COMPILER DESIGN WEEK 7 LAB EXERCISE AP19110010369
CHANDANA-CSE

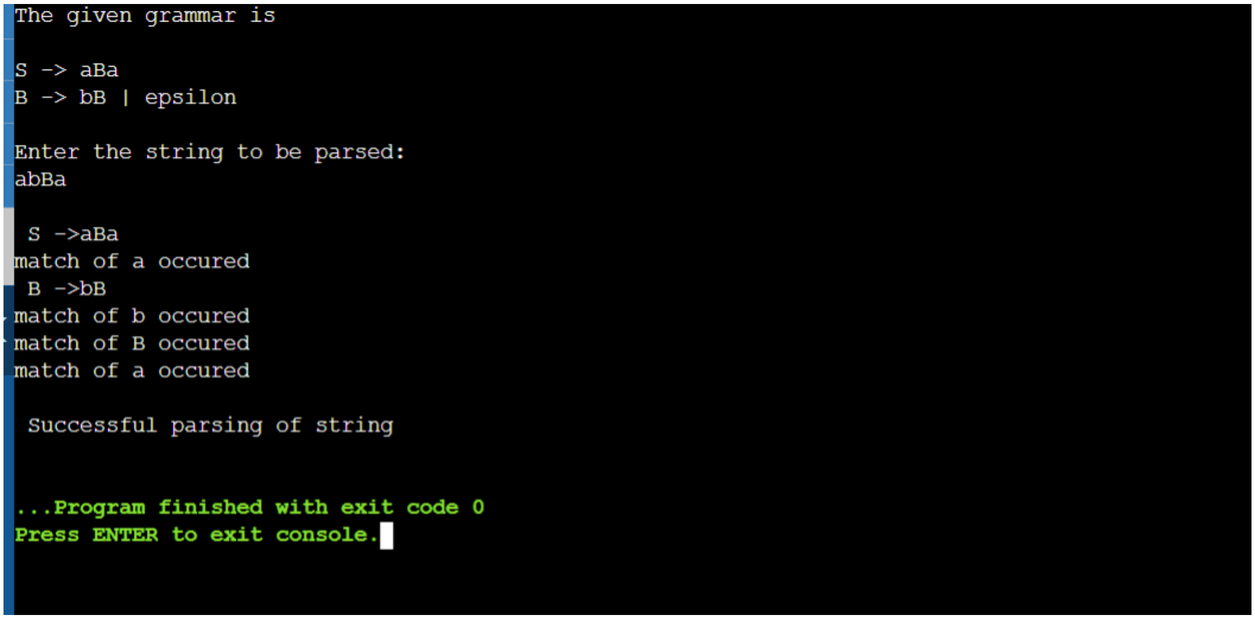
A)

CODE:

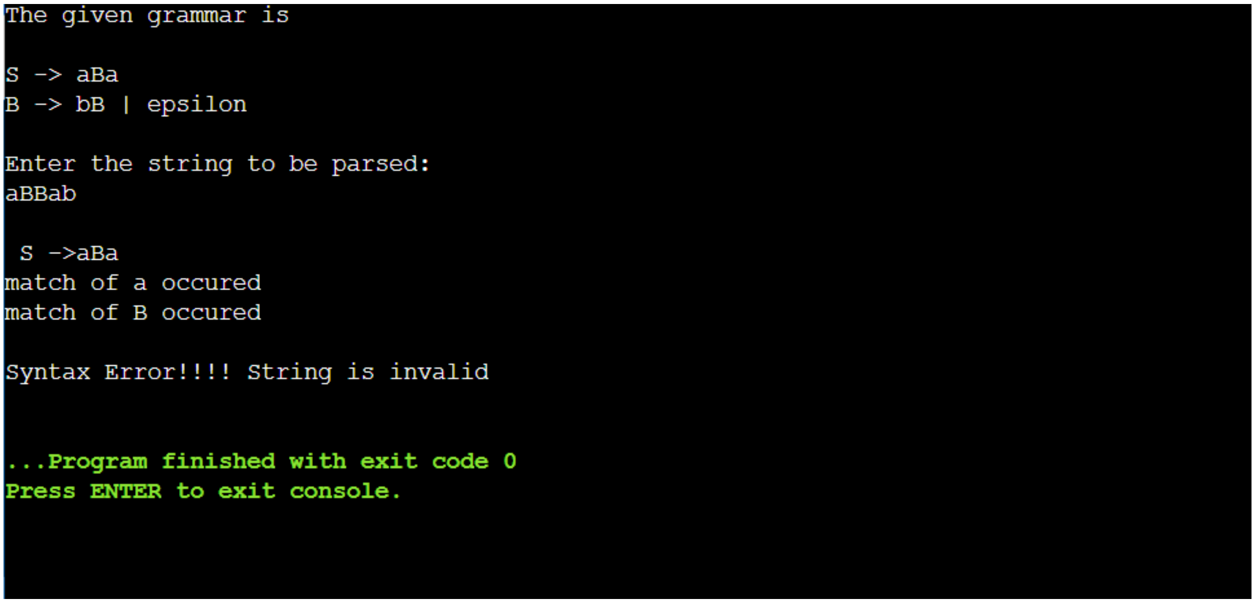
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
int i=0,top=0;
char stack[20],ip[20];

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');
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("\nmatch of %c occured ",ip[i]);
            i++;pop();
          }
          else
          {
                    if(stack[top-1]=='S' && ip[i]=='a')
                    {
                          printf(" \n S ->aBa");
                          pop();
                          push('a');
                          push('B');
                   push('a');
                    }
                    else
                    if(stack[top-1]=='B' && ip[i]=='b')
                    {
                              printf("\n B ->bB");
                              pop();push('B');push('b');
                    }
                        else
                          if(stack[top-1]=='B' && ip[i]=='a')
                          {
                              printf("\n B -> epsilon");
                              pop();
                          }
                          else
                              error();
          }
   }
}//end of main
```

OUTPUT:



```
The given grammar is

S -> aBa
B -> bB | epsilon

Enter the string to be parsed:
abBa

 S ->aBa
match of a occured
 B ->bB
match of b occured
match of B occured
match of a occured

 Successful parsing of string


...Program finished with exit code 0
Press ENTER to exit console.
```



```
The given grammar is

S -> aBa
B -> bB | epsilon

Enter the string to be parsed:
aBBab

 S ->aBa
match of a occured
match of B occured

Syntax Error!!!! String is invalid


...Program finished with exit code 0
Press ENTER to exit console.
```

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

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

CODE:

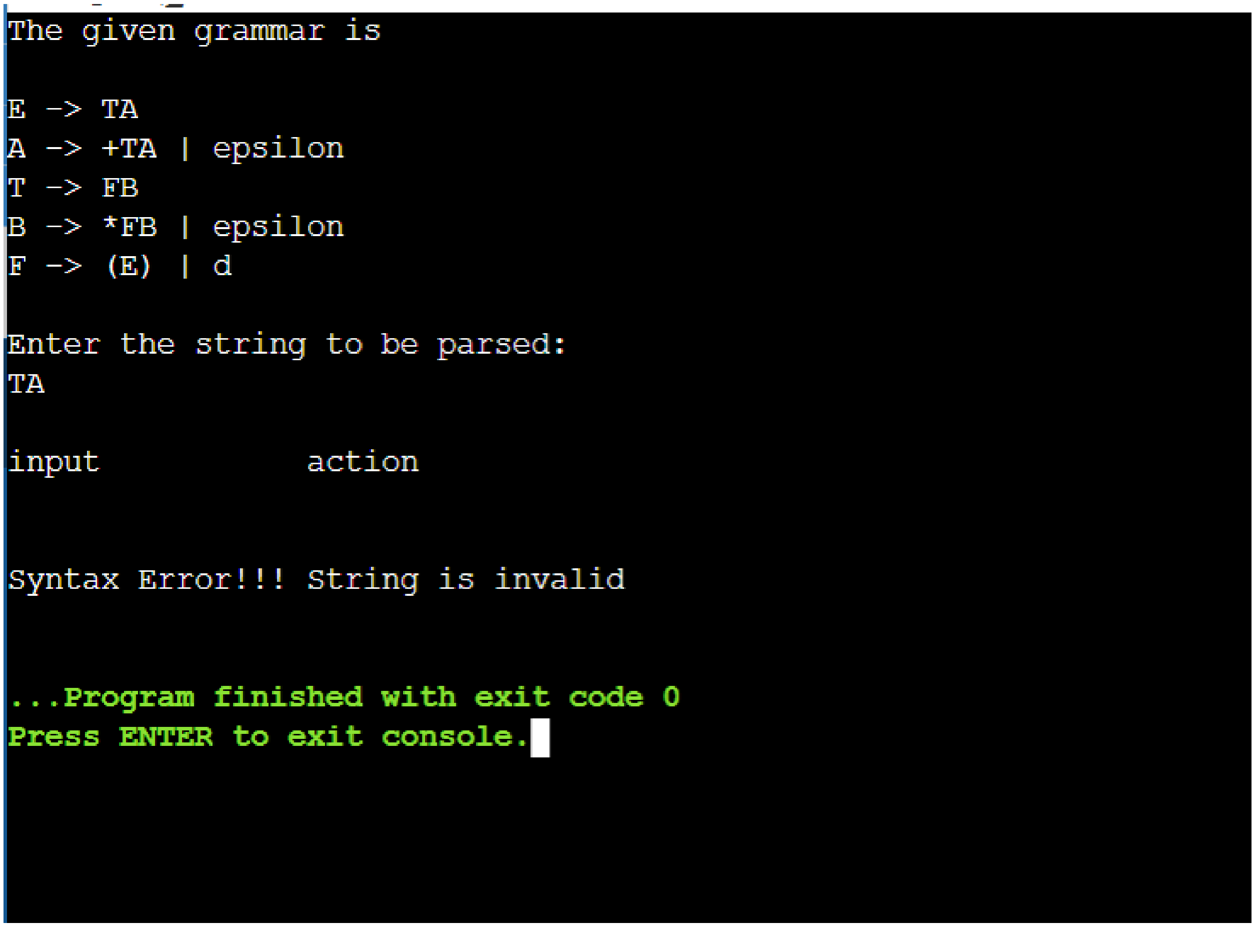
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
int i=0,top=0;
char stack[20],ip[20];

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("E -> TA\n");
    printf("A -> +TA | epsilon\n");
    printf("T -> FB\n");
    printf("B -> *FB | epsilon\n");
    printf("F -> (E) | d\n\n");
    printf("Enter the string to be parsed:\n");
    scanf("%s",ip);
    n=strlen(ip);
    ip[n]='$';
    ip[n+1]='\0';
    push('$');
    push('E');
    printf("\ninput\t\taction\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("match of %c occured ",ip[i]);
               i++;
               pop();
           }
           else
           {
                      if(stack[top-1]=='E' && ip[i]=='d')
                      {
                          printf("\nE ->TA\t\t");
                          pop();
                          push('A');
                          push('T');
                      }
                      else if(stack[top-1]=='E' && ip[i]=='(')
                      {
                          printf("\nE ->TA\t\t");
                              pop();
                              push('A');
                              push('T');
                      }
                      else if(stack[top-1]=='A' && ip[i]=='+')
                      {
                          printf("\nA -> +TA\t");
                          pop();
                          push('A');
                          push('T');
                          push('+');
                      }
                      else if(stack[top-1]=='A' && ip[i]==')')
                      {
                          printf("\nA -> epsilon\t");
                          pop();
                      }
                      else if(stack[top-1]=='A' && ip[i]=='$')
                      {
                          printf("\nA -> epsilon\t");
                          pop();
                      }
                      else if(stack[top-1]=='T' && ip[i]=='d')
                      {
                          printf("\nT ->FB\t\t");
                          pop();
                          push('B');
                          push('F');
                      }
                      else if(stack[top-1]=='T' && ip[i]=='(')
                      {
                          printf("\nT ->FB\t\t");
                              pop();
                              push('B');
                              push('F');
```

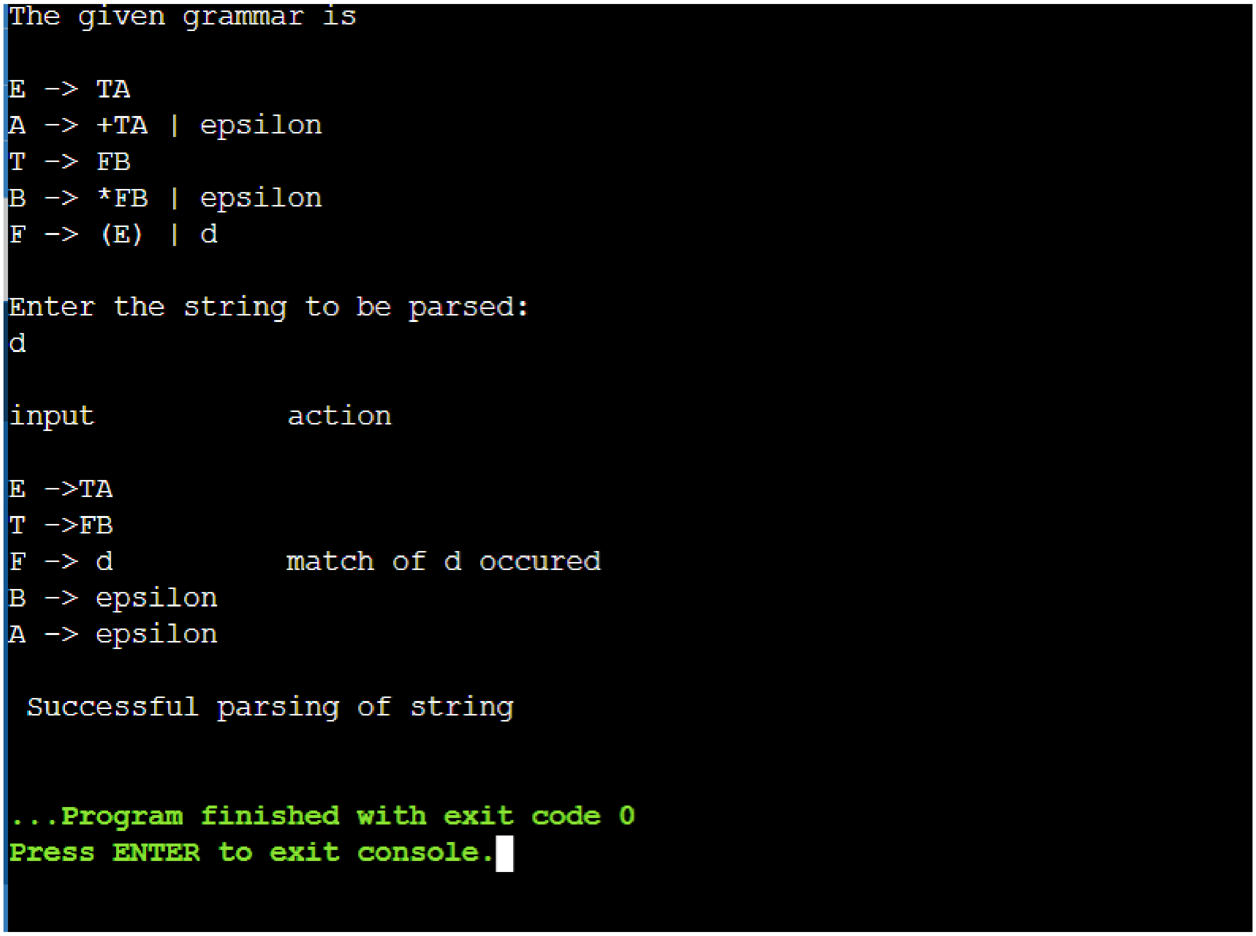
```
}
else if(stack[top-1]=='B' && ip[i]=='+')
{
    printf("\nB -> epsilon\t");
    pop();
}
else if(stack[top-1]=='B' && ip[i]=='*')
{
    printf("\nB -> *FB\t");
    pop();
    push('B');
    push('F');
    push('*');
}
else if(stack[top-1]=='B' && ip[i]==')')
{
    printf("\nB -> epsilon\t");
    pop();
}
else if(stack[top-1]=='B' && ip[i]=='$')
{
    printf("\nB -> epsilon\t");
    pop();
}
else if(stack[top-1]=='F' && ip[i]=='d')
{
    printf("\nF -> d\t\t");
    pop();
    push('d');
}
else if(stack[top-1]=='F' && ip[i]=='(')
{
    printf("\nF -> (E)\t");
    pop();
    push(')');
    push('E');
    push('(');
}
else
{
```

```
                    error();
                }
            }
        }
    }
```

OUTPUT:



```
The given grammar is

E -> TA
A -> +TA | epsilon
T -> FB
B -> *FB | epsilon
F -> (E) | d

Enter the string to be parsed:
TA

input           action


Syntax Error!!! String is invalid


...Program finished with exit code 0
Press ENTER to exit console.
```



```
The given grammar is

E -> TA
A -> +TA | epsilon
T -> FB
B -> *FB | epsilon
F -> (E) | d

Enter the string to be parsed:
d

input           action

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

 Successful parsing of string


...Program finished with exit code 0
Press ENTER to exit console.
```