Term work

of

Compiler Design Lab (PCS-601)

Submitted in partial fulfillment of the requirement for the VI semester of

Bachelor of Technology (Computer Science & Engineering)

By

Shreya Dungrakoti

2171156

Under the Guidance of

Mr. Anubhav Bewerwal

Assistant Professor

Department of Computer Science & Engineering

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS

SATTAL ROAD, P.O. BHOWALI

DISTRICT- NAINITAL-263132

2023-2024

CERTIFICATE

The term work of Compiler Design Lab (PCS-601), being submitted by Shreya Dungrakoti, Student Roll no 2171156 **to Graphic Era Hill University, Bhimtal Campus is abonafide work carried out by her. She has worked under my guidance and supervision and fulfilled the requirement for the submission of this lab file.**

(…………………) **(……………………)**

Faculty Incharge **HOD, Dept. of CSE**

ACKNOWLEDGEMENT

I take immense pleasure in thanking **Mr. Anubhav Bewerwal** (Assistant Professor, Dept. of CSE, GEHU, Bhimtal Campus) for allowing me to carry out this lab work under his excellent and optimistic supervision. This has all been possible due to his novel inspiration, able guidance and useful suggestions that have helped me in developing my subject concepts as a student.

I want to extend thanks to our President, Honorable **Prof. (Dr.) Kamal Ghanshala** for providing us all infrastructure and facilities to work in need without which this work would not be possible.

(Shreya Dungrakoti)

shreyadungrakoti@gmail.com

STUDENT’S DECLARATION

I, Shreya Dungrakoti hereby declare the work, which is being presented in the report, entitled **Term work** of **Compiler Design Lab (PCS-601)** in partial fulfillment of the requirement for the award of the degree **Bachelor of Technology (Computer Science& Engineering)** in the session **2023-2024** for semester VI, is an authentic record of my own work carried out under the supervision of **Mr. Anubhav Bewerwal,** Dept. of CSE (Graphic Era Hill University, Bhimtal Campus).

The matter embodied in this project has not been submitted by me for the award of any other degree.

Date: …………   ……………….

(Full signature of student)

Department of Computer Science and Engineering
COMPILER DESIGN LAB (PCS-601)

Requirements: Windows/Linux based Computer System

Index/List of Practicals

Sl. No.	Name of Practicals	Date	Page No.	Teacher's Remark & Signature
1	Write a program in C or C++ language for the following functions without using string.h header file: a: "to get the length of a string, you use the strlen() function" b: "To concatenate (combine) two strings, you can use the strcat() function c: "To copy the value of one string to another, you can use the strcpy()" d: "To compare two strings, you can use the strcmp() function." and other related functions.	17-02-2024		
2	Write a program in C or C++ language to generate tokens as identifiers, keywords, newline, tabs, whitespaces and characters.	24-02-2024		
3	Write a C or C++ program to convert NFA to its equivalent DFA.	06-03-2024		
4	Write a C or C++ program to convert RE to its equivalent NFA.	13-03-2024		
5	Write a Lex program to generate tokens as identifiers, keywords, newline, tabs, whitespaces and characters.	20-03-2024		
6	Write a program in C or C++ language to implement Predictive Parsing Algorithm.	01-05-2024		
7	Write a program in C or C++ language to find the FIRST and FOLLOW of all the variables. Create functions for FIRST and FOLLOW.	08-05-2024		
8	Write a program in C or C++ language to implement LR Parser.	15-05-2024		
9	Write a program in C or C++ to generate the three-address code.	22-05-2024		
10	Write a program in C or C++ to generate machine code from the abstract syntax tree generated by the parser.	29-05-2024		

1. Write a program in C or C++ language for the following functions without using string.h header file:
a: "to get the length of a string, you use the strlen() function"
b: "To concatenate (combine) two strings, you can use the strcat() function
c: "To copy the value of one string to another, you can use the strcpy()"
d: "To compare two strings, you can use the strcmp() function."
and other related functions.

PROGRAM:

a->

```
#include <stdio.h>

int  my_strlen(const char* str)
 {
   size_t len = 0;
   while (*str != '\0') {
      len++;
      str++;
   }
   return len;
}
int main() {
   char str[] = "C Snippets";
   int len = my_strlen(str);
   printf("Length of the string is %d\n", len);
   return 0;

}
```

OUTPUT:

```
Length of the string is 10.
```

b->

```
#include <stdio.h>
void my_strcat(char[],char[]);
int main()
{
   char S1[100], S2[100];
   printf("Enter the String-1: ");
   gets(S1);
   printf("Enter the String-2: ");
   gets(S2);
   my_strcat(S1,S2);
   printf("Concatenated String: ");
```

```
    puts(S1);
    return 0;
}
void my_strcat(char S1[], char S2[])
{
    int l=0, i;
    while(S1[l]!='\0')
    {
        l=l+1;
    }
    i=0;
    while(S2[i]!='\0')
    {
        S1[l] = S2[i];
        i=i+1;
        l=l+1;
    }
    S1[l] = '\0';
}
```

OUTPUT

```
Enter the String-1: Atharva
Enter the String-2:  Raj Sinha
Concatenated String: Atharva Raj Sinha
```

c->

```
#include <stdio.h>
void mystrcpy(char*,char*);
int main()
{
    char str[100], str2[100];
    printf("Enter the String: ");
    gets(str);
    mystrcpy(str2,str);
    printf("Copied String: ");
    puts(str2);
    return 0;
}
void mystrcpy(char *q, char *p)
{
    int i=0;
    while(*(p+i)!='\0')
    {
        *(q+i) = *(p+i);
        i=i+1;
    }
    *(q+i) = '\0';
}
```

OUTPUT:

```
Enter the String: Atharva Raj
Copied String: Atharva Raj
```

d->

```
#include <stdio.h>
int mystrcmp(char[],char[]);
int main()
{
    int c;
    char s1[100], s2[100];
    printf("Enter the String-1: ");
    gets(s1);
    printf("Enter the String-2: ");
    gets(s2);
    c = mystrcmp(s1,s2);
    if(c==0)
    {
        printf("Equal Strings\n");
    }
    else
    {
        printf("Unequal Strings\n");
    }
    return 0;
}
int mystrcmp(char s1[], char s2[])
{
    int i=0;
    while(s1[i]!='\0' || s2[i]!='\0')
    {
        if(s1[i] > s2[i])
        {
            return 1;
        }
        else if(s1[i] < s2[i])
        {
            return -1;
        }
        i=i+1;
    }
    return 0;
}
```

OUTPUT:

```
Enter the String-1: Atharva
Enter the String-2: atharva
Unequal Strings
Enter the String-1: Atharva
Enter the String-2: Atharva
Equal Strings
```

2. Write a program in C or C++ language to generate tokens as identifiers, keywords, newline, tabs, whitespaces and characters.

PROGRAM:
```
#include <stdbool.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

// Returns 'true' if the character is a DELIMITER.
bool isDelimiter(char ch)
{
   if (ch == ' ' || ch == '+' || ch == '-' || ch == '*' ||
      ch == '/' || ch == ',' || ch == ';' || ch == '>' ||
      ch == '<' || ch == '=' || ch == '(' || ch == ')' ||
      ch == '[' || ch == ']' || ch == '{' || ch == '}')
      return (true);
   return (false);
}

// Returns 'true' if the character is an OPERATOR.
bool isOperator(char ch)
{
   if (ch == '+' || ch == '-' || ch == '*' ||
      ch == '/' || ch == '>' || ch == '<' ||
      ch == '=')
      return (true);
   return (false);
}

// Returns 'true' if the string is a VALID IDENTIFIER.
bool validIdentifier(char* str)
{
    if (str[0] == '0' || str[0] == '1' || str[0] == '2' ||
       str[0] == '3' || str[0] == '4' || str[0] == '5' ||
      str[0] == '6' || str[0] == '7' || str[0] == '8' ||
      str[0] == '9' || isDelimiter(str[0]) == true)
      return (false);
   return (true);
}

// Returns 'true' if the string is a KEYWORD.
bool isKeyword(char* str)
{
   if (!strcmp(str, "if") || !strcmp(str, "else") ||
      !strcmp(str, "while") || !strcmp(str, "do") ||
      !strcmp(str, "break") ||
         !strcmp(str, "continue") || !strcmp(str, "int")
        || !strcmp(str, "double") || !strcmp(str, "float")
        || !strcmp(str, "return") || !strcmp(str, "char")
        || !strcmp(str, "case") || !strcmp(str, "char")
        || !strcmp(str, "sizeof") || !strcmp(str, "long")
        || !strcmp(str, "short") || !strcmp(str, "typedef")
        || !strcmp(str, "switch") || !strcmp(str, "unsigned")
        || !strcmp(str, "void") || !strcmp(str, "static")
        || !strcmp(str, "struct") || !strcmp(str, "goto"))
        return (true);
    return (false);
}

// Returns 'true' if the string is an INTEGER.
bool isInteger(char* str)
{
    int i, len = strlen(str);

    if (len == 0)
        return (false);
    for (i = 0; i < len; i++) {
        if (str[i] != '0' && str[i] != '1' && str[i] != '2'
            && str[i] != '3' && str[i] != '4' && str[i] != '5'
            && str[i] != '6' && str[i] != '7' && str[i] != '8'
            && str[i] != '9' || (str[i] == '-' && i > 0))
            return (false);
    }
    return (true);
}

// Returns 'true' if the string is a REAL NUMBER.
bool isRealNumber(char* str)
{
    int i, len = strlen(str);
    bool hasDecimal = false;

    if (len == 0)
        return (false);
    for (i = 0; i < len; i++) {
        if (str[i] != '0' && str[i] != '1' && str[i] != '2'
            && str[i] != '3' && str[i] != '4' && str[i] != '5'
            && str[i] != '6' && str[i] != '7' && str[i] != '8'
            && str[i] != '9' && str[i] != '.' ||
            (str[i] == '-' && i > 0))
            return (false);
        if (str[i] == '.')
            hasDecimal = true;
    }
    return (hasDecimal);
}

// Extracts the SUBSTRING.
char* subString(char* str, int left, int right)
{
    int i;
    char* subStr = (char*)malloc(
                sizeof(char) * (right - left + 2));

    for (i = left; i <= right; i++)
        subStr[i - left] = str[i];
    subStr[right - left + 1] = '\0';
    return (subStr);
}

// Parsing the input STRING.
void parse(char* str)
{
    int left = 0, right = 0;
    int len = strlen(str);

    while (right <= len && left <= right) {
        if (isDelimiter(str[right]) == false)
            right++;

        if (isDelimiter(str[right]) == true && left == right) {
            if (isOperator(str[right]) == true)
                printf("'%c' IS AN OPERATOR\n", str[right]);

            right++;
            left = right;
        } else if (isDelimiter(str[right]) == true && left != right
                   || (right == len && left != right)) {
            char* subStr = subString(str, left, right - 1);

            if (isKeyword(subStr) == true)
                printf("'%s' IS A KEYWORD\n", subStr);

            else if (isInteger(subStr) == true)
                printf("'%s' IS AN INTEGER\n", subStr);

            else if (isRealNumber(subStr) == true)
                printf("'%s' IS A REAL NUMBER\n", subStr);

            else if (validIdentifier(subStr) == true
                && isDelimiter(str[right - 1]) == false)
            printf("'%s' IS A VALID IDENTIFIER\n", subStr);

        else if (validIdentifier(subStr) == false
                && isDelimiter(str[right - 1]) == false)
            printf("'%s' IS NOT A VALID IDENTIFIER\n", subStr);
        left = right;
      }
    }
    return;
}

// DRIVER FUNCTION
int main()
{
     // maximum length of string is 100 here
    char str[100] = "int a = b + 1c; ";

    parse(str); // calling the parse function

    return (0);
}
```

OUTPUT:

```
'int' IS A KEYWORD
'a' IS A VALID IDENTIFIER
'=' IS AN OPERATOR
'b' IS A VALID IDENTIFIER
'+' IS AN OPERATOR
'1c' IS NOT A VALID IDENTIFIER
```

3. Write a C or C++ program to convert NFA to its equivalent DFA.

PROGRAM:

```
#include <stdio.h>
int main()
{
   int nfa[5][2];
   nfa[1][1]=12;
   nfa[1][2]=1;
   nfa[2][1]=0;
   nfa[2][2]=3;
   nfa[3][1]=0;
   nfa[3][2]=4;
   nfa[4][1]=0;
   nfa[4][2]=0;
   int dfa[10][2];
   int dstate[10];
   int i=1,n,j,k,flag=0,m,q,r;
   dstate[i++]=1;
   n=i;

   dfa[1][1]=nfa[1][1];
   dfa[1][2]=nfa[1][2];
   printf("\nf(%d,a)=%d",dstate[1],dfa[1][1]);
   printf("\nf(%d,b)=%d",dstate[1],dfa[1][2]);

 for(j=1;j<n;j++)
   {
      if(dfa[1][1]!=dstate[j])
        flag++;
   }
   if(flag==n-1)
   {
      dstate[i++]=dfa[1][1];
      n++;
   }
   flag=0;
   for(j=1;j<n;j++)
   {
       if(dfa[1][2]!=dstate[j])
          flag++;
   }
   if(flag==n-1)
   {
       dstate[i++]=dfa[1][2];
       n++;
   }
```

```
    k=2;
    while(dstate[k]!=0)
    {
        m=dstate[k];
        if(m>10)
        {
            q=m/10;
            r=m%10;
        }
        if(nfa[r][1]!=0)
             dfa[k][1]=nfa[q][1]*10+nfa[r][1];
        else
            dfa[k][1]=nfa[q][1];
        if(nfa[r][2]!=0)
            dfa[k][2]=nfa[q][2]*10+nfa[r][2];
        else
            dfa[k][2]=nfa[q][2];

        printf("\nf(%d,a)=%d",dstate[k],dfa[k][1]);
        printf("\nf(%d,b)=%d",dstate[k],dfa[k][2]);

       flag=0;
        for(j=1;j<n;j++)
        {
         if(dfa[k][1]!=dstate[j])
          flag++;
        }
      if(flag==n-1)
     {
        dstate[i++]=dfa[k][1];
        n++;
     }
    flag=0;
    for(j=1;j<n;j++)
    {
         if(dfa[k][2]!=dstate[j])
            flag++;
    }
    if(flag==n-1)
    {
         dstate[i++]=dfa[k][2];
         n++;
    }
    k++;
    }
    return 0;
}
```

OUTPUT:

```
f(1,a)=12
f(1,b)=1
f(12,a)=12
f(12,b)=13
f(13,a)=12
f(13,b)=14
f(14,a)=12
f(14,b)=1
```

4. Write a C or C++ program to convert RE to its equivalent NFA.

PROGRAM:

```
#include<stdio.h>
#include<string.h>
int main()
{
    char reg[20]; int q[20][3],i=0,j=1,len,a,b;
    for(a=0;a<20;a++) for(b=0;b<3;b++) q[a][b]=0;
    scanf("%s",reg);
    printf("Given regular expression: %s\n",reg);
    len=strlen(reg);
    while(i<len)
    {
        if(reg[i]=='a'&&reg[i+1]!='|'&&reg[i+1]!='*') { q[j][0]=j+1; j++; }
        if(reg[i]=='b'&&reg[i+1]!='|'&&reg[i+1]!='*') { q[j][1]=j+1; j++; }
        if(reg[i]=='e'&&reg[i+1]!='|'&&reg[i+1]!='*') { q[j][2]=j+1; j++; }
        if(reg[i]=='a'&&reg[i+1]=='|'&&reg[i+2]=='b')
        {
          q[j][2]=((j+1)*10)+(j+3); j++;
          q[j][0]=j+1; j++;
            q[j][2]=j+3; j++;
            q[j][1]=j+1; j++;
            q[j][2]=j+1; j++;
            i=i+2;
        }
        if(reg[i]=='b'&&reg[i+1]=='|'&&reg[i+2]=='a')
        {
            q[j][2]=((j+1)*10)+(j+3); j++;
            q[j][1]=j+1; j++;
            q[j][2]=j+3; j++;
            q[j][0]=j+1; j++;
            q[j][2]=j+1; j++;
            i=i+2;
        }
        if(reg[i]=='a'&&reg[i+1]=='*')
        {
            q[j][2]=((j+1)*10)+(j+3); j++;
            q[j][0]=j+1; j++;
            q[j][2]=((j+1)*10)+(j-1); j++;
        }
        if(reg[i]=='b'&&reg[i+1]=='*')
        {
            q[j][2]=((j+1)*10)+(j+3); j++;
            q[j][1]=j+1; j++;
            q[j][2]=((j+1)*10)+(j-1); j++;
        }
        if(reg[i]==')'&&reg[i+1]=='*')
        {
            q[0][2]=((j+1)*10)+1;
            q[j][2]=((j+1)*10)+1;
            j++;
        }
        i++;
    }
    printf("\n\tTransition Table \n");
    printf("_______________________________________\n");
    printf("Current State |\tInput |\tNext State");
    printf("\n_______________________________________\n");
    for(i=0;i<=j;i++)
    {
        if(q[i][0]!=0) printf("\n  q[%d]\t      |   a  |  q[%d]",i,q[i][0]);
        if(q[i][1]!=0) printf("\n  q[%d]\t      |   b  |  q[%d]",i,q[i][1]);
        if(q[i][2]!=0)
        {
            if(q[i][2]<10) printf("\n  q[%d]\t      |  e   |  q[%d]",i,q[i][2]);
            else printf("\n  q[%d]\t      |  e   |  q[%d] , q[%d]",i,q[i][2]/10,q[i][2]%10);
        }
    }
    printf("\n_______________________________________\n");
    return 0;
}
```

OUTPUT:

```
Given regular expression: (a|b)*

    Transition Table
_______________________________________
Current State |  Input |   Next State
_______________________________________

 q[0]         |   e   |  q[7] , q[1]
 q[1]         |   e   |  q[2] , q[4]
 q[2]         |   a   |  q[3]
 q[3]         |   e   |  q[6]
 q[4]         |   b   |  q[5]
 q[5]         |   e   |  q[6]
 q[6]         |   e  |  q[7] , q[1]
_______________________________________
```

5. Write a Lex program to generate tokens as identifiers, keywords, newline, tabs, whitespaces and characters.

PROGRAM:

```
%{
int n = 0 ;
%}

%%

"while"|"if"|"else" {n++;printf("\t keywords : %s", yytext);}


"int"|"float" {n++;printf("\t keywords : %s", yytext);}


 [a-zA-Z_][a-zA-Z0-9_]* {n++;printf("\t identifier : %s", yytext);}


"<="|"=="|"="|"++"|"-"|"*"|"+" {n++;printf("\t operator : %s", yytext);}

 [(){}|, ;] {n++;printf("\t separator : %s", yytext);}


[0-9]*"."[0-9]+ {n++;printf("\t float : %s", yytext);}


 [0-9]+ {n++;printf("\t integer : %s", yytext);}
. ;
%%

int main()
{
yylex();
printf("\n total no. of token = %d\n", n);
}
```

INPUT:

```
int p=0, d=1, c=2;
```

OUTPUT:

```
total no. of tokens = 13
```

6. Write a program in C or C++ language to implement Predictive Parsing Algorithm.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
char prol[7][10]={"s","A","A","B","B","C","C"};
char pror[7][10]={"Aa","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$"};

Dept. of CSE  17
char table[5][6][10];
{
switch(c)
{
case 'S':return0;
case 'A':return1;

case 'B':return2;
case 'C':return3;
case  'a':return0;
case 'b':return1;
case  'c':return2;
case 'd':return3;
case '$':return4;
}
retun(2);
}
void main()
{
int i,j,k;
clrscr();
for(i=0;i<5;i++)
for(j=0;j<6;j++)
strcpy(table[i][j]," ");
printf("\n The following is the predictive parsing table for the following grammar:\n");
for(i=0;i<7;i++)
printf("%s\n",prod[i]);
printf("\n Predictive 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("%s_10S",table[i][j]);
if(j==5)
printf("\n \n");
}
getch();
```

OUTPUT:

```
---------------------------------------------------------------------
       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->@       C->@
```

7. Write a program in C or C++ language to find the FIRST and FOLLOW of all the variables. Create functions for FIRST and FOLLOW.

PROGRAM:

```
#include<bits/stdc++.h>
using namespace std;

set<char> ss;
bool dfs(char i, char org, char last, map<char,vector<vector<char>>> &mp){
   bool rtake = false;
   for(auto r : mp[i]){
      bool take = true;
      for(auto s : r){
         if(s == i) break;
         if(!take) break;
         if(!(s>='A'&&s<='Z')&&s!='e'){
            ss.insert(s);
            break;
         }
         else if(s == 'e'){
            if(org == i||i == last)
            ss.insert(s);
            rtake = true;
            break;
         }
         else{
            take = dfs(s,org,r[r.size()-1],mp);
            rtake |= take;
         }
      }
   }
   return rtake;
}

int main(){
   int i,j;
   ifstream fin("inputfirstfollow.txt");
   string num;
   vector<int> fs;
   vector<vector<int>> a;
   map<char,vector<vector<char>>> mp;
   char start;
   bool flag = 0;
   cout<<"Grammar: "<<'\n';
```

```
while(getline(fin,num)){
    if(flag == 0) start = num[0],flag = 1;
    cout<<num<<'\n';
    vector<char> temp;
    char s = num[0];
    for(i=3;i<num.size();i++){
        if(num[i] == '|'){
            mp[s].push_back(temp);
            temp.clear();
        }
        else temp.push_back(num[i]);
    }
    mp[s].push_back(temp);
}
map<char,set<char>> fmp;
for(auto q : mp){
    ss.clear();
    dfs(q.first,q.first,q.first,mp);
    for(auto g : ss) fmp[q.first].insert(g);
}

cout<<'\n';
cout<<"FIRST: "<<'\n';
for(auto q : fmp){
    string ans = "";
    ans += q.first;
    ans += " = {";
    for(char r : q.second){
        ans += r;
        ans += ',';
    }
    ans.pop_back();
    ans+="}";
    cout<<ans<<'\n';
}

map<char,set<char>> gmp;
gmp[start].insert('$');
int count = 10;
while(count--){
    for(auto q : mp){
        for(auto r : q.second){
            for(i=0;i<r.size()-1;i++){
                if(r[i]>='A'&&r[i]<='Z'){
                    if(!(r[i+1]>='A'&&r[i+1]<='Z')) gmp[r[i]].insert(r[i+1]);
                    else {
                        char temp = r[i+1];
                        int j = i+1;
                        while(temp>='A'&&temp<='Z'){
                            if(*fmp[temp].begin()=='e'){
                                for(auto g : fmp[temp]){
                                    if(g=='e') continue;
                                    gmp[r[i]].insert(g);
                                }
                                j++;
                                if(j<r.size()){
                                    temp = r[j];
                                    if(!(temp>='A'&&temp<='Z')){
                                        gmp[r[i]].insert(temp);
                                        break;
                                    }
                                }
                                else{
                                    for(auto g : gmp[q.first]) gmp[r[i]].insert(g);
                                    break;
                                }
                            }
                            else{
                                for(auto g : fmp[temp]){
                                    gmp[r[i]].insert(g);
                                }
                                break;
                            }
                        }
                    }
                }
            }
            if(r[r.size()-1]>='A'&&r[r.size()-1]<='Z'){
                for(auto g : gmp[q.first]) gmp[r[i]].insert(g);
            }
        }
    }
}

cout<<'\n';
cout<<"FOLLOW: "<<'\n';
for(auto q : gmp){
    string ans = "";
    ans += q.first;
    ans += " = {";
    for(char r : q.second){
        ans += r;
        ans += ',';
    }
```

```
        ans.pop_back();
        ans+="}";
        cout<<ans<<'\n';
    }
    return 0;
}
```

OUTPUT:

```
Grammar:
S->ACB|CbB|Ba
A->da|BC
B->g|e
C->h|e

FIRST:
A = {d,e,g,h}
B = {e,g}
C = {e,h}
S = {a,b,d,e,g,h}

FOLLOW:
A = {$,g,h}
B = {$,a,g,h}
C = {$,b,g,h}
S = {$}
```

8. Write a program in C or C++ language to implement LR Parser.

PROGRAM:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
int i,j,k,m,n=0,o,p,ns=0,tn=0,rr=0,ch=0;
char cread[15][10],gl[15],gr[15][10],temp,templ[15],tempr[15][10],*ptr,temp2[5];
char dfa[15][10];
struct states
{
char lhs[15],rhs[15][10];
int n;//state number
}I[15];
int compstruct(struct states s1,struct states s2)
{
int t;
if(s1.n!=s2.n)
return 0;
if( strcmp(s1.lhs,s2.lhs)!=0 )
return  0;
for(t=0;t<s1.n;t++)
if( strcmp(s1.rhs[t],s2.rhs[t])!=0 )
return 0;
return 1;
}
void moreprod()
{
int r,s,t,l1=0,rr1=0;
char *ptr1,read1[15][10];
for(r=0;r<I[ns].n;r++)
{
ptr1=strchr(I[ns].rhs[l1],'.');
t=ptr1-I[ns].rhs[l1];
if( t+1==strlen(I[ns].rhs[l1]) )
```

```
l1++;
continue;
}
temp=I[ns].rhs[l1][t+1];
l1++;
for(s=0;s<rr1;s++)
if( temp==read1[s][0] )
break;
if(s==rr1)
```

```
{
read1[rr1][0]=temp;
rr1++;
}
else
continue;
for(s=0;s<n;s++)
{
if(gl[s]==temp)
{
I[ns].rhs[I[ns].n][0]='.';
I[ns].rhs[I[ns].n][1]='\0';
strcat(I[ns].rhs[I[ns].n],gr[s]);
I[ns].lhs[I[ns].n]=gl[s];
I[ns].lhs[I[ns].n+1]='\0';
I[ns].n++;
}
}
}
}
void canonical(int l)
{
int t1;
char read1[15][10],rr1=0,*ptr1;
for(i=0;i<I[l].n;i++)
{
temp2[0]='.';
ptr1=strchr(I[l].rhs[i],'.');
t1=ptr1-I[l].rhs[i];
if( t1+1==strlen(I[l].rhs[i]) )
continue;
temp2[1]=I[l].rhs[i][t1+1];
temp2[2]='\0';
for(j=0;j<rr1;j++)
```

```
if( strcmp(temp2,read1[j])==0 )
break;
if(j==rr1)
{
strcpy(read1[rr1],temp2);
read1[rr1][2]='\0';
rr1++;
}
else
continue;
for(j=0;j<I[0].n;j++)
```

```
{
ptr=strstr(I[l].rhs[j],temp2);
if( ptr )
{
templ[tn]=I[l].lhs[j];
templ[tn+1]='\0';
strcpy(tempr[tn],I[l].rhs[j]);
tn++;
}
}
for(j=0;j<tn;j++)
{
ptr=strchr(tempr[j],'.');
p=ptr-tempr[j];
tempr[j][p]=tempr[j][p+1];
tempr[j][p+1]='.';
I[ns].lhs[I[ns].n]=templ[j];
I[ns].lhs[I[ns].n+1]='\0';
strcpy(I[ns].rhs[I[ns].n],tempr[j]);
I[ns].n++;
}
moreprod();
for(j=0;j<ns;j++)
{
//if ( memcmp(&I[ns],&I[j],sizeof(struct states))==1 )
if( compstruct(I[ns],I[j])==1 )
{
I[ns].lhs[0]='\0';
for(k=0;k<I[ns].n;k++)
I[ns].rhs[k][0]='\0';
I[ns].n=0;
dfa[l][j]=temp2[1];
break;
```

```
}
}
if(j<ns)
{
tn=0;
for(j=0;j<15;j++)
{
templ[j]='\0';
tempr[j][0]='\0';
}
continue;
}
```

```
dfa[l][j]=temp2[1];
printf("\n\nI%d :",ns);
for(j=0;j<I[ns].n;j++)
printf("\n\t%c -> %s",I[ns].lhs[j],I[ns].rhs[j]);
//getch();
ns++;
tn=0;
for(j=0;j<15;j++)
{
templ[j]='\0';
tempr[j][0]='\0';
}
}
}
void main()
{
FILE *f;
int l;
//clrscr();
for(i=0;i<15;i++)
{
I[i].n=0;
I[i].lhs[0]='\0';
I[i].rhs[0][0]='\0';
dfa[i][0]= '\0';
}
f=fopen("tab6.txt","r");
while(!feof(f))
{
fscanf(f,"%c",&gl[n]);
fscanf(f,"%s\n",gr[n]);
```

```
n++;
}
printf("THE GRAMMAR IS AS FOLLOWS\n");
for(i=0;i<n;i++)
printf("\t\t\t\t%c -> %s\n",gl[i],gr[i]);
I[0].lhs[0]='Z';
strcpy(I[0].rhs[0],".S");
I[0].n++;
l=0;
for(i=0;i<n;i++)
{
temp=I[0].rhs[l][1];
l++;
for(j=0;j<rr;j++)
```

```
if( temp==cread[j][0] )
break;
if(j==rr)
{
cread[rr][0]=temp;
rr++;
}
else
continue;
for(j=0;j<n;j++)
{
if(gl[j]==temp)
{
I[0].rhs[I[0].n][0]='.';
strcat(I[0].rhs[I[0].n],gr[j]);
I[0].lhs[I[0].n]=gl[j];
I[0].n++;
}
}
}
ns++;
printf("\nI%d :\n",ns-1);
for(i=0;i<I[0].n;i++)
printf("\t%c -> %s\n",I[0].lhs[i],I[0].rhs[i]);
for(l=0;l<ns;l++)
canonical(l);
printf("\n\n\t\tPRESS ANY KEY FOR TABLE");
//getch();
```

```
//clrscr();
printf("\t\t\t\nDFA TABLE IS AS FOLLOWS\n\n\n");
for(i=0;i<ns;i++)
{
printf("I%d : ",i);
for(j=0;j<ns;j++)
if(dfa[i][j]!='\0')
printf("'%c'->I%d | ",dfa[i][j],j);
printf("\n\n\n");
}
printf("\n\n\n\t\tPRESS ANY KEY TO EXIT");
//getch();
}
```

OUTPUT:

```
I0 : 'a' -> I1 | 'b' -> I2
 I1 : 'b' -> I3
I2 : Completed Item: A -> b
I3 : Completed Item: S -> aA
```

9. Write a program in C or C++ to generate the three-address code.

PROGRAM:

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

struct three
```

```
{
char data[10],temp[7];
}s[30];
void main()
{
char d1[7],d2[7]="t";
int i=0,j=1,len=0;
FILE *f1,*f2;
clrscr();
f1=fopen("sum.txt","r");
f2=fopen("out.txt","w");
while(fscanf(f1,"%s",s[len].data)!=EOF)
len++;
itoa(j,d1,7);
strcat(d2,d1);
strcpy(s[j].temp,d2);
strcpy(d1,"");
strcpy(d2,"t");
if(!strcmp(s[3].data,"+"))
{
fprintf(f2,"%s=%s+%s",s[j].temp,s[i+2].data,s[i+4].data);
j++;
}
else if(!strcmp(s[3].data,"-"))
{
fprintf(f2,"%s=%s-%s",s[j].temp,s[i+2].data,s[i+4].data);
j++;
}
for(i=4;i<len-2;i+=2)
{
itoa(j,d1,7);
strcat(d2,d1);
strcpy(s[j].temp,d2);
if(!strcmp(s[i+1].data,"+"))
fprintf(f2,"\n%s=%s+%s",s[j].temp,s[j-1].temp,s[i+2].data);
else if(!strcmp(s[i+1].data,"-"))
fprintf(f2,"\n%s=%s-%s",s[j].temp,s[j-1].temp,s[i+2].data);
strcpy(d1,"");
strcpy(d2,"t");
j++;
}
fprintf(f2,"\n%s=%s",s[0].data,s[j-1].temp);
```

```
fclose(f1);
fclose(f2);
getch();

}
```

INPUT:

```
 sum.txt
out = in1 + in2 + in3 - in4
```

OUTPUT :

```
out.txt
t1=in1+in2
t2=t1+in3
t3=t2-in4
out=t3
```

10. Write a program in C or C++ to generate machine code from the abstract syntax tree generated by the parser.

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int label[20];
int no=0;
int main()
{
FILE *fp1,*fp2;
char fname[10],op[10],ch;
char operand1[8],operand2[8],result[8];
int i=0,j=0;
printf("\n Enter filename of the intermediate code");
scanf("%s",&fname);
fp1=fopen(fname,"r");
fp2=fopen("target.txt","w");
if(fp1==NULL || fp2==NULL)
{
printf("\n Error opening the file");
exit(0);
}
while(!feof(fp1))
{
fprintf(fp2,"\n");
fscanf(fp1,"%s",op);
i++;
if(check_label(i))
fprintf(fp2,"\nlabel#%d",i);
if(strcmp(op,"print")==0)
{
fscanf(fp1,"%s",result);
fprintf(fp2,"\n\t OUT %s",result);
}
 if(strcmp(op,"goto")==0)
{
fscanf(fp1,"%s %s",operand1,operand2);
fprintf(fp2,"\n\t JMP %s,label#%s",operand1,operand2);
label[no++]=atoi(operand2);
}
 if(strcmp(op,"[]=")==0)
{
fscanf(fp1,"%s %s %s",operand1,operand2,result);
```

```
fprintf(fp2,"\n\t STORE %s[%s],%s",operand1,operand2,result);
}
 if(strcmp(op,"uminus")==0)
{
fscanf(fp1,"%s %s",operand1,result);
fprintf(fp2,"\n\t LOAD -%s,R1",operand1);
fprintf(fp2,"\n\t STORE R1,%s",result);
}
 switch(op[0])
{
case '*': fscanf(fp1,"%s %s %s",operand1,operand2,result);
 fprintf(fp2,"\n \t LOAD",operand1);
 fprintf(fp2,"\n \t LOAD %s,R1",operand2);
 fprintf(fp2,"\n \t MUL R1,R0");
 fprintf(fp2,"\n \t STORE R0,%s",result);
 break;
case '+': fscanf(fp1,"%s %s %s",operand1,operand2,result);
 fprintf(fp2,"\n \t LOAD %s,R0",operand1);
 fprintf(fp2,"\n \t LOAD %s,R1",operand2);
 fprintf(fp2,"\n \t ADD R1,R0");
 fprintf(fp2,"\n \t STORE R0,%s",result);
 break;
case '-': fscanf(fp1,"%s %s %s",operand1,operand2,result);
 fprintf(fp2,"\n \t LOAD %s,R0",operand1);
 fprintf(fp2,"\n \t LOAD %s,R1",operand2);
 fprintf(fp2,"\n \t SUB R1,R0");
 fprintf(fp2,"\n \t STORE R0,%s",result);
 break;
case '/': fscanf(fp1,"%s %s %s",operand1,operand2,result);
 fprintf(fp2,"\n \t LOAD %s,R0",operand1);
 fprintf(fp2,"\n \t LOAD %s,R1",operand2);
 fprintf(fp2,"\n \t DIV R1,R0");
 fprintf(fp2,"\n \t STORE R0,%s",result);
 break;
case '%': fscanf(fp1,"%s %s %s",operand1,operand2,result);
 fprintf(fp2,"\n \t LOAD %s,R0",operand1);
 fprintf(fp2,"\n \t LOAD %s,R1",operand2);
 fprintf(fp2,"\n \t DIV R1,R0");
 fprintf(fp2,"\n \t STORE R0,%s",result);
 break;
case '=': fscanf(fp1,"%s %s",operand1,result);
 fprintf(fp2,"\n\t STORE %s %s",operand1,result);
 break;
case '>': j++;
 fscanf(fp1,"%s %s %s",operand1,operand2,result);
 fprintf(fp2,"\n \t LOAD %s,R0",operand1);
 fprintf(fp2,"\n\t JGT %s,label#%s",operand2,result);
 label[no++]=atoi(result);
 break;
```

```
case '<': fscanf(fp1,"%s %s %s",operand1,operand2,result);
 fprintf(fp2,"\n \t LOAD %s,R0",operand1);
 fprintf(fp2,"\n\t JLT %s, label#%d",operand2,result);
 label[no++]=atoi(result);
 break;
}
}
fclose(fp2); fclose(fp1);
fp2=fopen("target.txt","r");
if(fp2==NULL)
 {
printf("Error opening the file\n");
exit(0);
 }
do
 {
ch=fgetc(fp2);
printf("%c",ch);
 }while(ch!=EOF);
fclose(fp1);
return 0;
}
int check_label(int k)
{
int i;
for(i=0;i<no;i++)
{
if(k==label[i])
return 1;
}
 return 0;
 }
```

OUTPUT:

```
Enter filename of the intermediate code: int.txt
STORE t1, 2
STORE a[0], 1
STORE a[1], 2
STORE a[2], 3
LOAD t1, R0
LOAD 6, R1
ADD R1, R0
STORE R0, t3
LOAD a[2], R0
LOAD t2, R1
ADD R1,R0
STORE R0,t3
LOAD a[t2],R0
LOAD t1,R1
SUB R1,R0
STORE R0,t2
LOAD t3,R0
LOAD t2,R1
DIV R1,R0
STORE R0,t2
LOAD t2,R1
STORE R1,t2
LOAD t2,R0
JGT 5, label#11
Label#11: OUT t2
JMP t2, label#13
Label#13: STORE t3, 99
LOAD 25, R1
STORE R1,t2
LOAD t2,R0
LOAD t3,R1
MUL R1,R0
STORE R0,t3
LOAD t1,R1
STORE R1,t1
LOAD t1,R0
LOAD t3,R1
ADD R1,R0
STORE R0,t4
OUT t4
```