UCS 1602 - Compiler Design

Exercise 4 - Implementation of Recursive Descent Parser in C

Name:	Mahesh Bharadwaj K
Reg No:	185001089
Semester:	VI
Date:	March 1, 2021

Aim:

To implement Recursive Descent Parser using C program

Program

Header file

```
typedef char string[30];

typedef enum
{
    FAIL,
    SUCCESS
} Result;

void indent(const int level)
{
    for (int i = 0; i < level; i++)
        printf("    ");
}

Result T(string, int *, int);
Result E(string, int *, int);
Result Eprime(string, int *, int);
Result Tprime(string, int *, int);
Result F(string, int *, int);

Result recursiveParser(string input)
{
    int index = 0;
    return E(input, &index, 0);
}

Result E(string input, int *idx, int depth)
{
    if (!input[*idx])
        return FAIL;

    indent(depth);
    printf("E()\n");
    Result result = T(input, idx, depth + 1);

    if (result == FAIL)
        return FAIL;
```

```
    result = Eprime(input, idx, depth + 1);
    return result;
}

Result Eprime(string input, int *idx, int depth)
{
    if (!input[*idx])
        return FAIL;
    indent(depth);
    printf("E'()\n");
    Result result;
    int current_idx = *idx;
    if (input[*idx] == '+')
    {
        indent(depth);
        printf("Non terminal \'+\' found!\n");
        (*idx) = (*idx) + 1;
        result = T(input, idx, depth + 1);
        if (result == FAIL)
            return FAIL;
        result = Eprime(input, idx, depth + 1);
        if (result == SUCCESS)
            return SUCCESS;
    }
    *idx = current_idx;
    indent(depth);
    printf("Non terminal \'+\' found!\n");
    return SUCCESS;
}

Result T(string input, int *idx, int depth)
{
    if (!input[*idx])
        return FAIL;
    indent(depth);
    printf("T()\n");
    Result result = F(input, idx, depth + 1);
    if (result == FAIL)
        return FAIL;
    result = Tprime(input, idx, depth + 1);
    return result;
}

Result Tprime(string input, int *idx, int depth)
{
    if (!input[*idx])
        return FAIL;
    indent(depth);
    printf("T'()\n");
    Result result;
    int current_idx = *idx;
    if (input[*idx] == '*')
    {
        indent(depth);
        printf("Non terminal \'*\' found!\n");
        (*idx) = (*idx) + 1;
        result = F(input, idx, depth + 1);
        if (result == FAIL)
            return FAIL;
        result = Tprime(input, idx, depth + 1);
```

```
        if (result == SUCCESS)
            return SUCCESS;
    }
    *idx = current_idx;
    indent(depth);
    printf("Non terminal \'*\' not found!\n");
    return SUCCESS;
}

Result F(string input, int *idx, int depth)
{
    if (!input[*idx])
        return FAIL;
    indent(depth);
    printf("F()\n");
    int current_idx = *idx;
    if (input[*idx] == 'i' && input[*idx + 1] == 'd')
    {
        (*idx) += 2;
        indent(depth);
        printf("Non terminal \'id\' found!\n");
        return SUCCESS;
    }
    else if (input[*idx] == '(')
    {
        indent(depth);
        printf("Non terminal \'(\' found!\n");
        Result result = E(input, idx, depth + 1);
        if (result == FAIL)
            return FAIL;

        if (input[*idx] == ')')
            return SUCCESS;
    }
    return FAIL;
}
```

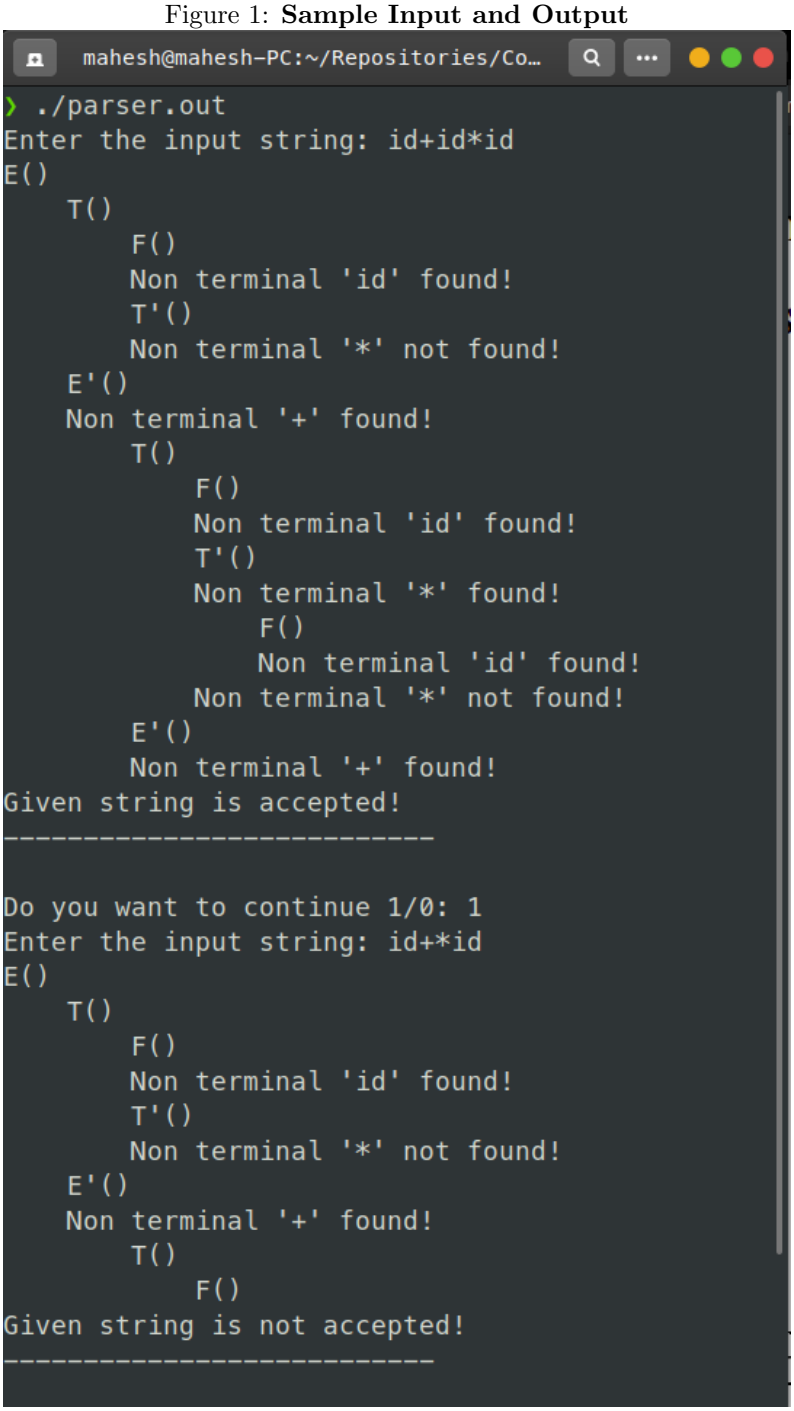
Main Program

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

#include "procedures.h"

void printResult(Result);

int main(void)
{
    string input;
    int opt = -1;
    while (opt != 0)
    {
        printf("Enter the input string: ");
        scanf("%s", input);
        printResult(recursiveParser(input));
        printf("--------------------------\n\n");
        printf("Do you want to continue 1/0: ");
        scanf("%d", &opt);
    }
```

```
}

void printResult(Result result)
{
    if (result == SUCCESS)
        printf("Given string is accepted!\n");
    else
        printf("Given string is not accepted!\n");
}
```

Output

Figure 1: **Sample Input and Output**



Learning Outcomes

1. We learn to identify left recursion
2. We learn to remove left recursion
3. We learn to write procedures given productions of grammar
4. We learn to create recursive descent parser to parse given input strings and check if they belong to the grammar or not.
