UCS 1602 - Compiler Design

Assignment-4

Recursive Descent Parser Using C

Swetha Saseendran
CSE-C
185001183

Aim:

Write a program in C to construct Recursive Descent Parser for the following grammar which is for arithmetic expression involving + and *. Check the Grammar for left recursion and convert into suitable for this parser. Write recursive functions for every non-terminal. Call the function for start symbol of the Grammar in main().

G: E->E+T|T
T->T*F | F
F->i

Extend this parser to include division, subtraction and parenthesis operators

G: E->E+T|E-T|T
T->T*F | T/F|F
F->(E)|i

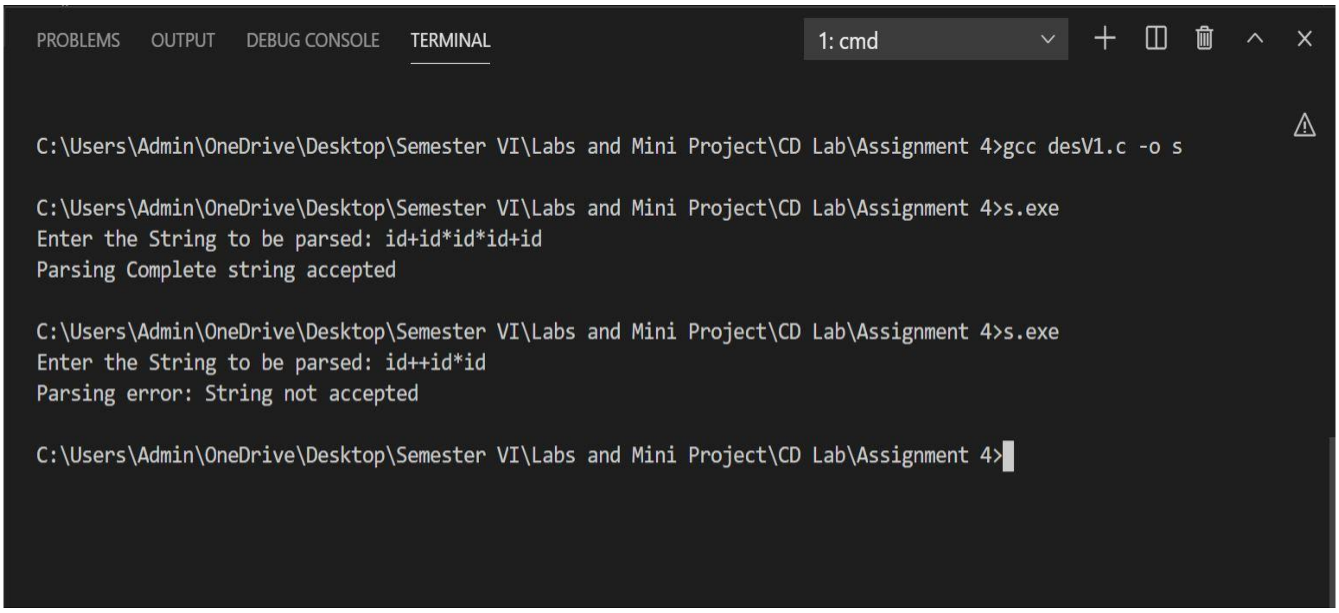
Code: Grammar 1

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

/* STRUCTUERE */
typedef struct
{
    char ip[100];
    int n;
}input;
```

```c
/* FUNCTIONS */
void E(input *in);
void T(input *in);
void Tprime(input *in);
void Eprime(input *in);
void F(input *in);

/* DEFINED FUNCTIONS */
void E(input *in)
{
  T(in);
  Eprime(in);
}

void Eprime(input *in)
{
  if(in->ip[in->n]=='+')
  {
    (in->n)++;
      T(in);
      Eprime(in);
  }
  else
    return;
}

void T(input *in)
{
  F(in);
  Tprime(in);
}

void Tprime(input *in)
{
  if(in->ip[in->n]=='*')
  {
    (in->n)++;
      F(in);
      Tprime(in);
  }
  else
    return;
}
```

```c
void F(input *in)
{
    if(in->ip[in->n]=='i' && in->ip[in->n+1]=='d')
    {
        (in->n)+=2;
        if(in->n==strlen(in->ip))
        {
        printf("Parsing Complete string accepted\n");
        exit(0);
        }
    }
    else
    {
        printf("Parsing error: String not accepted\n");
        exit(0);
    }
}

/* MAIN FUNCTION */
int main()
{
  input *in = malloc(sizeof(input));
  printf("Enter the String to be parsed: ");
  scanf("%s",in->ip);
  in->n = 0;
  E(in);

  if(in->n!=strlen(in->ip))
    printf("Parser Error : String not accepted\n");
}
```

OUTPUT:



```
C:\Users\Admin\OneDrive\Desktop\Semester VI\Labs and Mini Project\CD Lab\Assignment 4>gcc desV1.c -o s

C:\Users\Admin\OneDrive\Desktop\Semester VI\Labs and Mini Project\CD Lab\Assignment 4>s.exe
Enter the String to be parsed: id+id*id*id+id
Parsing Complete string accepted

C:\Users\Admin\OneDrive\Desktop\Semester VI\Labs and Mini Project\CD Lab\Assignment 4>s.exe
Enter the String to be parsed: id++id*id
Parsing error: String not accepted

C:\Users\Admin\OneDrive\Desktop\Semester VI\Labs and Mini Project\CD Lab\Assignment 4>
```

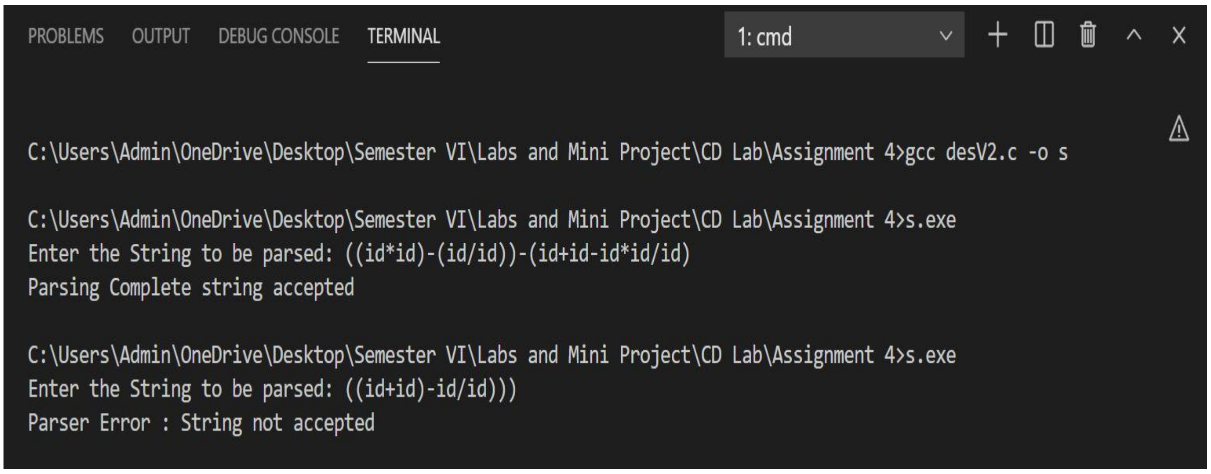
Code: Grammar 2

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

/* STRUCTUERE */
typedef struct
{
    char ip[100];
    int n;
}input;

/* FUNCTIONS */
void E(input *in);
void T(input *in);
void Tprime(input *in);
void Eprime(input *in);
void F(input *in);

/* DEFINED FUNCTIONS */
void E(input *in)
{
  T(in);
  Eprime(in);
}

void Eprime(input *in)
{
  if(in->ip[in->n]=='+')
  {
    (in->n)++;
      T(in);
      Eprime(in);
  }
  else if(in->ip[in->n]=='-')
  {
    (in->n)++;
      T(in);
      Eprime(in);
  }
  else
    return;
}
```

```
void T(input *in)
{
  F(in);
  Tprime(in);
}

void Tprime(input *in)
{
  if(in->ip[in->n]=='*')
  {
    (in->n)++;
      F(in);
      Tprime(in);
  }
  else if(in->ip[in->n]=='/')
  {
    (in->n)++;
      F(in);
      Tprime(in);
  }
  else
    return;
}

void F(input *in)
{
    if(in->ip[in->n]=='(')
    {
        (in->n)++;
        E(in);
        if(in->ip[in->n]==')')
        {
            (in->n)++;
            if(in->n==strlen(in->ip))
            {
                printf("Parsing Complete string accepted\n");
                exit(0);
            }
        }
    }
    else if(in->ip[in->n]=='i' && in->ip[in->n+1]=='d')
    {
        (in->n)+=2;
        if(in->n==strlen(in->ip))
        {
        printf("Parsing Complete string accepted\n");
        exit(0);
        }
```

```c
    }
    else
    {
        printf("Parsing error: String not accepted\n");
        exit(0);
    }
}

/* MAIN FUNCTION */
int main()
{
  input *in = malloc(sizeof(input));
  printf("Enter the String to be parsed: ");
  scanf("%s",in->ip);
  in->n = 0;
  E(in);

  if(in->n!=strlen(in->ip))
    printf("Parser Error : String not accepted\n");
}
```

OUTPUT:



```
C:\Users\Admin\OneDrive\Desktop\Semester VI\Labs and Mini Project\CD Lab\Assignment 4>gcc desV2.c -o s

C:\Users\Admin\OneDrive\Desktop\Semester VI\Labs and Mini Project\CD Lab\Assignment 4>s.exe
Enter the String to be parsed: ((id*id)-(id/id))-(id+id-id*id/id)
Parsing Complete string accepted

C:\Users\Admin\OneDrive\Desktop\Semester VI\Labs and Mini Project\CD Lab\Assignment 4>s.exe
Enter the String to be parsed: ((id+id)-id/id)))
Parser Error : String not accepted
```

LEARNING OUTCOME:

- Understood about the working of a Recursive Descent Parser,
  that the Recursive Descent Parser, being a Top-Down Parser, does not work with Left-Recursive Grammars.
- Understood the need for this type of conversion, as top-down parsers cannot handle left recursive grammars.
- Learnt to describe how automated scanner generators construct a finite automation from regular expression.
- Learnt to select and use appropriate data types and data structures to solve problems.
- Strengthened my knowledge and skills in string operations and to parse each production in input to check if its belongs to the grammar or not.
- Learnt to modularise long code into functions and follow the best practices.

RESULT:

Successfully implemented the code to construct Recursive Descent Parser for the given grammars.