Practical No: 4

Name : Patel Savankumar P.
Enroll No : 19BCE519
Subject : Compiler Construction

AIM: To implement the Left most derivation removal algorithm.

Code:

```
import java.util.*;

class First_Follow
{
    static class Production
    {
        char variable;
        String production;
        String[] terms;

        // A=Bb|cC|a|$   , # means epsilon , all capitals are Variables
, others are terminals
        Production(String production)
        {
            this.production = production;
            variable = production.charAt(0);
            terms = production.split("=")[1].split("\\|");
            // System.out.println(Arrays.asList(terms));
        }

        public String toString()
        {
            return production+"\n"+variable+"\n"+Arrays.toString(terms);
```

```
        }

    }

    static class Grammar
    {
        int n;
        Production productions[];
        Scanner sc = new Scanner(System.in);
        HashMap<Character,Production> map = new HashMap<>();
        HashMap<Character,ArrayList<Character>> follow = new HashMap<>()
;
        Grammar()
        {
            System.out.print("Enter number of productions : ");
            this.n = sc.nextInt();
            productions = new Production[n];
            for(int i=0;i<n;i++)
            {
                System.out.print("Enter production "+(i+1)+" : ");
                productions[i] = new Production(sc.next());
                map.put(productions[i].variable,productions[i]);
            }
        }

        public void FindFirst(char i,HashSet<Character> ls)
        {
            if(i<'A' || i>'Z')
            {
                ls.add(i);
                return;
            }
            for(String j:map.get(i).terms)
            {
                boolean lastHas = true;
                for(char k:j.toCharArray())
                {
                    if(k==i) continue;
                    HashSet<Character> tmp = new HashSet<Character>();
```

```
                    FindFirst(k,tmp);
                    ls.addAll(tmp);
                    ls.remove('#');
                    lastHas &= tmp.contains('#');
                    if(!tmp.contains('#'))
                    {
                        break;
                    }
                }
                if(lastHas) ls.add('#');
            }
        }

        public void FindFollow(char i,HashSet<Character> ls)
        {
            if(i==productions[0].variable) ls.add('$');
            for(Production j:productions)
            {
                for(String term:j.terms)
                {
                    char arr[] = term.toCharArray();
                    int m = arr.length;
                    for(int k=0;k<m;k++)
                    {
                        if(arr[k]==i)
                        {
                            if(k+1==m)
                            {
                                if(j.variable!=i)
                                {
                                    HashSet<Character> tmp = new HashSet
<Character>();

                                    FindFollow(j.variable,tmp);
                                    ls.addAll(tmp);
                                }
                            }
                            else
                            {
                                boolean lastHas = false;
```

```
                                for(int l=k+1;l<m;l++)
                                {
                                    if(arr[l]==i) continue;
                                    HashSet<Character> tmp = new HashSet
<Character>();
                                    FindFirst(arr[l],tmp);
                                    if(l==m-
1 && tmp.contains('#')) lastHas = true;
                                    ls.addAll(tmp);
                                    if(!tmp.contains('#')) break;
                                }
                                if(lastHas)
                                {
                                    if(j.variable!=i)
                                    {
                                        HashSet<Character> tmp = new Has
hSet<Character>();
                                        FindFollow(j.variable,tmp);
                                        ls.addAll(tmp);
                                    }
                                }
                            }
                        }
                    }
                }
            }

            public String toString()
            {
                String tmp = "";
                tmp+="====================================\n";
                for(Production i:productions)
                {
                    tmp+=i.toString()+"\n";
                    tmp+="====================================\n";
                }
                return tmp;
            }
```

```
    }

    public static void main(String[] args)
    {
        Grammar g = new Grammar();
        // System.out.println(g);

        System.out.println("----------------------------");
        for(Production i:g.productions)
        {
            HashSet<Character> ls = new HashSet<Character>();
            g.FindFirst(i.variable,ls);
            System.out.println("First of ("+i.variable+") : "+ls);
            System.out.println("----------------------------");
        }
        System.out.println();
        for(int i=0;i<g.n;i++)
        {
            HashSet<Character> ls = new HashSet<Character>();
            g.FindFollow(g.productions[i].variable, ls);
            ls.remove('#');
            System.out.println("Follow of ("+g.productions[i].variable+"
) : "+ls);
            // g.follow.put(g.productions[i].variable,ls);
            System.out.println("----------------------------");
        }

    }
}

/*
    5
    E=TR
    R=+TR|#
    T=FY
    Y=*FY|#
    F=(E)|i

    5
```

```
    S=ABCD
    A=b|#
    B=c
    C=d
    D=e

    Enter number of productions : 6
    Enter production 1 : S=aBDh
    Enter production 2 : B=cC
    Enter production 3 : C=bC|#
    Enter production 4 : D=EF
    Enter production 5 : E=g|#
    Enter production 6 : F=f|#



*/
```

Output:

```
PS E:\Semester 7\CC\Lab> java LeftRecursionRemoval
Enter terminals: (comma-separated)
(,),i,+,*,#
Enter non-terminals: (comma-separated)
E,T,R,Y,F
----> ENTER # for epsilon <----
Enter production rule for E:
TR
Enter production rule for T:
FY
Enter production rule for R:
*FY|#
Enter production rule for Y:
+TR|#
Enter production rule for F:
(E)|i
R -> *FY | # | FY | # | *FY
E -> TR
Y -> +TR | #
T -> FY
F -> (E) | i
PS E:\Semester 7\CC\Lab>
```

Conclusion:

From this practical I learned how to Implement a program to remove left most derivation for given productions in Java.