Compiler Design Lab – UCS1602 Programming
Assignment-4
Implementation of Recursive Decent Parser

Harshini S
185001058

Program Code:

```
#include<stdio.h>
#include<stdlib.h>
void E();
void Eprime();
void T();
void Tprime();
void F();
char s[20];
int pos = 0;
int tabs = 0;
void printfOP(char t[]){
 printf("\n");
 for (int i = 0; i < tabs; ++i)
 printf(" ");
 printf("%s",t);
}
void advance(){
 printfOP("Adv()");
 if(s[pos]!='\0')
 pos++;
}
void E()
{
 printfOP("E()");
 tabs++;
 T();
 Eprime();
 tabs--;
}
```

```
void Eprime()
{
 printfOP("Eprime()");
 tabs++;
 if(s[pos] == '+') {
 printfOP("+ - successfully parsed");
 advance();
 T();
 Eprime();
 }
 tabs--;
}
void T()
{
 printfOP("T()");
 tabs++;
 F();
 Tprime();
 tabs--;
}
void Tprime()
{
 printfOP("Tprime()");
 tabs++;
 if(s[pos] == '*') {
 printfOP("* - successfully parsed");
 advance();
 F();
 Tprime();
 }
 tabs--;
}
void F()
{
 printfOP("F()");
 tabs++;
 if(s[pos] == '(') {
```

```
 printfOP("( - successfully parsed");
 advance();
 E();
 if(s[pos]==')'){
 printfOP(") - successfully parsed");
 advance();
 }
 else
 {
 printf("\nInvalid Input String..!!\n");
 exit(0);
 }
 }
 else if(s[pos] == 'i') {
 pos++;
 if(s[pos]=='d'){
 printfOP("id - successfully parsed");
 advance();
 }
 else{
 printf("\nInvalid Input String..!!\n");
 exit(0);
 }
 }
 else {
 printf("\nInvalid Input String..!!\n");
 exit(0);
 }
 tabs--;
}
int main()
{
 printf("\nEnter string to parse: ");
 scanf("%s",s);
 E();
 printf("\nInput String Accepted..!!\n");
 return 0;
```

```
}

/*Sample input/output
Enter string to parse: id+id*id

E()
 T()
  F()
   id - successfully parsed
   Adv()
  Tprime()
 Eprime()
  + - successfully parsed
  Adv()
  T()
   F()
    id - successfully parsed
    Adv()
   Tprime()
    * - successfully parsed
    Adv()
    F()
     id - successfully parsed
     Adv()
    Tprime()
  Eprime()
Input String Accepted..!!

Enter string to parse: (id+id*id

E()
 T()
  F()
   ( - successfully parsed
   Adv()
   E()
    T()
```

```
     F()
      id - successfully parsed
      Adv()
     Tprime()
    Eprime()
     + - successfully parsed
     Adv()
     T()
      F()
       id - successfully parsed
       Adv()
      Tprime()
       * - successfully parsed
       Adv()
       F()
        id - successfully parsed
        Adv()
       Tprime()
     Eprime()
Invalid Input String..!!

Enter string to parse: id-id

E()
 T()
  F()
   id - successfully parsed
   Adv()
  Tprime()
 Eprime()
Input String Accepted..!!

*/
```