Ex-3 M. Rohith
11-09-2024 3122 21 5001 085

SSN COLLEGE OF ENGINEERING, KALAVAKKAM
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UCS2702 - Compiler Design

Programming Assignment-3 Implementation of Recursive Decent Parser

Implement Recursive Decent Parser for the given grammar (left recursion eliminated)

Grammar

E →TE’

E’ →+TE’ | Ɛ

T→ FT’

T’ →*FT’ | Ɛ

F →(E) | id

1. Write the procedures for all the non terminals in the left recursion eliminated grammar.
2. Parse the following input using the above procedures
   (a) id+id*id
   (b) (id+id*id
   (c) id-id

PROGRAM CODE:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

char *input;

char lookahead;

void E();

void E_prime();

void T();

void T_prime();

void F();

void match(char token)

{

    while (isspace(lookahead))

    {

        lookahead = *++input;

    }

    if (lookahead == token)

    {

        lookahead = *++input;

    }

    else

    {

        printf("Error: Expected %c\n", token);

        exit(1);

    }

}

void E()

{

    printf("E -> TE'\n");
    T();
    E_prime();
}
void E_prime()
{
    if (lookahead == '+')
    {
        printf("E' -> +TE'\n");
        match('+');
        T();
        E_prime();
    }
    else
    {
        printf("E' -> epsilon\n");
    }
}
void T()
{
    printf("T -> FT'\n");
    F();
    T_prime();
}
void T_prime()
{
    if (lookahead == '*')
    {
        printf("T' -> *FT'\n");
        match('*');
        F();
        T_prime();
    }
    else
    {
        printf("T' -> epsilon\n");
    }
}
void F()
{
    if (lookahead == '(')
    {
        printf("F -> (E)\n");
        match('(');
        E();
        match(')');
    }
    else if (isalpha(lookahead))
    {
        printf("F -> id\n");
        while (isalnum(lookahead))
        { // Match the entire identifier (id)
            match(lookahead);
        }
    }
    else
    {
        printf("Error: Unexpected token %c\n", lookahead);
        exit(1);
    }
}
void parse(char *inputString)
{
```

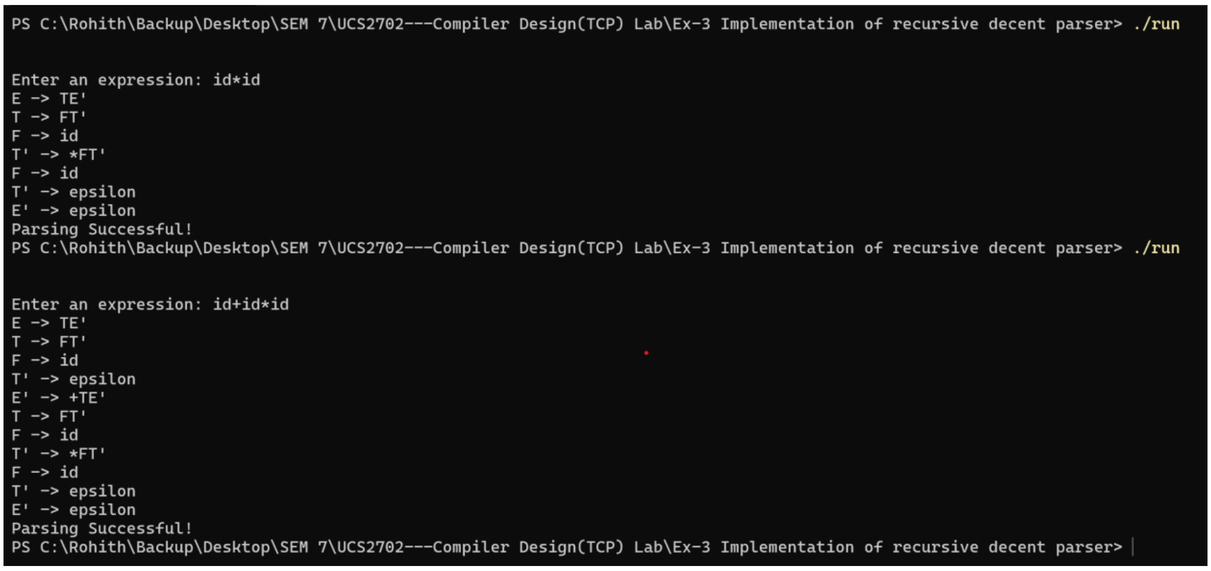
M. Rohith
3122 21 5001 085

```
    input = inputString;
    size_t len = strlen(inputString);
    if (len > 0 && inputString[len - 1] == '\n')
    {
        inputString[len - 1] = '\0';
    }
    lookahead = *input;
    E();
    if (lookahead == '\0')
    {
        printf("Parsing Successful!\n");
    }
    else
    {
        printf("Parsing Failed. Remaining input: %s\n", input);
    }
}
int main()
{
    char inputString[100];
    printf("\n\nEnter an expression: ");
    fgets(inputString, sizeof(inputString), stdin);
    parse(inputString);
    return 0;
}
```

```
gcc RDParser.c -o run
./run
```

Ex-3 M. Rohith
11-09-2024 3122 21 5001 085

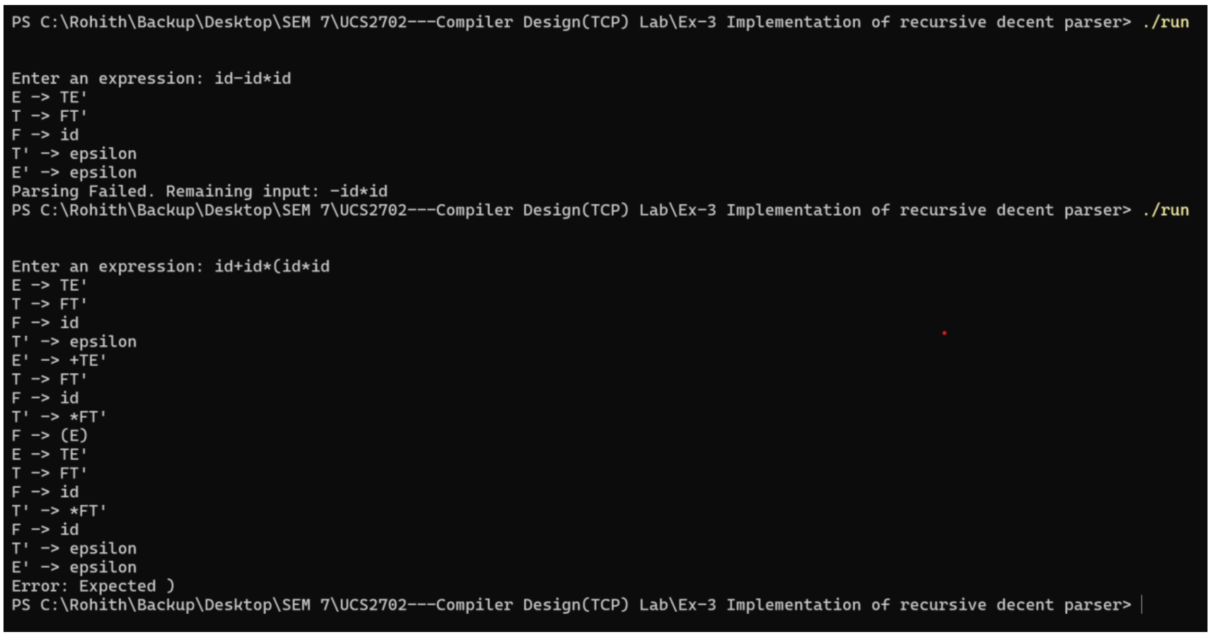
OUTPUT:



```
PS C:\Rohith\Backup\Desktop\SEM 7\UCS2702---Compiler Design(TCP) Lab\Ex-3 Implementation of recursive decent parser> ./run


Enter an expression: id*id
E -> TE'
T -> FT'
F -> id
T' -> *FT'
F -> id
T' -> epsilon
E' -> epsilon
Parsing Successful!
PS C:\Rohith\Backup\Desktop\SEM 7\UCS2702---Compiler Design(TCP) Lab\Ex-3 Implementation of recursive decent parser> ./run


Enter an expression: id+id*id
E -> TE'
T -> FT'
F -> id
T' -> epsilon
E' -> +TE'
T -> FT'
F -> id
T' -> *FT'
F -> id
T' -> epsilon
E' -> epsilon
Parsing Successful!
PS C:\Rohith\Backup\Desktop\SEM 7\UCS2702---Compiler Design(TCP) Lab\Ex-3 Implementation of recursive decent parser>
```



```
PS C:\Rohith\Backup\Desktop\SEM 7\UCS2702---Compiler Design(TCP) Lab\Ex-3 Implementation of recursive decent parser> ./run


Enter an expression: id-id*id
E -> TE'
T -> FT'
F -> id
T' -> epsilon
E' -> epsilon
Parsing Failed. Remaining input: -id*id
PS C:\Rohith\Backup\Desktop\SEM 7\UCS2702---Compiler Design(TCP) Lab\Ex-3 Implementation of recursive decent parser> ./run


Enter an expression: id+id*(id*id
E -> TE'
T -> FT'
F -> id
T' -> epsilon
E' -> +TE'
T -> FT'
F -> id
T' -> *FT'
F -> (E)
E -> TE'
T -> FT'
F -> id
T' -> *FT'
F -> id
T' -> epsilon
E' -> epsilon
Error: Expected )
PS C:\Rohith\Backup\Desktop\SEM 7\UCS2702---Compiler Design(TCP) Lab\Ex-3 Implementation of recursive decent parser>
```