Recursive Decent Parser

1. **Grammar -**

E → TE’
E’→ +TE’ | ϵ
T → FT’
T’→ *FT’ | ϵ
F → (E) | i

Program -

```
#include<stdio.h>
#include<string.h>
int E(),Edash(),T(),Tdash(),F();
char *ip;
char string[50];
int main()
{
printf("Enter the string\n");
scanf("%s",string);
ip=string;
printf("\n\nInput\tAction\n--------------------------------\n");

if(E() && *ip=='\0'){
printf("\n--------------------------------\n");
printf("\n String is successfully parsed\n");
}
else{
printf("\n--------------------------------\n");
printf("Error in parsing String\n");
}
}
int E()
{
printf("%s\tE->TE' \n",ip);
if(T())
{
if(Edash())
{
return 1;
```

```
}
else
return 0;
}
else
return 0;
}
int Edash()
{
if(*ip=='+')
{
printf("%s\tE'->+TE' \n",ip);
ip++;
if(T())
{
if(Edash())
{
return 1;
}
else
return 0;
}
else
return 0;
}
else
{
printf("%s\tE'->^ \n",ip);
return 1;
}
}
int T()
{
printf("%s\tT->FT' \n",ip);
if(F())
{

if(Tdash())
{
return 1;
```

```
}
else
return 0;
}
else
return 0;
}
int Tdash()
{
if(*ip=='*')
{
printf("%s\tT'->*FT' \n",ip);
ip++;
if(F())
{
if(Tdash())
{
return 1;
}
else
return 0;
}
else
return 0;
}
else
{
printf("%s\tT'->^ \n",ip);
return 1;
}
}
int F()
{
if(*ip=='(')
{
printf("%s\tF->(E) \n",ip);
ip++;
if(E())
{
if(*ip==')')
```

```
{
ip++;
return 0;
}
else
return 0;
}
else
return 0;
}

else if(*ip=='i')
{
ip++;
printf("%s\tF->id \n",ip);
return 1;
}
else
return 0;
}
```

Test Cases -

i+i*i	String is successfully parsed
i+i	String is successfully parsed
i*i	String is successfully parsed
i*i+i*i+i	String is successfully parsed
i+*+i	Error in parsing String
i+i*	Error in parsing String

2. **Grammar -**

S -> (L) | a
L -> ST
T -> ,ST | ε

Program -

```
i = 0
def S():
    global i
    if (s[i] == '('):
        print(f"{s[i:]} \t S -> (L)\n")
        i = i + 1
        if (L()):
            if (s[i] == ')'):
                i = i + 1
                return 1
            else:
                return 0
        else:
            return 0
    elif (s[i] == 'a'):
        print(f"{s[i:]} \t S -> a\n")
        i = i + 1
        return 1
    else:
        return 0
def L():
    global i
    print(f"{s[i:]} \t L -> ST\n")
    if (S()):
        if (T()):
            return 1
        else:
            return 0
    else:
        return 0
def T():
    global i
```

```
    if (s[i] == ','):
        print(f"{s[i:]} \t T -> ,ST\n")
        i = i + 1
        if (S()):
            if (T()):
                return 1
            else:
                return 0
        else:
            return 0
    else:
        print(f"{s[i:]} \t T -> ^\n")
        return 1

s = input("Enter a string: ")
print("Input \t Action\n")
if (S() and i == len(s)):
    print("String is parsed successfully")
else:
    print("Error in parsing")
```

Test Cases -

(a,(a,a))	String is parsed successfully
(a,((a,a),(a,a)))	String is parsed successfully
(a,a))	Error in parsing
(a,(a,a)))	Error in parsing