Compiler Construction

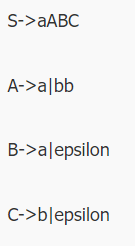
19BCE248

D2

Practical 3

AIM: Write a program to find first( ), and follow() set for each non-terminal of given grammar.

Grammar:



Code:

import java.util.\*;

import java.io.\*;

class prac3 {

    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:

