NAME : Upmanyu Jha

ROLL NO. : 8875

TE-COMPUTERS-A (SUB-BATCH : A)

SPCC EXPERIMENT 5 -RDP

Code :

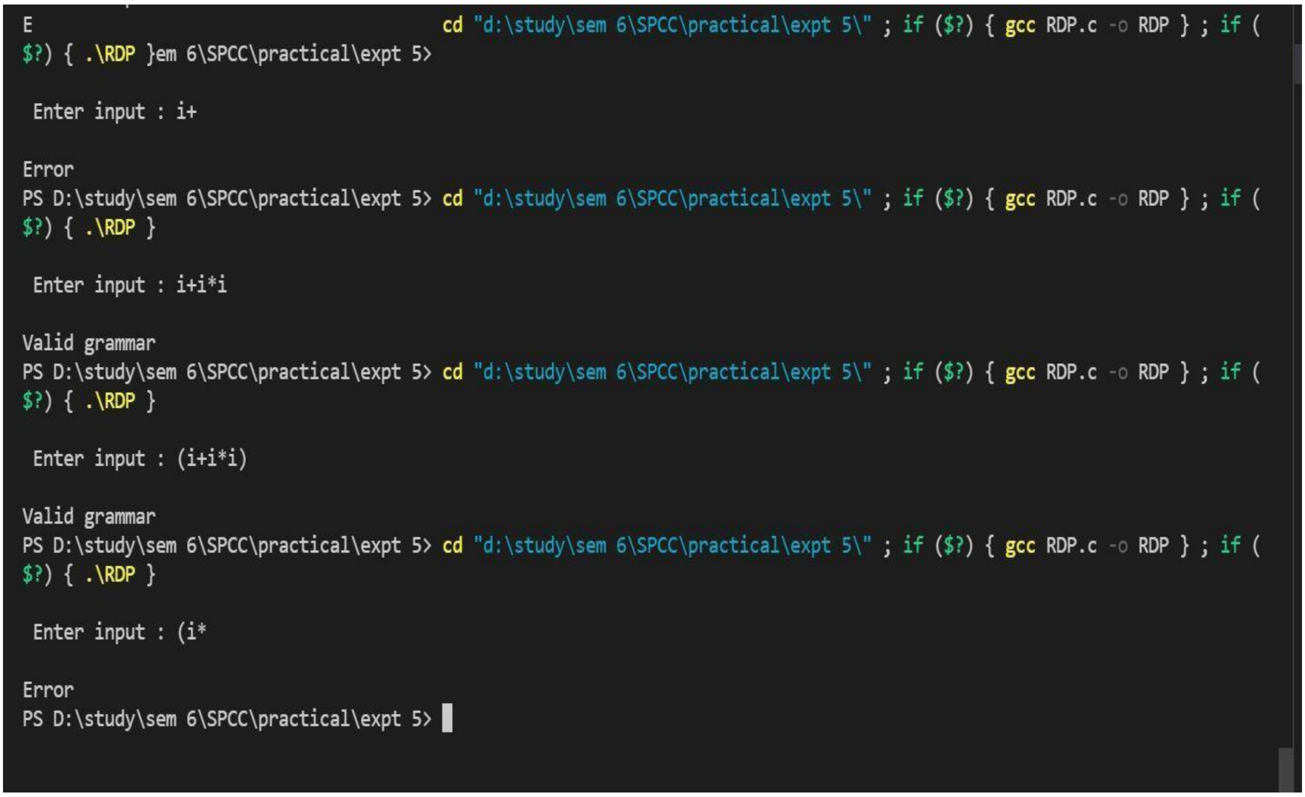
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>


int i = 0;

void E(char str[], int n);
void E_(char str[], int n);
void T(char str[], int n);
void T_(char str[], int n);
void F(char str[], int n);
```

```
int main()
    {
        char str[100];
        printf("\n Enter input : ");
        gets(str);
        int n = strlen(str);
        // puts(str);
        E(str, n);
        if (i == n)
        printf("\nValid grammar ");
        else
        printf("\nInValid grammar ");
        return 0;
    }
void E(char str[], int n)
    {
        T(str, n);
        E_(str, n);
        return;
    }
void E_(char str[], int n)
    {
        if (str[i] == '+')
            {
                i++;
            T(str, n);
            E_(str, n);
        }
      return;
    }
void T(char str[], int n)
    {
      F(str, n);
      T_(str, n);
      return;
    }
void T_(char str[], int n)
    {
      if (str[i] == '*')
        {
            i++;
            F(str, n);
            T_(str, n);
        }
      return;
    }
void F(char str[], int n)
    {
      if (str[i] == '(')
        {
            i++;
            E(str, n);
            if (str[i] == ')')
                {
                    i++;
                }
            else
                {
                    printf("\nError");
                    exit(0);
                }
        }
    else if (str[i] == 'i')
        {
            i++;
        }
    else
        {
            printf("\nError");
            exit(0);
        }
    return;
}
```

OUTPUT :



```
E                                            cd "d:\study\sem 6\SPCC\practical\expt 5\" ; if ($?) { gcc RDP.c -o RDP } ; if (
$?) { .\RDP }em 6\SPCC\practical\expt 5>

 Enter input : i+

Error
PS D:\study\sem 6\SPCC\practical\expt 5> cd "d:\study\sem 6\SPCC\practical\expt 5\" ; if ($?) { gcc RDP.c -o RDP } ; if (
$?) { .\RDP }

 Enter input : i+i*i

Valid grammar
PS D:\study\sem 6\SPCC\practical\expt 5> cd "d:\study\sem 6\SPCC\practical\expt 5\" ; if ($?) { gcc RDP.c -o RDP } ; if (
$?) { .\RDP }

 Enter input : (i+i*i)

Valid grammar
PS D:\study\sem 6\SPCC\practical\expt 5> cd "d:\study\sem 6\SPCC\practical\expt 5\" ; if ($?) { gcc RDP.c -o RDP } ; if (
$?) { .\RDP }

 Enter input : (i*

Error
PS D:\study\sem 6\SPCC\practical\expt 5>
```

PosHob (Exp5) SPCE

Q1 A grammar G(V, T, P, S) is left recursive if it has a Production in the form

$$A \rightarrow A\alpha \mid \beta$$

The above grammar is left recursive because the left of production is occurring at a first position on the right side of production. It can eliminate left ~~recursion~~ recursion by replacing a pair of production with

$$A \rightarrow \beta A'$$

$$A' \rightarrow \alpha A' \mid e$$

Elimination of Left Recursion.

Left Recursion can be eliminated by introducing new non-terminal A such that.

$$A \rightarrow A\alpha \mid \beta \xrightarrow{\text{Removal}} \begin{array}{l} A \rightarrow \beta A' \\ A' \rightarrow \alpha A' \mid \epsilon \end{array}$$

The general form for left recursion is (P.T.O) →

$$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \ldots\ldots \mid A\alpha_n \mid \beta_1 \mid \beta_2 \mid \ldots\ldots \beta_n$$

can be replaced by

$$A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \ldots\ldots \mid \ldots \mid \beta_n A'$$

$$A \rightarrow \alpha_1 A' \mid \alpha_2 A' \mid \ldots \mid \alpha_m A' \mid \epsilon$$

Q2 → Left Factoring is a process of factoring out the common prefixes which is transformed to make it useful for TOP down Parsers. In left factoring we make one production for each common prefixes.

The common prefix may be a terminal or a non-terminal or a combination of both. Rest of the derivation is added by new productions. The grammar obtained after the process of ~~the~~ left factoring is called as left factoring grammar.

Example :

$$A \rightarrow a\alpha_1 \mid a\alpha_2 \mid a\alpha_3$$

↓ left factoring

$$A \rightarrow \alpha A'$$

$$A' \rightarrow \alpha_1 \mid \alpha_2 \mid \alpha_3$$