Code:

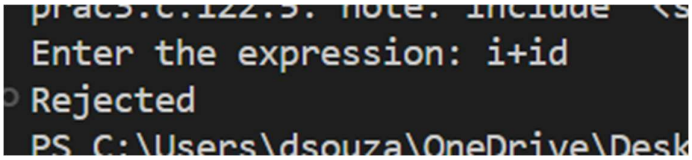
```
#include<stdio.h>
#include<string.h>
int i = 0;
char exp[50];
void advance(){
   i++;
}
void pE()
{
   pT();
   pEdash();
}
void pEdash()
{
   if(exp[i] =='+')
   {
      advance();
      pT();
      pEdash();
   }
}
void pT()
{
   pF();
```

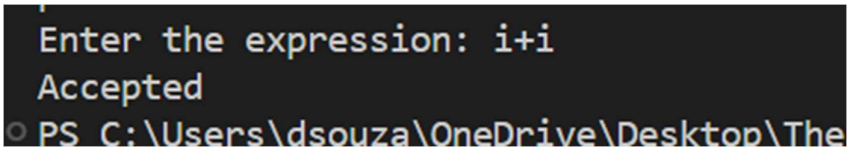
```
    pTdash();
}
void pTdash()
{
    if(exp[i] == '*')
    {
        advance();
        pF();
        pTdash();
    }
}
void pF()
{
    if(exp[i]=="id"){
        advance();
    }
    if(exp[i]=='(')
    {
        advance();
        pE();

        if(exp[i]==')')
            advance();
        else
            er();
    }
```

```
    else if(exp[i]=='i')
    {advance();}
    else
        er();
}
void er()
{ exit(0);}
int main()
{
    printf("Enter the expression: ");
    scanf("%s",exp);
    int n = strlen(exp);
    pE(exp);
    if(i == n)
        printf("Accepted");
    else
        printf("Rejected");
}
```

Output:





Conclusion: Thus we have designed a recursive descent parser is a kind of top-down parser built from a set of mutually-recursive procedures (or a non-recursive equivalent)