Compiler Design Lab (CS 306)

Week 7: Implementation of LL(1) parser using C

KREETHI MISHRA
AP19110010424
CSE-C

Week 7 Program

1. Implement non-recursive Predictive Parser for the grammar

S -> aBa
B -> bB | ε

	a	b	$
S	S → aBa		
B	B → ε	B → b B	

Programs:

Code of first 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");
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 ",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

        }
```

Testcases: Test your program with test cases covering all requirements.

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

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

PROGRAM:

Code of second 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");
exit(0);
}
int main()
{
int n;
printf("The given grammar is\n\n");
printf("E → TE'\n");
printf("E'→ +TE'| epsilon \n\n");
printf("T → FT'\n");
printf("T'→ *FT'| epsilon \n\n");
printf("F → (E) | d \n");
printf("Enter the string to be parsed:\n");
scanf("%s",ip);
n=strlen(ip);
ip[n]='$';
ip[n+1]='\0';
push('$');
push('E');
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 ",ip[i]);
i++;pop();
}
else
{
if(stack[top-1]=='E' && (ip[i]=='d' || ip[i]='(' ))
{
printf(" \n E → TE' ");
pop();
push('A');
push('T');

}
else
if(stack[top-1]=='A' && ip[i]=='+')

{
printf("\n E'→ +TE' ");
pop();push('A');push('T');push('+');
}
else
if(stack[top-1]=='A' && (ip[i]==')' || ip[i]=='$'))
```

```
{
printf("\n  E'-> epsilon");
pop();
}
else
if(stack[top-1]=='T' && (ip[i]=='d' || ip[i]=='('))
{
printf("\n  T → FT'");
pop();push('B');push('F');
}
else
if(stack[top-1]=='B' && ip[i]=='*')

{
printf("\n T'→ *FT' ");
pop();push('B');push('F');push('*');
}
else
if(stack[top-1]=='B' && (ip[i]=='+'|| ip[i]==')' ||ip[i]=='$'))
{
printf("\n  T'-> epsilon");
pop();
}
else
if(stack[top-1]=='F' && ip[i]=='d')

{
printf("\n F → d ");
pop();push('d');
}
else
if(stack[top-1]=='F' && ip[i]=='(')

{
printf("\n F → (E)  ");
pop();push(')');push('E');push('(');
}
else
error();
}
}
}
```

Output:

```
The given grammar is

E → TE'
E'→ +TE'| epsilon

T → FT'
T'→ *FT'| epsilon

F → (E) | d
Enter the string to be parsed:
d*d+d

 E → TE'
   T → FT'
 F → d
match of d
 T'→ *FT'
match of *
 F → d
match of d
   T'-> epsilon
 E'→ +TE'
match of +
   T → FT'
 F → d
match of d
   T'-> epsilon
   E'-> epsilon

 Successful parsing of string


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