# **Project Instructions**

# **Objective:** To write a Recursive Descent parser and implement it in C

## **OVERVIEW:**

Suppose we need to construct a Recursive Descent Parser for the following grammar:  
G = ({S,L}, {(, ), a, , }, {S →(L) | a; L →L, S | S } , S)

The above construct is representing the fact that:

1. Non terminals are S, L
2. Terminal symbols are { **(**, **)**, **a**, **,**}
3. Production Rules:

S →(L) |a

L →L,S | S

1. Start symbol is S

We can use the above grammar to determine if the strings (a,(a,a)) and (a,(a,(a,a),(a,a))) are acceptable in the grammar.

Note that the grammar has left recursion and we need to remove it. The following becomes the new grammar:

S →(L) | a

L →S L’

L’ →, SL’ | Ɛ

Below is the C code that can be used to implement a Recursive Descent parser for the above grammar

**/\* Recursive Descent Parser for the Expression Grammar:**

**S → (L) |a**

**L' →,SL'|ε**

**L → SL'**

**Valid inputs: (a,(a,a)) and (a,(a,(a,a),(a,a)))**

**Invalid inputs:(aa,a)**

**\*/**

**#include <stdio.h>**

**#include <string.h>**

**int S(), Ldash(), L();**

**char \*ip;**

**char string[50];**

**int main()**

**{**

**printf("Enter the string\n");**

**scanf("%s", string);**

**ip = string;**

**printf("\n\nInput\t\tAction\n ------------------------------\n");**

**if (S())**

**{**

**printf("\n------------------------------------------------\n");**

**printf("\n String is successfully parsed\n");**

**}**

**else**

**{**

**printf("\n ------------------------------------------------\n");**

**printf("Error in parsing string\n");**

**}**

**}**

**int L()**

**{**

**printf("%s\t\tL →SL' \n", ip);**

**if (S())**

**{**

**if (Ldash())**

**{**

**return 1;**

**}**

**else**

**return 0;**

**}**

**else**

**return 0;**

**}**

**int Ldash()**

**{**

**if (\*ip == ',')**

**{**

**printf("%s\t\tL' →, SL' \n", ip);**

**ip++;**

**if (S())**

**{**

**if (Ldash())**

**{**

**return 1;**

**}**

**else**

**return 0;**

**}**

**else**

**return 0;**

**}**

**else**

**{**

**printf("%s\t\tL' →ε \n", ip);**

**return 1;**

**}**

**}**

**int S()**

**{**

**if (\*ip == '(')**

**{**

**printf("%s\t\tS →(L) \n", ip);**

**ip++;**

**if (L())**

**{**

**if (\*ip == ')')**

**{**

**ip++;**

**return 1;**

**}**

**else**

**return 0;**

**}**

**else**

**return 0;**

**}**

**else if (\*ip == 'a')**

**{**

**ip++;**

**printf("%s\t\tS →a \n", ip);**

**return 1;**

**}**

**else**

**return 0;**

**}**

What you need to do:

1. Type that code up and run it to make sure that it is doing what it is supposed to do. Provide the strings in the header of the program when prompted.
2. Use the logic in that program to design your own Recursive Descent parser in C for the following grammar:

S →aAB

A →Abc | b

B →d

1. Make sure that you choose at least two strings that pass the grammar and one string that does not pass the grammar just like I did in the model program.
2. Submit both the code and a screen shot of your results for the choice of strings that you provided.