**NAME:** Mitali Dave **CLASS:** TE-A **SAP ID:** 60004150014

**AIM: Find FIRST and FOLLOW of LL(1) grammar.**

**CODE:**

import java.util.\*; import java.io.\*; class First\_Follow {

static char ntermnl[], termnl[];

static int ntlen, tlen;

static String grmr[][], fst[], flw[];

public static void main(String args[]) throws IOException {

String nt, t;

int i, j, n;

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the non-terminals");

nt = br.readLine();

ntlen = nt.length();

ntermnl = new char[ntlen];

ntermnl = nt.toCharArray();

System.out.println("Enter the terminals");

t = br.readLine();

tlen = t.length();

termnl = new char[tlen];

termnl = t.toCharArray();

System.out.println("Specify the grammar(Enter 9 for epsilon production)");

grmr = new String[ntlen][];

for (i = 0; i < ntlen; i++) {

System.out.println("Enter the number of productions for " + ntermnl[i]);

n = Integer.parseInt(br.readLine());

grmr[i] = new String[n];

System.out.println("Enter the productions");

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

grmr[i][j] = br.readLine();

}

fst = new String[ntlen];

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

fst[i] = first(i);

System.out.println("First Set");

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

System.out.println(removeDuplicates(fst[i]));

flw = new String[ntlen];

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

flw[i] = follow(i);

System.out.println("Follow Set");

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

System.out.println(removeDuplicates(flw[i]));

}

static String first(int i) {

int j, k, l = 0, found = 0;

String temp = "", str = "";

for (j = 0; j < grmr[i].length; j++) //number of productions

{

for (k = 0; k < grmr[i][j].length(); k++, found = 0) //when nonterminal has epsilon production

{

for (l = 0; l < ntlen; l++) //finding nonterminal

{

if (grmr[i][j].charAt(k) == ntermnl[l]) //for nonterminal in first set

{

str = first(l);

if (!(str.length() == 1 && str.charAt(0) == '9')) //when epsilon production is the only nonterminal

production

temp = temp + str;

found = 1;

break;

}

}

if (found == 1) {

if (str.contains("9")) //here epsilon will lead to next nonterminal’s first set

continue;

} else //if first set includes terminal

temp = temp + grmr[i][j].charAt(k);

break;

}

}

return temp;

}

static String follow(int i) {

char pro[], chr[];

String temp = "";

int j, k, l, m, n, found = 0;

if (i == 0)

temp = "$";

for (j = 0; j < ntlen; j++) {

for (k = 0; k < grmr[j].length; k++) //entering grammar matrix

{

pro = new char[grmr[j][k].length()];

pro = grmr[j][k].toCharArray();

for (l = 0; l < pro.length; l++) //entering each production

{

if (pro[l] == ntermnl[i]) //finding the nonterminal whose follow set is to be found

{

if (l == pro.length - 1) //if it is the last terminal/non-terminal then follow of current non-terminal

{

if (j < i)

temp = temp + flw[j];

} else {

for (m = 0; m < ntlen; m++) {

if (pro[l + 1] == ntermnl[m]) //first of next non-terminal otherwise (else later…)

{

chr = new char[fst[m].length()];

chr = fst[m].toCharArray();

for (n = 0; n < chr.length; n++) {

if (chr[n] == '9') //if first includes epsilon

{

if (l + 1 == pro.length - 1)

temp = temp + follow(j); //when non-terminal is second last

else

temp = temp + follow(m);

} else

temp = temp + chr[n]; //include whole first set except epsilon

}

found = 1;

}

}

if (found != 1)

temp = temp + pro[l + 1]; //follow set will include terminal(else is here)

}

}

}

}

}

return temp;

}

static String removeDuplicates(String str) {

int i;

char ch;

boolean seen[] = new boolean[256];

StringBuilder sb = new StringBuilder(seen.length);

for (i = 0; i < str.length(); i++) {

ch = str.charAt(i);

if (!seen[ch]) {

seen[ch] = true;

sb.append(ch);

}

}

return sb.toString();

}

}

**OUTPUT:**

1.

Enter the non-terminals

SA

Enter the terminals

abc

Specify the grammar(Enter 9 for epsilon production)

Enter the number of productions for S

2

Enter the productions

Aa

Ac

Enter the number of productions for A

1

Enter the productions

b

First Set

b

b

Follow Set

$

Ac

**AIM: Remove Left Recursion and Left Factoring in LL(1) grammar.**

**CODE FOR LEFT RECURSION:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

int main()

{

int i=0,j=0,k=0,l=0;

char inp[50]={0},part1[50]={0},part2[50]={0};

printf("Enter production in the form A-> Aa1|Aa2|...|b1|b2..\nA-> ");

gets(inp);

inp[strlen(inp)]='\0';

if(inp[0]=='A')

{

abc:for(i++,j=0;inp[i]!='|';i++,l++)

part1[l]=inp[i];

part1[l]='A';

part1[++l]='\'';

part1[++l]='|';

part1[++l]='\0';

j=0;

do

{

{

i++;

k=i+1;

if(inp[k]=='A')

goto abc;

}

else{

for(i++;inp[i]!='|'&&inp[i];i++,j++)

part2[j]=inp[i];

if(inp[i]=='|')

{

part2[j]='A'; part2[++j]='\''; part2[++j]='|'; j++;

}

}

}while(inp[i]);

part2[j]='A';

part2[++j]='\''; part2[++j]='\0'; printf("A-> %s",part2);

printf("\nA'-> %s",part1);

printf("e");

}

else

printf("No LR");

return 0;

}

**CODE FOR LEFT FACTORING:**

#include<stdio.h>

#include<conio.h>

#include<string.h>

int main()

{

int slash[20]={0},i=0,j=0,k=0,l=0,slash\_no=0;

char inp[50]={0},part1[50]={0};

printf("Enter production in the form A-> aB1|aB2|.....\nA-> ");

gets(inp);

inp[strlen(inp)]='\0';

for(i=0;inp[i];i++)

{

if(inp[i]=='|')

{

slash[slash\_no]=i;

slash\_no++;

}

}

for(i=0;i<slash\_no;i++)

{

if(inp[0]!=inp[slash[i]+1]||(inp[0]<='a'&&inp[0]>='z')) //accept only terminals (small case)

{

printf("No LF");

return 0;

}

}

i=0; j=0; while(inp[i])

{

for(i++;inp[i]!='|'&&inp[i]&&inp[i]!=inp[0];i++,j++)

part1[j]=inp[i];

if(inp[i]=='|')

part1[j++]='|';

}

part1[++j]='\0';

printf("A-> %cA'",inp[0]);

printf("\nA'-> %s",part1);

return 0;

}

**OUTPUT FOR LEFT RECURSION:**

Enter production in the form A-> Aa1|Aa2|...|b1|b2..

A-> Ac|Aad|Ab|bd|acd

A-> bdA'|acdA'

A'-> cA'|adA'|bA'|e

**OUTPUT FOR LEFT RECURSION:**

Enter production in the form A-> aB1|aB2|.....

A-> aV|abj|asdG

A-> aA'

A'-> V|bj|sdG

**AIM: To implement Intermediate Code Generation**

**CODE:** package icg; import java.util.\*; public class icg {

public static String replaceStuff(String equation, int i, String toReplace) {

int midIndex = equation.indexOf(toReplace);

/\*

\* System.out.println(midIndex); System.out.println(toReplace);

\*/

String newEquation = "";

String toFind = "";

if (!(equation.charAt(midIndex - 2) == 't') && !(equation.charAt(midIndex + 1) == 't')) {

toFind = equation.substring(midIndex - 1, midIndex + 2);

newEquation = equation.substring(0, midIndex - 1) + "t" + Integer.toString(i) +

equation.substring(midIndex + 2, equation.length());

} else if (equation.charAt(midIndex - 2) == 't') {

if (equation.charAt(midIndex + 1) == 't') {

toFind = equation.substring(midIndex - 2, midIndex + 3);

newEquation = equation.substring(0, midIndex - 2) + "t" + Integer.toString(i) +

equation.substring(midIndex + 3, equation.length());

} else {

toFind = equation.substring(midIndex - 2, midIndex + 2);

newEquation = equation.substring(0, midIndex - 2) + "t" + Integer.toString(i) +

equation.substring(midIndex + 2, equation.length());

}

} else {

toFind = equation.substring(midIndex - 1, midIndex + 3);

newEquation = equation.substring(0, midIndex - 1) + "t" + Integer.toString(i) +

equation.substring(midIndex + 3, equation.length());

}

System.out.println("t" + i + "=" + toFind);

return newEquation;

}

public static void assignment(String equation) {

int i = 1;

while (equation.contains("^") || equation.contains("\*") || equation.contains("/") ||

equation.contains("+") || equation.contains("-")) {

if (equation.contains("^")) {

String toReplace = "^";

int midIndex = equation.lastIndexOf(toReplace);

String toFind;

String newEquation = "";

if (!(equation.charAt(midIndex - 2) == 't') && !(equation.charAt(midIndex + 1) == 't')) {

toFind = equation.substring(midIndex - 1, midIndex + 2);

newEquation = equation.substring(0, midIndex - 1) + "t" + Integer.toString(i) +

equation.substring(midIndex + 2, equation.length());

} else if (equation.charAt(midIndex - 2) == 't') {

if (equation.charAt(midIndex + 1) == 't') {

toFind = equation.substring(midIndex - 2, midIndex + 3);

newEquation = equation.substring(0, midIndex - 2) + "t" + Integer.toString(i) +

equation.substring(midIndex + 3, equation.length());

} else {

toFind = equation.substring(midIndex - 2, midIndex + 2);

newEquation = equation.substring(0, midIndex - 2) + "t" + Integer.toString(i) +

equation.substring(midIndex + 2, equation.length());

}

} else {

toFind = equation.substring(midIndex - 1, midIndex + 3);

newEquation = equation.substring(0, midIndex - 1) + "t" + Integer.toString(i) +

equation.substring(midIndex + 3, equation.length());

}

System.out.println("t" + i + "=" + toFind);

equation = newEquation;

} else if (equation.contains("\*") || equation.contains("/")) {

if (equation.contains("\*") && equation.contains("/")) {

if (equation.indexOf("\*") < equation.indexOf("/")) {

String toReplace = "\*";

equation = replaceStuff(equation, i, toReplace);

} else {

String toReplace = "/";

equation = replaceStuff(equation, i, toReplace);

}

} else if (equation.contains("\*")) {

String toReplace = "\*";

equation = replaceStuff(equation, i, toReplace);

} else {

String toReplace = "/";

equation = replaceStuff(equation, i, toReplace);

}

} else {

if (equation.contains("+") && equation.contains("-")) {

if (equation.indexOf("+") < equation.indexOf("-")) {

String toReplace = "+";

equation = replaceStuff(equation, i, toReplace);

} else {

String toReplace = "-";

equation = replaceStuff(equation, i, toReplace);

}

} else if (equation.contains("+")) {

String toReplace = "+";

equation = replaceStuff(equation, i, toReplace);

} else {

String toReplace = "-";

equation = replaceStuff(equation, i, toReplace);

}

}

System.out.println(equation);

i = i + 1;

}

}

public static int replaceBool(int index, String equation, int total) { String condition = equation.substring(0, 3);

System.out.println(Integer.toString(index) + ": if " + condition + " go to " + Integer.toString(index + 3));

index = index + 1;

System.out.println(Integer.toString(index) + ": t" + Integer.toString(total) + "=1");

index++;

System.out.println(Integer.toString(index) + ": go to" + Integer.toString(index + 2));

index++;

System.out.println(Integer.toString(index) + ": t" + Integer.toString(total) + "=0");

index++;

return index;

}

public static void boolState(String equation) {

int index = 100;

int total = 0;

int[] andCount = new int[5];

for (int i = 0; i < 5; i++)

andCount[i] = 0;

while (equation.contains("&&") || equation.contains("||")) {

String operand = equation.substring(3, 5);

if (operand.equals("&&"))

andCount[total] = 1;

total = total + 1;

equation = equation.substring(5, equation.length());

index = replaceBool(index, equation, total);

}

total = total + 1;

index = replaceBool(index, equation, total);

int totalConditions = total;

for (int i = 0; i < totalConditions - 1; i++) {

total = total + 1;

if (andCount[i] == 1)

System.out.println(Integer.toString(index) + ": t" + Integer.toString(total) + " = t" + Integer.toString(i + 1)

+ " and t" + Integer.toString(i + 2));

else

System.out.println(Integer.toString(index) + ": t" + Integer.toString(total) + " = t" + Integer.toString(i + 1)

+ " or t" + Integer.toString(i + 2));

index++;

}

}

public static void controlState(String[] equations, int number) {

int index = 100;

for (int i = 0; i < number; i++) {

String equation = equations[i];

if (equation.equals("{"))

;

else if (equation.contains("while")) {

String condition = equation.substring(equation.indexOf("(") + 1, equation.indexOf(")"));

System.out.println("Lstart if " + condition + " go to Lstop");

} else if (equation.equals("}")) {

System.out.println(Integer.toString(index) + ": go to Lstart");

System.out.println("Lstop");

} else {

System.out.println(Integer.toString(index) + ": " + equation.substring(0, equation.length() - 1));

index++;

}

}

}

public static void main(String args[]) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the assignment equation.");

String equation = sc.next();

assignment(equation);

System.out.println("Enter the boolean equation.");

equation = sc.next();

boolState(equation);

System.out.println("Enter number of lines of the control statement. { and } on one line each.");

int n = sc.nextInt();

String[] control = new String[n];

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

control[i] = sc.next();

}

controlState(control, n);

}

}

**OUTPUT:**

Enter the assignment equation. k=a-b^c^d/f

t1=c^d

k=a-b^t1/f

t2=b^t1

k=a-t2/f

t3=t2/f

k=a-t3

t4=a-t3

k=t4

Enter the boolean equation. p<q&&r>s||u>v

100: if r>s go to 103

101: t1=1

102: go to104

103: t1=0

104: if u>v go to 107

105: t2=1

106: go to108

107: t2=0

108: if u>v go to 111

109: t3=1

110: go to112

111: t3=0

112: t4 = t1 and t2

113: t5 = t2 or t3

Enter number of lines of the control statement. { and } on one line each.

5 while(i<10)

{ x=10; i=i+1;

}

Lstart if i<10 go to Lstop

100: x=10

101: i=i+1

102: go to Lstart

Lstop

**NAME:** Mitali Dave **CLASS:** TE-A **SAP ID:** 60004150014

**AIM: To implement pass 1 of two pass assemblers**

**CODE:**

**pass1assembler.c**

#include<stdio.h>

#include<conio.h>

#include<string.h>

int main()

{

char id[10],str[10], mot[][50]={{"L"},{"ST"},{"A"}}, pot[][50]={{"START"},{"END"},{"DC"},{"DS"},{"USING"}}, st[][50]={{"PRG"},{"FOUR"},{"FIVE"},{"TEMP"}}, lt[][50]={{"=F'4'"},{"=F'5'"}},

test[][50]={{""},{""}};

char ch[10]={0};

int x,i,j,k,parsed,motlen,potlen,stlen,ltlen,oc=0; FILE \*fp1,\*fpmot,\*fppot,\*fpst,\*fplt; fp1=fopen("pass1File.txt","r"); fpmot=fopen("mot.txt","a"); fppot=fopen("pot.txt","a"); fpst=fopen("symbol.txt","a"); fplt=fopen("literal.txt","a"); motlen=sizeof(mot)/sizeof(mot[0]); potlen=sizeof(pot)/sizeof(pot[0]); stlen=sizeof(st)/sizeof(st[0]); ltlen=sizeof(lt)/sizeof(lt[0]);

//printf("%d %d %d %d\n",motlen,potlen,stlen,ltlen); for(k=0;fgets(str,50,fp1)!=NULL;k+=2); //to find address of END k-=2;

fclose(fp1); fp1=fopen("pass1File.txt","r"); for(x=0;fgets(ch,50,fp1)!=NULL;x+=2)

{

parsed=0;

printf("%d %s",x,ch);

for(i=0;i<motlen&&parsed==0;i++)

{

if(strstr(ch,mot[i]))

{

strcpy(str,strstr(ch,mot[i]));

oc=0;

for(j=0;j<strlen(str);j++)

{

if(str[j]==' ')

oc++; //calc operands

}

//printf("\n1. %d %s",oc,mot[i]); fprintf(fpmot,"\n%s\t\t2\t\t%d",mot[i],oc); parsed=1;

break;

}

}

for(i=0;i<potlen&&parsed==0;i++)

{

if(strstr(ch,pot[i]))

{

// printf("\n2. %d %s",x,pot[i]); fprintf(fppot,"\n%s\t\t%d",pot[i],x); parsed=1;

break;

}

}

//symbols are followed by pseudo-opcodes for(i=0;i<stlen;i++)

{

parsed=0;

for(j=0;j<potlen&&parsed==0;j++)

{

//printf("\n %s",pot[j]);

if((strstr(ch,":")&&strstr(ch,st[i])) || (strstr(ch,pot[j])&&strstr(ch,st[i])))

{

//printf("\n3. %d %s",x,st[i]); fprintf(fpst,"\n%s\t%d\t2\tR",st[i],x); parsed=1;

break;

}

}

if(parsed==1)

i++; //to not let N1 twice

}

for(i=0;i<ltlen&&parsed==0;i++)

{

if((strstr(ch,"=")&&strstr(ch,lt[i])))

{

// printf("\n4. %d %s",k,lt[i]);

fprintf(fplt,"\n%s\t%d\t2\tR",lt[i],k);

parsed=1;

break;

}

}

}

fclose(fp1); fclose(fpmot); fclose(fppot); fclose(fpst); fclose(fplt);

printf("\nPass 1 Complete...Tables updated...");

return 0;

}

**pass1File.txt**

PRG START 0

USING \*,15

L 1,FIVE

A 1,FOUR ST 1,TEMP FOUR DC =F’4’ FIVE DC =F’5’ TEMP DS 1F END

**OUTPUT:**

**mot.txt**

ST 2 1

L 2 1

A 2 1

ST 2 1

**pot.txt**

USING 2

DC 10

DC 12

DS 14

END 16

**symbol.txt**

PRG 0 2 R FOUR 10 2 R FIVE 12 2 R TEMP 14 2 R

**AIM: To implement single pass macro processor**

**CODE:**

**macrocode.c**

#include<stdio.h>

#include<conio.h>

void main()

{

FILE \*finput, \*fmdt, \*fmnt,\*fpala,\*fcala;

char t1[20],t2[20],t3[20]; int count=0; finput=fopen("input.txt","r"); fmdt=fopen("mdt.txt","w"); fmnt=fopen("mnt.txt","w"); fpala=fopen("pala.txt","w"); fcala=fopen("cala.txt","w"); fscanf(finput,"%s%s%s",t1,t2,t3); while(count!=2)

{

if(strcmp(t1,"Macro")==0)

{

fprintf(fmdt,"\t%s\t%s\n",t2,t3); fprintf(fmnt,"%s\n",t2); fprintf(fpala,"%s\n",t3); fscanf(finput,"%s%s%s",t1,t2,t3);

//fprintf(fpala,"%s\n",t3);

//count++;

//fprintf(fmdt,"Count = %d",count);

}

else if(strcmp(t1,"Mend")==0)

{

}

else

{

}

}

count++; fprintf(fmdt,"%s\n",t1); strcpy(t1,t2); strcpy(t2,t3); fscanf(finput,"%s",t3);

fprintf(fmdt,"%s\t%s\t%s\n",t1,t2,t3);

fscanf(finput,"%s%s%s",t1,t2,t3);

//fprintf(fmdt,"%s\n",t1);

fprintf(fcala,"%s\n",t2);

fscanf(finput,"%s",t1); fprintf(fcala,"%s\n",t1); fclose(finput); fclose(fmnt); fclose(fmdt); fclose(fpala); fclose(fcala);

//fprintf(fmdt,"%s\t%s\t%s",t1,t2,t3);

}

**INPUT.TXT**

Macro M1 &Arg1,&Arg2,&Arg3

Load A, &Arg1

ADD B, &Arg2

Store C, &Arg3

Mend

Macro M2 &Arg1,&Arg2,&Arg3

Store A, &Arg1

ADD B, &Arg2

SUB C, &Arg3

Mend

M1 2,3,4

M2 A1,A2,A3

**OUTPUT: PALA.TXT**

&Arg1,&Arg2,&Arg3

&Arg1,&Arg2,&Arg3

**CALA.TXT**

2,3,4

A1,A2,A3

**MDT.TXT**

M1 &Arg1,&Arg2,&Arg3

Load A, &Arg1

ADD B, &Arg2

Store C, &Arg3

Mend

M2 &Arg1,&Arg2,&Arg3

Store A, &Arg1

ADD B, &Arg2

SUB C, &Arg3

Mend

**MNT.TXT**

M1

M2

**AIM: To implement code generation**

**CODE:**

#include<bits/stdc++.h>

#define FORA(i, n, ii, s) for(int i=s;i<n;i+=ii)

#define FOR(i, n) for(int i=0;i<n;i++)

#define FORD(i, n) for(int i=n;i>=0;i--)

#define FORF(i, n, s) for(int i=s;i<n;i++)

#define MIN(x, y) ((x<y)?x:y)

#define MAX(x, y) ((x>y)?x:y)

#define ll long long

#define il long

#define ii pair<int,int>

#define vll vector<ll>

#define vi vector<int>

#define vii vector<ii>

#define MOD (int)(1e9 + 7)

#define INF (int)(1e9)

#define INFLL (ll)(1e18)

#define flash ios\_base::sync\_with\_stdio(false),cin.tie(NULL)

using namespace std; map<char,string> assm; int main() {

Flash;

assm['='] = "MOV"; assm['+'] = "ADD"; assm['-'] = "SUB"; assm['\*'] = "MUL"; assm['/'] = "DIV"; string inp;

int rg\_no = 0; vector<string> stk; cin>>inp;

string tem = "";

int lsi=0;

char orp = '='; FOR(i,inp.size()){

if(inp[i] != '='){

tem += inp[i];

}

else {lsi=i; break;}

}

stk.push\_back(tem);

tem = "";

bool fl = false;

FORF(i,inp.size(),lsi+1){

if(inp[i] != '+' && inp[i] != '-' && inp[i] != '\*' && inp[i] != '/'){

tem += inp[i];

}

else {lsi=i;fl=true;break;}

}

stk.push\_back(tem);

if(fl){

orp=inp[lsi]; tem = ""; FORF(i,inp.size(),lsi+1){

if(inp[i] != '+' && inp[i] != '-' && inp[i] != '\*' && inp[i] != '/'){

tem += inp[i];

}

else {lsi=i;fl=true;break;}

}

}

if(fl){

}else{

}

return 0;

}

stk.push\_back(tem);

cout<<"MOV R"<<rg\_no<<" "<<stk.back()<<endl;

rg\_no++;

stk.pop\_back();

cout<<"MOV R"<<rg\_no<<" "<<stk.back()<<endl;

stk.pop\_back();

cout<<assm[orp]<<" R"<<(rg\_no)<<" R"<<(rg\_no-1)<<endl; cout<<"MOV "<<stk.back()<<" R"<<(rg\_no)<<endl; stk.pop\_back();

cout<<"MOV R"<<rg\_no<<" "<<stk.back()<<endl;

stk.pop\_back();

rg\_no++;

**OUTPUT:** a=b\*f MOV R0 f MOV R1 b

MUL R1 R0

MOV a R1

a=b

MOV R0 b