# IMPLEMENTATION OF LL(1) PARSER

**By**
**ROHAN KUNDU (53)**
**UPASANA MANDAL (54)**
**MAHARSHI BANERJEE THAKURTA (5)**
**SANJU DUTTA (38)**
**SOURIK KABIRAJ (50)**
**SANJAY DUTTA (37)**
**ARCHISMAN DAS (30)**
**KRISHNENDU DAS (31)**

**Under the esteemed guidance of**
Dr. Mousumi Dutt

## TABLE OF CONTENTS

## INTRODUCTION:-

A top-down parser that uses a one-token lookahead is called an LL(1) parser.

- The first L indicates that the input is read from left to right.
- The second L says that it produces a left-to-right derivation.
- And the 1 says that it uses one lookahead token i.e one lookahead symbol is checked at a time.

**Construction of LL(1) Parsing Table:**

To construct the Parsing table, we have two functions:

**1:** First()**:** If there is a variable, and from that variable if we try to drive all the strings then the beginning *Terminal Symbol* is called the first.

**2:** Follow()**:** What is the *Terminal Symbol* which follows a variable in the process of derivation.

Now, after computing the First and Follow set for each *Non-Terminal symbol* we have to construct the Parsing table. In the table Rows will contain the Non-Terminals and the column will contain the Terminal Symbols.

All the **Null Productions** of the Grammars will go under the Follow elements and the remaining productions will lie under the elements of First set.

## OBJECTIVE:-

LL(1) grammars, are of great practical interest, as parsers for these grammars are easy to construct, and many computer languages are designed to be LL(1) for this reason.In this mini-project,with the help of LL(1) Parser, it will be thoroughly implemented  whether a given input string is accepted or rejected.

## CODE:-

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

void followfirst(char , int , int);
void findfirst(char , int , int);
void follow(char c);

int count,n=0;
char calc_first[10][100];
char calc_follow[10][100];
int m=0;
char production[10][10], first[10];
char f[10];
int k;
char ck;
int e;

int main(int argc,char **argv)
{
        int jm=0;
        int km=0;
        int i,choice;
        char c,ch;
        printf("How many productions ? :");
        scanf("%d",&count);
        printf("\nEnter %d productions in form A=B where A and B are
grammar symbols :\n\n",count);
        for(i=0;i<count;i++)
        {
                scanf("%s%c",production[i],&ch);
        }
        int kay;
        char done[count];
        int ptr = -1;
        for(k=0;k<count;k++){
                for(kay=0;kay<100;kay++){
                        calc_first[k][kay] = '!';
                }
        }
        int point1 = 0,point2,xxx;
        for(k=0;k<count;k++)
        {
                c=production[k][0];
                point2 = 0;
                xxx = 0;
                for(kay = 0; kay <= ptr; kay++)
                        if(c == done[kay])
                                xxx = 1;
                if (xxx == 1)
                        continue;
                findfirst(c,0,0);
                ptr+=1;
                done[ptr] = c;
                printf("\n First(%c)= { ",c);
                calc_first[point1][point2++] = c;
```

```
    for(i=0+jm;i<n;i++){
        int lark = 0,chk = 0;
        for(lark=0;lark<point2;lark++){
            if (first[i] == calc_first[point1][lark]){
                chk = 1;
                break;
            }
        }
        if(chk == 0){
            printf("%c, ",first[i]);
            calc_first[point1][point2++] = first[i];
        }
    }
    printf(" }\n");
    jm=n;
    point1++;
}
printf("\n");
printf("--------------------------------------------------\n\n");
char donee[count];
ptr = -1;
for(k=0;k<count;k++){
    for(kay=0;kay<100;kay++){
        calc_follow[k][kay] = '!';
    }
}
point1 = 0;
int land = 0;
for(e=0;e<count;e++)
{
    ck=production[e][0];
    point2 = 0;
    xxx = 0;
    for(kay = 0; kay <= ptr; kay++)
```

```
            if(ck == donee[kay])
                xxx = 1;
        if (xxx == 1)
            continue;
        land += 1;
        follow(ck);
        ptr+=1;
        donee[ptr] = ck;
        printf(" Follow(%c) = { ",ck);
        calc_follow[point1][point2++] = ck;
        for(i=0+km;i<m;i++){
            int lark = 0,chk = 0;
            for(lark=0;lark<point2;lark++){
                if (f[i] == calc_follow[point1][lark]){
                    chk = 1;
                    break;
                }
            }
            if(chk == 0){
                printf("%c, ",f[i]);
                calc_follow[point1][point2++] = f[i];
            }
        }
        printf(" }\n\n");
        km=m;
        point1++;
    }
    char ter[10];
    for(k=0;k<10;k++){
        ter[k] = '!';
    }
    int ap,vp,sid = 0;
    for(k=0;k<count;k++){
        for(kay=0;kay<count;kay++){
```

```
                        if(!isupper(production[k][kay]) &&
production[k][kay]!= '#' && production[k][kay] != '=' &&
production[k][kay] != '\0'){
                                vp = 0;
                                for(ap = 0;ap < sid; ap++){
                                        if(production[k][kay] == ter[ap]){
                                                vp = 1;
                                                break;
                                        }
                                }
                                if(vp == 0){
                                        ter[sid] = production[k][kay];
                                        sid ++;
                                }
                        }
                }
        }
        ter[sid] = '$';
        sid++;
        printf("\n\t\t\t\t\t\t\t The LL(1) Parsing Table for the above
grammer :-");

printf("\n\t\t\t\t\t\t\t^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^\n");

printf("\n\t\t\t============================================
=======================================================
=============================\n");
        printf("\t\t\t\t|\t");
        for(ap = 0;ap < sid; ap++){
                printf("%c\t\t",ter[ap]);
        }

printf("\n\t\t\t============================================
=================================================
==========================\n");
    char first_prod[count][sid];
    for(ap=0;ap<count;ap++){
        int destiny = 0;
        k = 2;
        int ct = 0;
        char tem[100];
        while(production[ap][k] != '\0'){
            if(!isupper(production[ap][k])){
                tem[ct++] = production[ap][k];
                tem[ct++] = '_';
                tem[ct++] = '\0';
                k++;
                break;
            }
            else{
                int zap=0;
                int tuna = 0;
                for(zap=0;zap<count;zap++){
                    if(calc_first[zap][0] ==
production[ap][k]){
                        for(tuna=1;tuna<100;tuna++){
                            if(calc_first[zap][tuna] != '!'){
                                tem[ct++] =
calc_first[zap][tuna];
                            }
                            else
                                break;
                        }
                    break;
                    }
                }
                tem[ct++] = '_';
                }
                k++;
        }
        int zap = 0,tuna;
        for(tuna = 0;tuna<ct;tuna++){
                if(tem[tuna] == '#'){
                        zap = 1;
                }
                else if(tem[tuna] == '_'){
                        if(zap == 1){
                                zap = 0;
                        }
                        else
                                break;
                }
                else{
                        first_prod[ap][destiny++] = tem[tuna];
                }
        }
}
char table[land][sid+1];
ptr = -1;
for(ap = 0; ap < land ; ap++){
        for(kay = 0; kay < (sid + 1) ; kay++){
                table[ap][kay] = '!';
        }
}
for(ap = 0; ap < count ; ap++){
        ck = production[ap][0];
        xxx = 0;
        for(kay = 0; kay <= ptr; kay++)
                if(ck == table[kay][0])
                        xxx = 1;
        if (xxx == 1)
```

```
                continue;
        else{
                ptr = ptr + 1;
                table[ptr][0] = ck;
        }
}
for(ap = 0; ap < count ; ap++){
        int tuna = 0;
        while(first_prod[ap][tuna] != '\0'){
                int to,ni=0;
                for(to=0;to<sid;to++){
                        if(first_prod[ap][tuna] == ter[to]){
                                ni = 1;
                        }
                }
                if(ni == 1){
                        char xz = production[ap][0];
                        int cz=0;
                        while(table[cz][0] != xz){
                                cz = cz + 1;
                        }
                        int vz=0;
                        while(ter[vz] != first_prod[ap][tuna]){
                                vz = vz + 1;
                        }
                        table[cz][vz+1] = (char)(ap + 65);
                }
                tuna++;
        }
}
for(k=0;k<sid;k++){
        for(kay=0;kay<100;kay++){
                if(calc_first[k][kay] == '!'){
                        break;
```

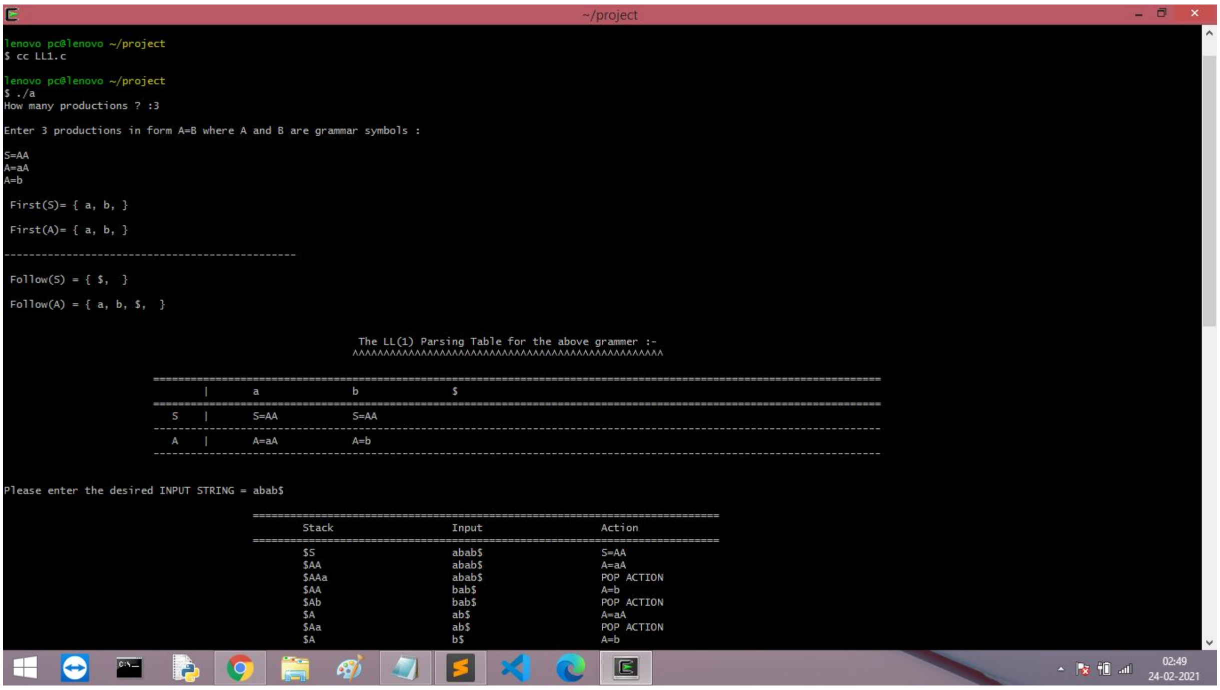
```
				}
				else if(calc_first[k][kay] == '#'){
					int fz = 1;
					while(calc_follow[k][fz] != '!'){
						char xz = production[k][0];
						int cz=0;
						while(table[cz][0] != xz){
							cz = cz + 1;
						}
						int vz=0;
						while(ter[vz] != calc_follow[k][fz]){
							vz = vz + 1;
						}
						table[k][vz+1] = '#';
						fz++;
					}
					break;
				}
			}
		}
		for(ap = 0; ap < land ; ap++){
			printf("\t\t\t   %c\t|\t",table[ap][0]);
			for(kay = 1; kay < (sid + 1) ; kay++){
				if(table[ap][kay] == '!')
					printf("\t\t");
				else if(table[ap][kay] == '#')
					printf("%c=#\t\t",table[ap][0]);
				else{
					int mum = (int)(table[ap][kay]);
					mum -= 65;
					printf("%s\t\t",production[mum]);
				}
			}
			printf("\n");
```

```
printf("\t\t\t-------------------------------------------------------------
----------------------------------------");
            printf("\n");
        }
        int j;
        printf("\n\nPlease enter the desired INPUT STRING = ");
        char input[100];
        scanf("%s%c",input,&ch);

printf("\n\t\t\t\t\t=================================
================================\n");
        printf("\t\t\t\t\t\tStack\t\t\tInput\t\t\tAction");

printf("\n\t\t\t\t\t=================================
================================\n");
        int i_ptr = 0,s_ptr = 1;
        char stack[100];
        stack[0] = '$';
        stack[1] = table[0][0];
        while(s_ptr != -1){
            printf("\t\t\t\t\t\t");
            int vamp = 0;
            for(vamp=0;vamp<=s_ptr;vamp++){
                printf("%c",stack[vamp]);
            }
            printf("\t\t\t");
            vamp = i_ptr;
            while(input[vamp] != '\0'){
                printf("%c",input[vamp]);
                vamp++;
            }
            printf("\t\t\t");
            char her = input[i_ptr];
        char him = stack[s_ptr];
        s_ptr--;
        if(!isupper(him)){
            if(her == him){
                i_ptr++;
                printf("POP ACTION\n");
            }
            else{
                printf("\nString Not Accepted by LL(1) Parser
!!\n");
                exit(0);
            }
        }
        else{
            for(i=0;i<sid;i++){
                if(ter[i] == her)
                    break;
            }
            char produ[100];
            for(j=0;j<land;j++){
                if(him == table[j][0]){
                    if (table[j][i+1] == '#'){
                        printf("%c=#\n",table[j][0]);
                        produ[0] = '#';
                        produ[1] = '\0';
                    }
                    else if(table[j][i+1] != '!'){
                        int mum = (int)(table[j][i+1]);
                        mum -= 65;
                        strcpy(produ,production[mum]);
                        printf("%s\n",produ);
                    }
                    else{
                        printf("\nString Not Accepted by
LL(1) Parser !!\n");
                        exit(0);
                    }
                }
            }
            int le = strlen(produ);
            le = le - 1;
            if(le == 0){
                continue;
            }
            for(j=le;j>=2;j--){
                s_ptr++;
                stack[s_ptr] = produ[j];
            }
        }
    }

printf("\n\t\t\t======================================
================================================
============================\n");
    if (input[i_ptr] == '\0'){
        printf("\t\t\t\t\t\t\t\tYOUR STRING HAS BEEN
ACCEPTED !!\n");
    }
    else
        printf("\n\t\t\t\t\t\t\t\tYOUR STRING HAS BEEN
REJECTED !!\n");

printf("\t\t\t========================================
================================================
==========================\n");
}

void follow(char c)
```

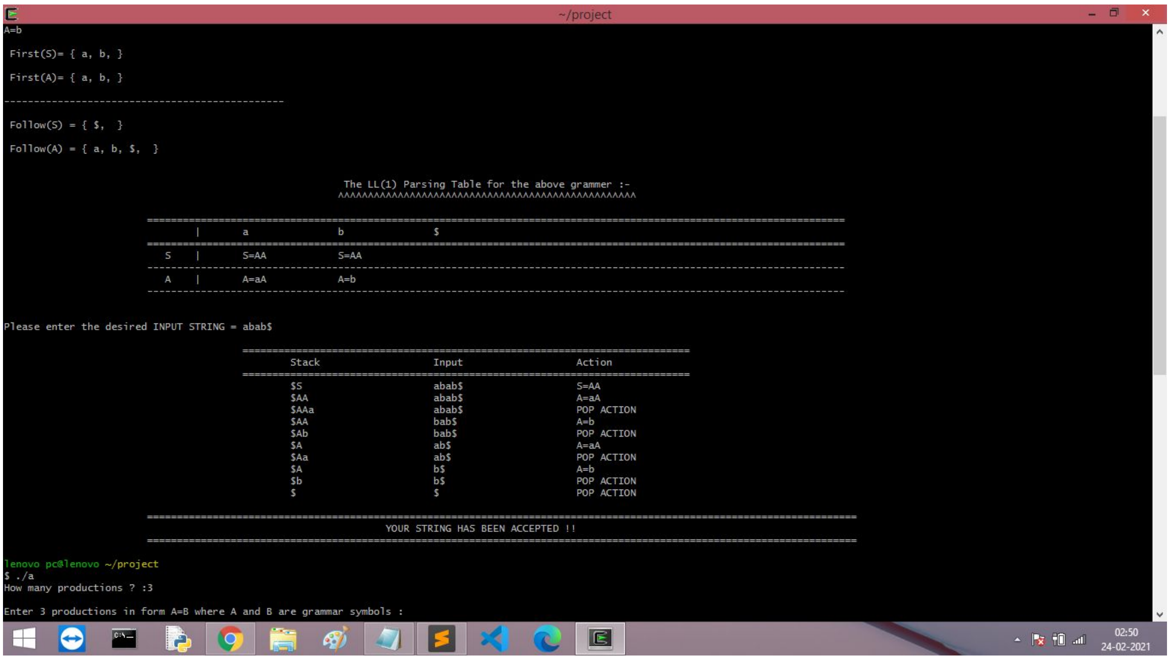
```
{
    int i ,j;
    if(production[0][0]==c){
        f[m++]='$';
    }
    for(i=0;i<10;i++)
    {
        for(j=2;j<10;j++)
        {
            if(production[i][j]==c)
            {
            if(production[i][j+1]!='\0'){
                    followfirst(production[i][j+1],i,(j+2));
                }
            if(production[i][j+1]=='\0'&&c!=production[i][0]){
                follow(production[i][0]);
                }
            }
        }
    }
}

void findfirst(char c ,int q1 , int q2)
{
    int j;
    if(!(isupper(c))){
        first[n++]=c;
    }
    for(j=0;j<count;j++)
    {
        if(production[j][0]==c)
        {
            if(production[j][2]=='#'){
                if(production[q1][q2] == '\0')
```

```
                    first[n++]='#';
                else if(production[q1][q2] != '\0' && (q1 != 0 || q2 != 0))
                {
                    findfirst(production[q1][q2], q1, (q2+1));
                }
                else
                    first[n++]='#';
            }
            else if(!isupper(production[j][2])){
                first[n++]=production[j][2];
            }
            else {
                findfirst(production[j][2], j, 3);
            }
        }
    }
}

void followfirst(char c, int c1 , int c2)
{
  int k;
  if(!(isupper(c)))
        f[m++]=c;
    else{
        int i=0,j=1;
        for(i=0;i<count;i++)
        {
            if(calc_first[i][0] == c)
                break;
        }
        while(calc_first[i][j] != '!')
        {
            if(calc_first[i][j] != '#'){
                f[m++] = calc_first[i][j];
            }
            else{
                if(production[c1][c2] == '\0'){
                    follow(production[c1][0]);
                }
                else{
                    followfirst(production[c1][c2],c1,c2+1);
                }
            }
            j++;
        }
    }
}
```

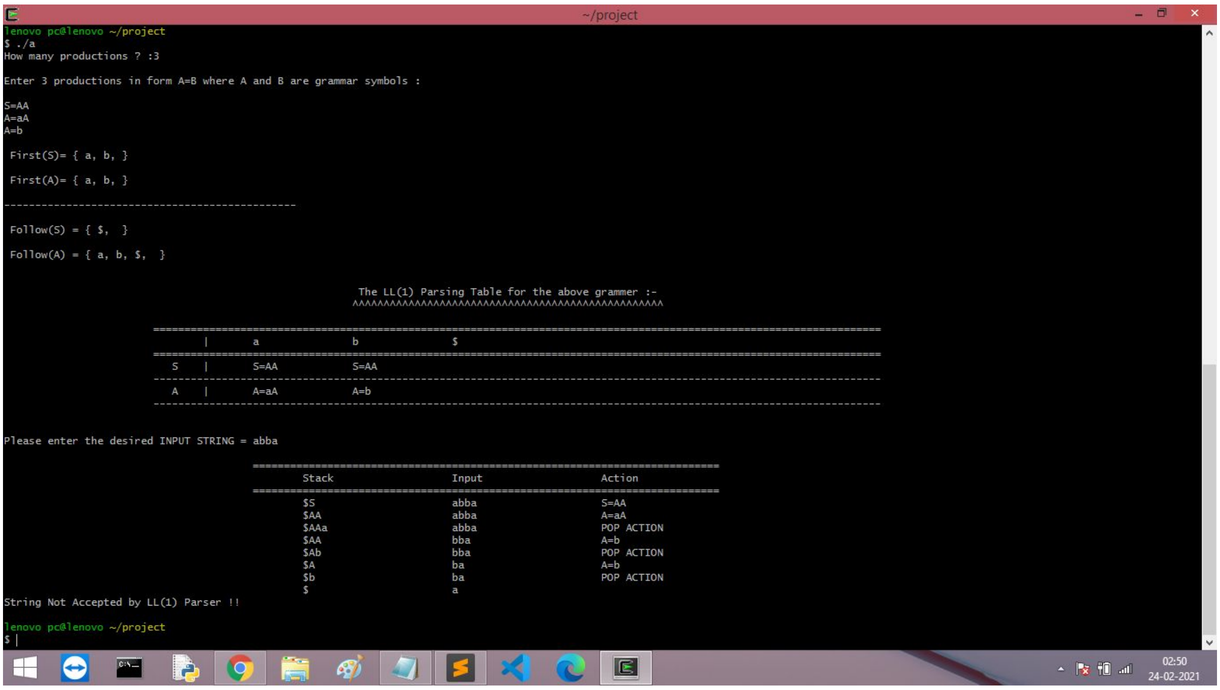
# OUTPUT:-



```
lenovo pc@lenovo ~/project
$ cc LL1.c

lenovo pc@lenovo ~/project
$ ./a
How many productions ? :3

Enter 3 productions in form A=B where A and B are grammar symbols :

S=AA
A=aA
A=b

 First(S)= { a, b, }

 First(A)= { a, b, }

-----------------------------------------------


 Follow(S) = { $,  }

 Follow(A) = { a, b, $,  }


                                                         The LL(1) Parsing Table for the above grammer :-
                                                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^


                        =======================================================================================================================
                                |       a               b               $
                        =======================================================================================================================
                           S    |       S=AA            S=AA
                        -----------------------------------------------------------------------------------------------------------------------
                           A    |       A=aA            A=b
                        -----------------------------------------------------------------------------------------------------------------------


Please enter the desired INPUT STRING = abab$

                                        ===========================================================================
                                                Stack                   Input                   Action
                                        ===========================================================================
                                                $S                      abab$                   S=AA
                                                $AA                     abab$                   A=aA
                                                $AAa                    abab$                   POP ACTION
                                                $AA                     bab$                    A=b
                                                $Ab                     bab$                    POP ACTION
                                                $A                      ab$                     A=aA
                                                $Aa                     ab$                     POP ACTION
                                                $A                      b$                      A=b
```



```
A=b

 First(S)= { a, b, }

 First(A)= { a, b, }

-----------------------------------------------

 Follow(S) = { $,  }

 Follow(A) = { a, b, $,  }


                                               The LL(1) Parsing Table for the above grammer :-
                                              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

                       =======================================================================================================================
                               |       a               b               $
                       =======================================================================================================================
                         S     |       S=AA            S=AA
                       -----------------------------------------------------------------------------------------------------------------------
                         A     |       A=aA            A=b
                       -----------------------------------------------------------------------------------------------------------------------


Please enter the desired INPUT STRING = abab$

                                       ============================================================================
                                               Stack                   Input                   Action
                                       ============================================================================
                                               $S                      abab$                   S=AA
                                               $AA                     abab$                   A=aA
                                               $AAa                    abab$                   POP ACTION
                                               $AA                     bab$                    A=b
                                               $Ab                     bab$                    POP ACTION
                                               $A                      ab$                     A=aA
                                               $Aa                     ab$                     POP ACTION
                                               $A                      b$                      A=b
                                               $b                      b$                      POP ACTION
                                               $                       $                       POP ACTION

                       ========================================================================================================================
                                                                  YOUR STRING HAS BEEN ACCEPTED !!
                       ========================================================================================================================

lenovo pc@lenovo ~/project
$ ./a
How many productions ? :3

Enter 3 productions in form A=B where A and B are grammar symbols :
```



```
lenovo pc@lenovo ~/project
$ ./a
How many productions ? :3

Enter 3 productions in form A=B where A and B are grammar symbols :

S=AA
A=aA
A=b

 First(S)= { a, b, }

 First(A)= { a, b, }

-----------------------------------------------

 Follow(S) = { $,  }

 Follow(A) = { a, b, $,  }


                                               The LL(1) Parsing Table for the above grammer :-
                                              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

                       =======================================================================================================================
                               |       a               b               $
                       =======================================================================================================================
                         S     |       S=AA            S=AA
                       -----------------------------------------------------------------------------------------------------------------------
                         A     |       A=aA            A=b
                       -----------------------------------------------------------------------------------------------------------------------


Please enter the desired INPUT STRING = abba

                                       ============================================================================
                                               Stack                   Input                   Action
                                       ============================================================================
                                               $S                      abba                    S=AA
                                               $AA                     abba                    A=aA
                                               $AAa                    abba                    POP ACTION
                                               $AA                     bba                     A=b
                                               $Ab                     bba                     POP ACTION
                                               $A                      ba                      A=b
                                               $b                      ba                      POP ACTION
                                               $                       a
String Not Accepted by LL(1) Parser !!

lenovo pc@lenovo ~/project
$ |
```

# CONCLUSION:-

With this implementation it has been understood that how a given input string is checked by the LL(1) parser and whether they are accepted or not and here, two input strings have been taken as input and as a result, the first one has been accepted and the second one hasn’t been accepted by the LL(1) Parser.