Compiler Design Lab
Week 7 : implementation of LL(1) parser

Lab Assignment: Implement Predictive Parser using C for the Expression Grammar

$E \rightarrow TE'$
$E' \rightarrow +TE' \mid \varepsilon$
$T \rightarrow FT'$
$T' \rightarrow *FT' \mid \varepsilon$
$F \rightarrow (E) \mid d$

The grammar is renamed for convenience the renamed grammar is as follows:

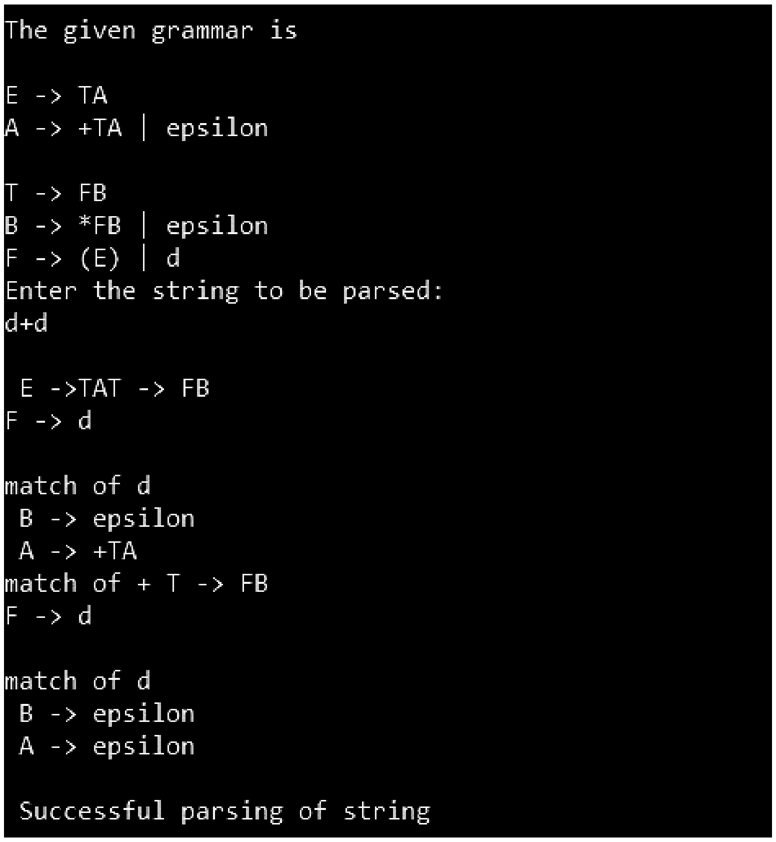
$E \rightarrow TA$
$A \rightarrow +TA \mid \varepsilon$
$T \rightarrow FB$
$B \rightarrow *FB \mid \varepsilon$
$F \rightarrow (E) \mid d$

Program:

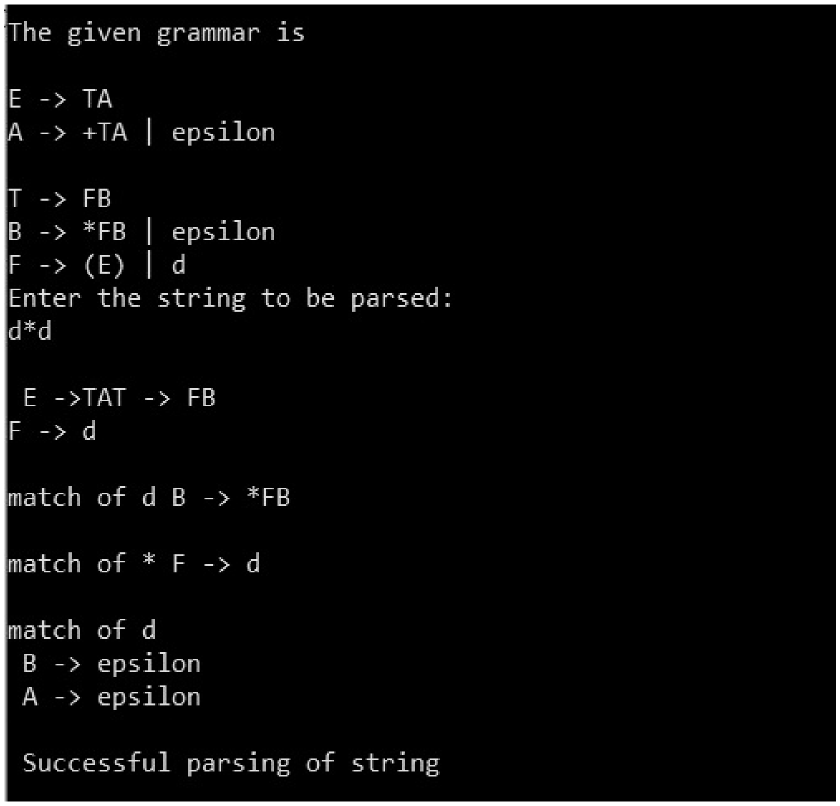
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
int i=0,top=0;
char stack[20],ip[20];

void push(char c)
{
        if (top>=20)
                printf("Stack Overflow");
        else
                stack[top++]=c;
}

void pop(void)
{
        if(top<0)
                printf("Stack underflow");
        else
                top--;
}

void error(void)
{
   printf("\n\nSyntax Error, String is invalid\n");
   getch();
   exit(0);
}

int main()
{
   int n;

   printf("The given grammar is\n\n");
   printf("S -> aBa\n");
   printf("B -> bB | epsilon \n\n");
   printf("Enter the string to be parsed:\n");
   scanf("%s",ip);
   n=strlen(ip);
   ip[n]='$';
   ip[n+1]='\0';
   push('$');
   push('S');
   printf("\nproductions\t\t\tcharacter match\n");
   while(ip[i]!='\0')
   {
      if(ip[i]=='$' && stack[top-1]=='$')
      {
         printf("\n\n Successful parsing of string \n");
         return(1);

      }
      else if(ip[i]==stack[top-1])
```

```
          {
             printf("\t\tmatch of %c occured ",ip[i]);
             i++;
             pop();
          }
          else
          {
                  if(stack[top-1]=='S' && ip[i]=='a')
                  {
                     printf("\nS ->aBa\t\t");
                     pop();
                     push('a');
                     push('B');
                     push('a');
                  }
                  else if(stack[top-1]=='B' && ip[i]=='b')
                  {
                     printf("\nB ->bB\t\t");
                          pop();
                          push('B');
                          push('b');
                  }
                  else if(stack[top-1]=='B' && ip[i]=='a')
                  {
                     printf("\nB -> epsilon\t");
                          pop();
                  }
                  else
                  {
                     error();
                  }
          }
      }
}
```

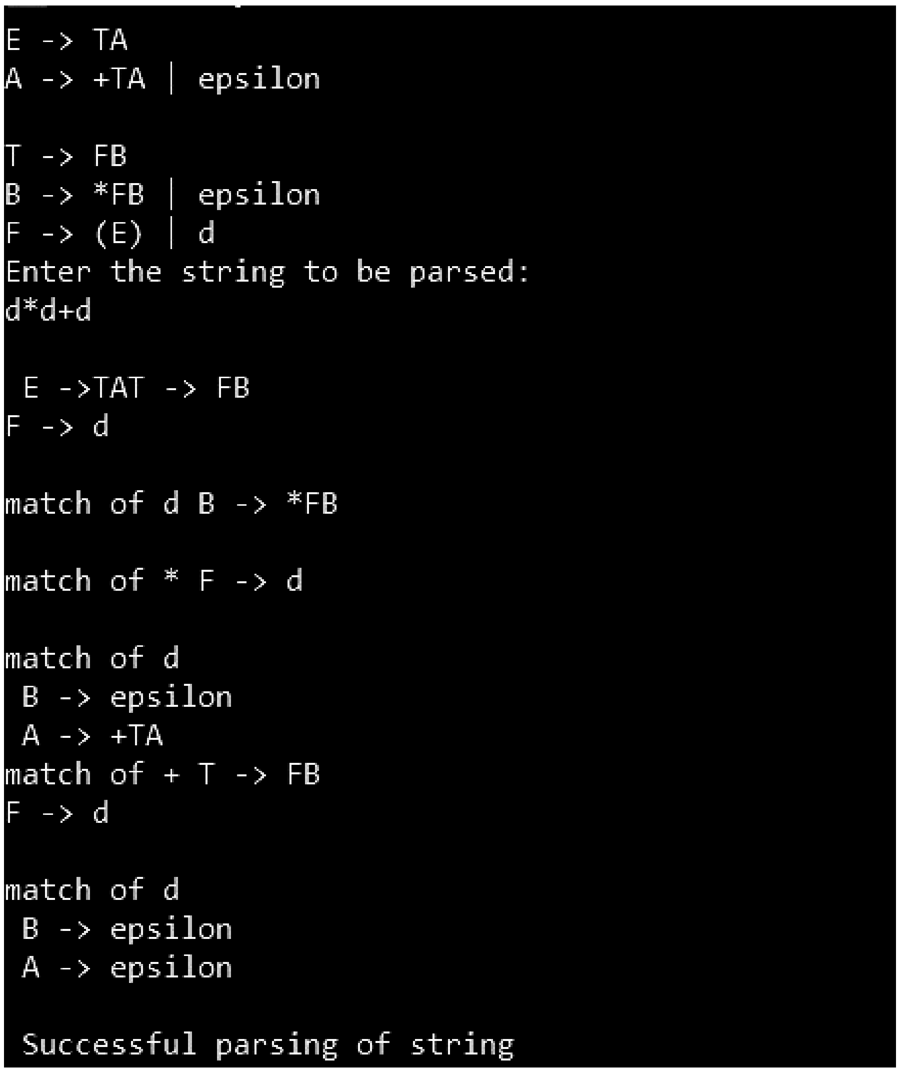
Test Cases:



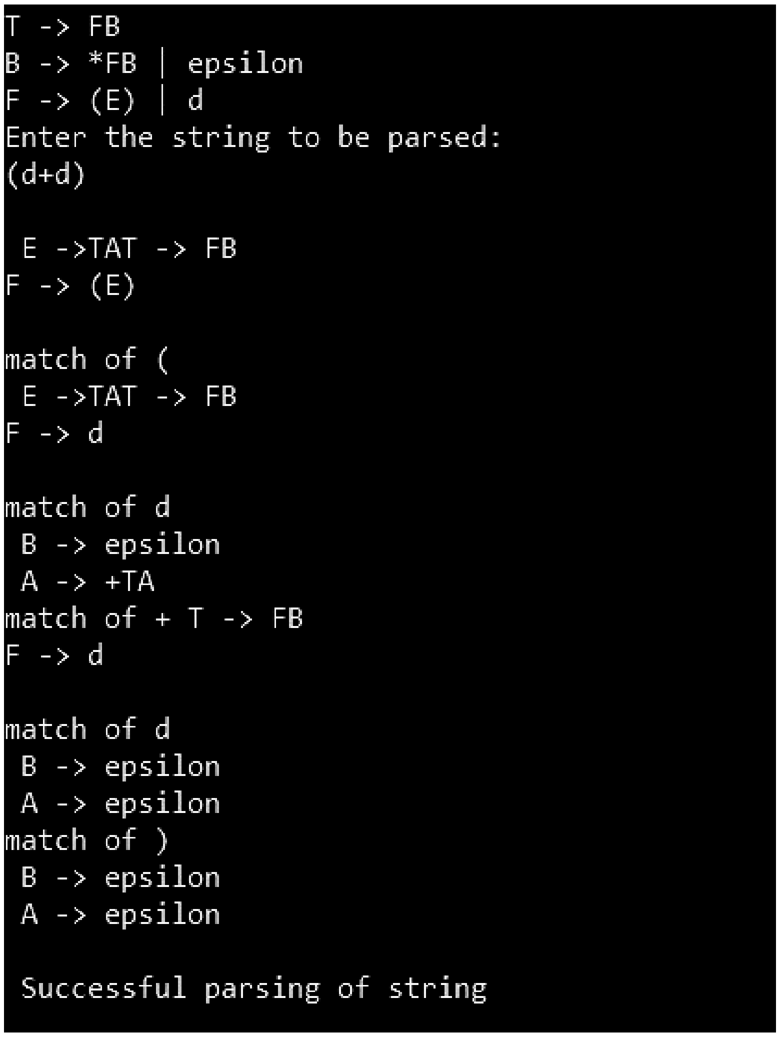
```
The given grammar is

E -> TA
A -> +TA | epsilon

T -> FB
B -> *FB | epsilon
F -> (E) | d
Enter the string to be parsed:
d+d

 E ->TAT -> FB
F -> d

match of d
 B -> epsilon
 A -> +TA
match of + T -> FB
F -> d

match of d
 B -> epsilon
 A -> epsilon

 Successful parsing of string
```



```
The given grammar is

E -> TA
A -> +TA | epsilon

T -> FB
B -> *FB | epsilon
F -> (E) | d
Enter the string to be parsed:
d*d

 E ->TAT -> FB
F -> d

match of d B -> *FB

match of * F -> d

match of d
 B -> epsilon
 A -> epsilon

 Successful parsing of string
```



```
E -> TA
A -> +TA | epsilon

T -> FB
B -> *FB | epsilon
F -> (E) | d
Enter the string to be parsed:
d*d+d

 E ->TAT -> FB
F -> d

match of d B -> *FB

match of * F -> d

match of d
 B -> epsilon
 A -> +TA
match of + T -> FB
F -> d

match of d
 B -> epsilon
 A -> epsilon

 Successful parsing of string
```



```
T -> FB
B -> *FB | epsilon
F -> (E) | d
Enter the string to be parsed:
(d+d)

 E ->TAT -> FB
F -> (E)

match of (
 E ->TAT -> FB
F -> d

match of d
 B -> epsilon
 A -> +TA
match of + T -> FB
F -> d

match of d
 B -> epsilon
 A -> epsilon
match of )
 B -> epsilon
 A -> epsilon

 Successful parsing of string
```