Week 6: Implementation of Recursive Descent Parser

Week 6 Programs

1. Implement Recursive Descent Parser for the Expression Grammar given below.

E → TE’
E’→ +TE’ | ϵ
T → FT’
T’→ *FT’ | ϵ
F → (E) | i

Program:

```
#include<stdio.h>
#include<string.h>
int E(),Edash(),T(),Tdash(),F();
char *ip;
char string[50];
int main()
{
 printf("Enter the string\n");
 scanf("%s",string);
 ip=string;
 printf("\n\nInput\tAction\n--------------------------------\n");
 if(E() && ip==’\0’){
 printf("\n--------------------------------\n");
printf("\n String is successfully parsed\n");  }
 else{
 printf("\n--------------------------------\n");
printf("Error in parsing String\n");  }
 }
int E()
{
 printf("%s\tE->TE' \n",ip);
 if(T())
 {
 if(Edash())
 {
 return 1;
 }
 else
 return 0;
 }
 else
 return 0;
 }
int Edash()
{
 if(*ip=='+')
 {
```

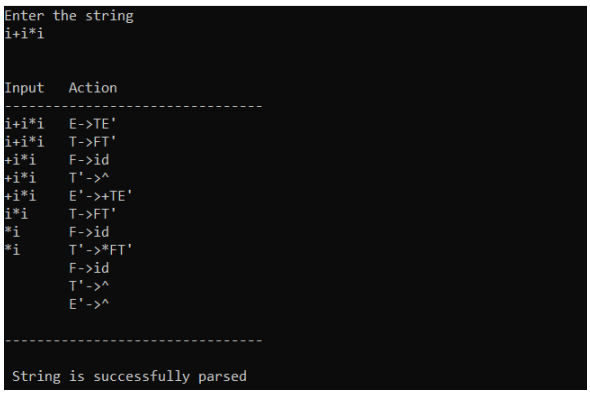
```
 printf("%s\tE'->+TE' \n",ip);
 ip++;
 if(T())
 {
 if(Edash())
 {
 return 1;
 }
 else
 return 0;
 }
 else
 return 0;
 }
 else
 {
 printf("%s\tE'->^ \n",ip);
 return 1;
 }
 }
int T()
 {
 printf("%s\tT->FT' \n",ip);
 if(F())
 {
 if(Tdash())
 {
 return 1;
 }
 else
 return 0;
 }
 else
 return 0;
 }
int Tdash()
 {
 if(*ip=='*')
 {
 printf("%s\tT'->*FT' \n",ip);
ip++;
 if(F())
 {
 if(Tdash())
 {
 return 1;
 }
 else
 return 0;
 }
 else
 return 0;
```

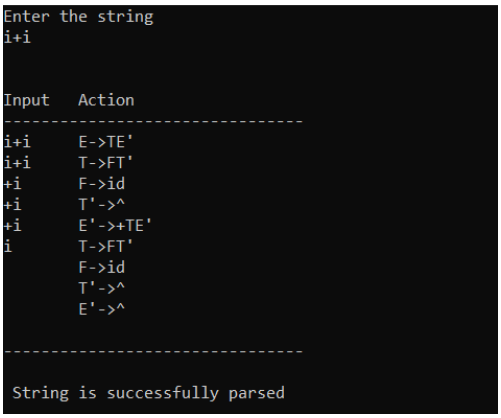
```
 }
 else
 {
 printf("%s\tT'->^ \n",ip);
return 1;
 }
 }
int F()
 {
 if(*ip=='(')
 {
 printf("%s\tF->(E) \n",ip);
ip++;
 if(E())
 {
 if(*ip==')')
 {
 ip++;
 return 0;
 }
 else
 return 0;
 }
 else
 return 0;
 }
 else if(*ip=='i')
 {
 ip++;
 printf("%s\tF->id \n",ip);
 return 1;
 }
 else
 return 0;
 }
```

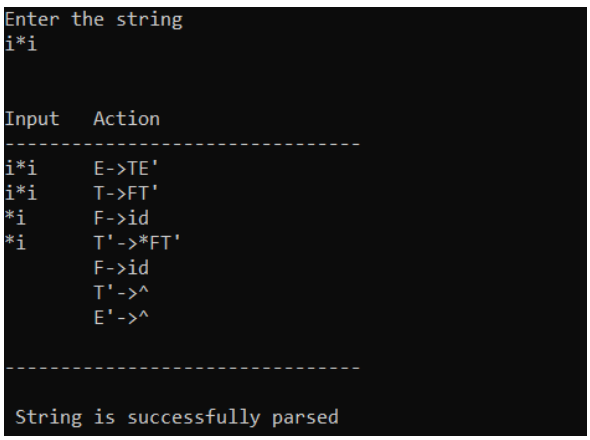
Test cases:

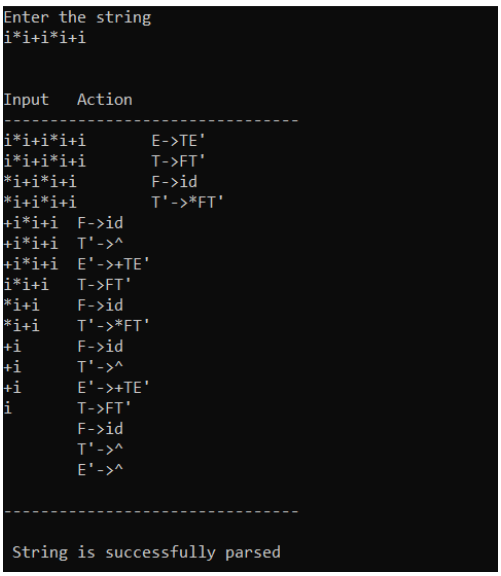
i+i*i	String is successfully parsed
i+i	String is successfully parsed
i*i	String is successfully parsed
i*i+i*i+i	String is successfully parsed
i+*+i	Error in parsing String
i+i*	Error in parsing String

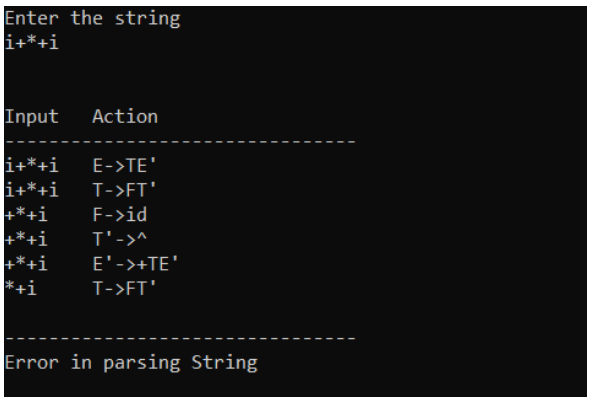
Output:

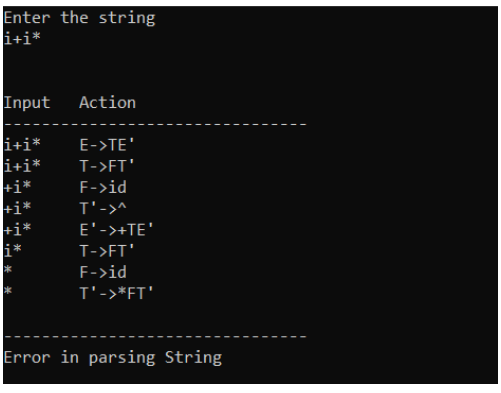

Enter the string
i+i*i
Input Action
i+i*i E->TE'
i+i*i T->FT'
+i*i F->id
+i*i T'->^
+i*i E'->+TE'
i*i T->FT'
*i F->id
*i T'->*FT'
F->id
T'->^
E'->^
String is successfully parsed


Enter the string
i+i
Input Action
i+i E->TE'
i+i T->FT'
+i F->id
+i T'->^
+i E'->+TE'
i T->FT'
F->id
T'->^
E'->^
String is successfully parsed


Enter the string
i*i
Input Action
i*i E->TE'
i*i T->FT'
*i F->id
*i T'->*FT'
F->id
T'->^
E'->^
String is successfully parsed


Enter the string
i*i+i*i+i
Input Action
i*i+i*i+i E->TE'
i*i+i*i+i T->FT'
*i+i*i+i F->id
*i+i*i+i T'->*FT'
+i*i+i F->id
+i*i+i T'->^
+i*i+i E'->+TE'
i*i+i T->FT'
*i+i F->id
*i+i T'->*FT'
+i F->id
+i T'->^
+i E'->+TE'
i T->FT'
F->id
T'->^
E'->^
String is successfully parsed


Enter the string
i+*+i
Input Action
i+*+i E->TE'
i+*+i T->FT'
+*+i F->id
+*+i T'->^
+*+i E'->+TE'
*+i T->FT'
Error in parsing String


Enter the string
i+i*
Input Action
i+i* E->TE'
i+i* T->FT'
+i* F->id
+i* T'->^
+i* E'->+TE'
i* T->FT'
* F->id
* T'->*FT'
Error in parsing String

2. Construct Recursive Descent Parser for the grammar
G = ({S, L}, {(,), a, ,}, {S → (L) | a ; L→ L, S | S}, S) and verify the acceptability of the following strings:
   i. (a,(a,a))
   ii. (a,((a,a),(a,a)))

Program:

```
#include<stdio.h>
#include<string.h>
int E(),L(),Ldash();
char *ip;
char string[50];
int main()
{
   printf("Enter the string:\n");
   scanf("%s",string);
   ip=string;
   printf("\n\nInput\tAction\n--------------------------------\n");
   if(E() && *ip=='\0'){
      printf("\n--------------------------------\n");
      printf("\n String is successfully parsed\n");
   }
   else{
      printf("\n--------------------------------\n");
      printf("Error in parsing String\n");
   }
}
int E()
{
   if(*ip=='(')
   {
      printf("%s\tS->(L) \n",ip);
      ip++;
      if(L())
      {
         if(*ip==')')
         {
            ip++;
            return 1;
         }
         else
            return 0;
      }
      else
         return 0;
```

```
    }
    else if(*ip=='a')
    {
        ip++;
        printf("%s\tS->a \n",ip);
        return 1;
    }
    else
        return 0;
}
int L()
{
    printf("%s\tL->SL' \n",ip);
    if(E())
    {
        if(Ldash())
        {
            return 1;
        }
        else
            return 0;
    }
    else
        return 0;
}
int Ldash()
{
    if(*ip==',')
    {
        printf("%s\tL'->,SL' \n",ip);
        ip++;
        if(E())
        {
            if(Ldash())
            {
                return 1;
            }
            else
                return 0;
        }
        else
            return 0;
    }
    else
    {
        printf("%s\tL'->^ \n",ip);
        return 1;
    }
}
```

Test cases:

(a,(a,a))	String is successfully parsed
(a,((a,a),(a,a)))	String is successfully parsed
(a,a	Error in parsing String
(a,(a,a)))	Error in parsing String

Output:

```
Enter the string:
(a,(a,a))


Input   Action
------------------------------
(a,(a,a))       S->(L)
a,(a,a))        L->SL'
,(a,a)) S->a
,(a,a)) L'->,SL'
(a,a))  S->(L)
a,a))   L->SL'
,a))    S->a
,a))    L'->,SL'
))      S->a
))      L'->^
)       L'->^

------------------------------

 String is successfully parsed
```

```
Enter the string:
(a,a


Input   Action
--------------------------------
(a,a    S->(L)
a,a     L->SL'
,a      S->a
,a      L'->,SL'
        S->a
        L'->^

--------------------------------
Error in parsing String
```

```
Enter the string:
(a,(a,a)))


Input   Action
--------------------------------
(a,(a,a)))      S->(L)
a,(a,a)))       L->SL'
,(a,a)))        S->a
,(a,a)))        L'->,SL'
(a,a))) S->(L)
a,a)))  L->SL'
,a)))   S->a
,a)))   L'->,SL'
)))     S->a
)))     L'->^
))      L'->^

--------------------------------
Error in parsing String
```

```
Enter the string:
(a,((a,a),(a,a)))


Input   Action
--------------------------------
(a,((a,a),(a,a)))       S->(L)
a,((a,a),(a,a)))        L->SL'
,((a,a),(a,a))) S->a
,((a,a),(a,a))) L'->,SL'
((a,a),(a,a)))  S->(L)
(a,a),(a,a)))   L->SL'
(a,a),(a,a)))   S->(L)
a,a),(a,a)))    L->SL'
,a),(a,a)))     S->a
,a),(a,a)))     L'->,SL'
),(a,a)))       S->a
),(a,a)))       L'->^
,(a,a)))        L'->,SL'
(a,a))) S->(L)
a,a)))  L->SL'
,a)))   S->a
,a)))   L'->,SL'
)))     S->a
)))     L'->^
))      L'->^
)       L'->^

--------------------------------

 String is successfully parsed
```