\_back(c); op1 += ip + "\'->";

ip += "->";

op2+=ip;

cout<<"Enter the number of productions : "; cin>>n;

for(int i=0;i<n;i++)

{ cout<<"Enter Production "<<i+1<<" : "; cin>>temp;

sizes[i] = temp.size(); ip+=temp;

if(i!=n-1)

ip += "|";

}

cout<<"Production Rule : "<<ip<<endl; for(int i=0,k=3;i<n;i++)

{

if(ip[0] == ip[k])

{

cout<<"Production "<<i+1<<" has left recursion."<<endl; if(ip[k] != '#')

{

for(l=k+1;l<k+sizes[i];l++) op1.push\_back(ip[l]);

k=l+1; op1.push\_back(ip[0]);

op1 += "\'|";

}

}

else

{

cout<<"Production "<<i+1<<" does not have left recursion."<<endl; if(ip[k] != '#')

{

for(j=k;j<k+sizes[i];j++) op2.push\_back(ip[j]);

k=j+1; op2.push\_back(ip[0]); op2 += "\'|";

}

else

{

op2.push\_back(ip[0]); op2 += "\'";

}}}

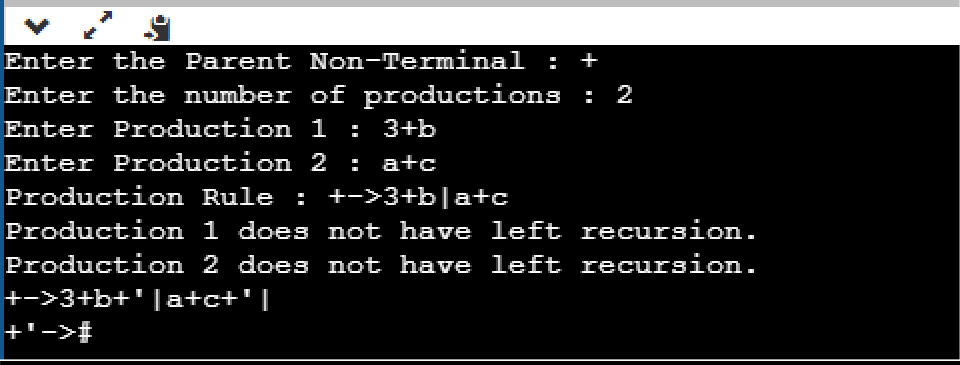
op1 += "#";

cout<<op2<<endl; cout<<op1<<endl; getch();

return 0;

}

# OUTPUT:



**Task 01:**

**LAB # 12**

LEFT RECURSION AND LEFT FACTORING:

# CODING:

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

void followfirst(char, int, int); void follow(char c);

void findfirst(char, int, int); int count, n = 0;

char calc\_first[10][100];

char calc\_follow[10][100]; int m = 0;

char production[10][10], f[10], first[10]; int k, e;

char ck;

int main(int argc, char \*\*argv)

{

int jm = 0 , km = 0, i, choice, kay , ptr = -1; char c, ch;

count = 8;

strcpy(production[0], "E=TR"); strcpy(production[1], "R=+TR"); strcpy(production[2], "R=#"); strcpy(production[3], "T=FS"); strcpy(production[4], "S=\*FS"); strcpy(production[5], "S=#"); strcpy(production[6], "F=(E)"); strcpy(production[7], "F=i");

char done[count];

for(k = 0; k < count; k++) { for(kay = 0; kay < 100; kay++) {

calc\_first[k][kay] = '!';

}

}

int point1 = 0, point2, xxx;

for(k = 0; k < count; k++)

{

c = production[k][0]; point2 = 0;

xxx = 0;

for(kay = 0; kay <= ptr; kay++) if(c == done[kay])

xxx = 1;

if (xxx == 1) continue;

findfirst(c, 0, 0);

ptr += 1;

done[ptr] = c;

printf("\n First(%c) = { ", c); calc\_first[point1][point2++] = c;

for(i = 0 + jm; i < n; i++) { int lark = 0, chk = 0;

for(lark = 0; lark < point2; lark++) {

if (first[i] == calc\_first[point1][lark])

{

chk = 1; break;

}

}

if(chk == 0)

{

printf("%c, ", first[i]); calc\_first[point1][point2++] = first[i];

}

}

printf("}\n"); jm = n; point1++;

}

printf("\n");

printf(" \n\n"); char donee[count];

ptr = -1;

for(k = 0; k < count; k++) { for(kay = 0; kay < 100; kay++) {

calc\_follow[k][kay] = '!';

}

}

point1 = 0; int land = 0;

for(e = 0; e < count; e++)

{

ck = production[e][0]; point2 = 0;

xxx = 0;

for(kay = 0; kay <= ptr; kay++) if(ck == donee[kay])

xxx = 1;

if (xxx == 1) continue;

land += 1;

follow(ck); ptr += 1;

donee[ptr] = ck;

printf(" Follow(%c) = { ", ck); calc\_follow[point1][point2++] = ck;

for(i = 0 + km; i < m; i++) { int lark = 0, chk = 0;

for(lark = 0; lark < point2; lark++)

{

if (f[i] == calc\_follow[point1][lark])

{

chk = 1; break;

}

}

if(chk == 0)

{

printf("%c, ", f[i]); calc\_follow[point1][point2++] = f[i];

}

}

printf(" }\n\n"); km = m; point1++;

}

}

void follow(char c)

{

int i, j;

if(production[0][0] == c) { f[m++] = '$';

}

for(i = 0; i < 10; i++)

{

for(j = 2;j < 10; j++)

{

if(production[i][j] == c)

{

if(production[i][j+1] != '\0')

{

followfirst(production[i][j+1], i, (j+2));

}

if(production[i][j+1]=='\0' && c!=production[i][0])

{

follow(production[i][0]);

}} } }}

void findfirst(char c, int q1, int q2)

{

int j; if(!(isupper(c))) {

first[n++] = c;

}

for(j = 0; j < count; j++)

{

if(production[j][0] == c)

{

if(production[j][2] == '#')

{

if(production[q1][q2] == '\0') first[n++] = '#';

else if(production[q1][q2] != '\0' && (q1 != 0 || q2 != 0))

{

findfirst(production[q1][q2], q1, (q2+1));

}

else

first[n++] = '#';

}

else if(!isupper(production[j][2]))

{

first[n++] = production[j][2];

}

else

{

findfirst(production[j][2], j, 3);

}}}}

void followfirst(char c, int c1, int c2)

{

int k;

if(!(isupper(c))) f[m++] = c;

else

{

int i = 0, j = 1;

for(i = 0; i < count; i++)

{

if(calc\_first[i][0] == c) break;

}

while(calc\_first[i][j] != '!')

{

if(calc\_first[i][j] != '#')

{

f[m++] = calc\_first[i][j];

}

else

{

if(production[c1][c2] == '\0')

{

follow(production[c1][0]);

}

else

{

followfirst(production[c1][c2], c1, c2+1);

}

}

j++;

}

}

}

}

# OUTPUT:

