**Compiler Design(18CSC304J)**

**Experiment 5**

**LEFT FACTORING**

Harsh Goel

RA1811003010185

**Aim:** To study and implement Left Factoring

**Language: C**

**Procedure:**

1. Start
2. Ask the user to enter the set of productions
3. Check for common symbols in the given set of productions by comparing with:

A->aB1|aB2

1. If found, replace the particular productions with:

A->aA’

A’->B1 | B2|ɛ

1. Display the output
2. Exit

**Theory:**

Left factoring transforms the grammar to make it useful for top-down parsers. In this technique, we make one production for each common prefixes and the rest of the derivation is added by new productions.

**Code Snippet:**

  #include<stdio.h>

  #include<string.h>

  int main()

  {

       char gram[20],part1[20],part2[20],modifiedGram[20],newGram[20],tempGram[20];

       int i,j=0,k=0,l=0,pos;

       printf("Enter Production : A->");

       gets(gram);   // input

       for(i=0;gram[i]!='|';i++,j++)

            part1[j]=gram[i];   // divide

       part1[j]='\0';   //eol

       for(j=++i,i=0;gram[j]!='\0';j++,i++)

            part2[i]=gram[j];   // divide

       part2[i]='\0';   //eol

       for(i=0;i<strlen(part1)||i<strlen(part2);i++)  //loop

       {

            if(part1[i]==part2[i])

            {

                 modifiedGram[k]=part1[i];

                 k++;

                 pos=i+1;

            }

       }

       for(i=pos,j=0;part1[i]!='\0';i++,j++){

            newGram[j]=part1[i];

       }

       newGram[j++]='|';

       for(i=pos;part2[i]!='\0';i++,j++){

            newGram[j]=part2[i];

       }

       modifiedGram[k]='X';

       modifiedGram[++k]='\0';

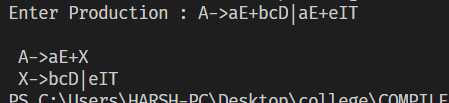
       newGram[j]='\0';

       printf("\n A->%s",modifiedGram);

       printf("\n X->%s\n",newGram);

 }

**Output Screenshots:**



**Result:**

The code was successfully implemented in C and output was recorded. Hence, A program for implementation Of Left Factoring was compiled and run successfully