ASSIGNMENT 5: OPERATOR PRECEDENCE PARSER

NAME : BIJAY KHATRI , DEEPAK VERMA

ROLL No. : 12/CS/45 , 12/CS/46

```
#include <stdio.h>
#include <string.h>
int top=-1;
char stack[60];
char grammar[10][10];
char input[10];
int precedence[256][256];
int i=0;
int lim;

char pop()
{
	char val=stack[top];
	top--;
	return val;
}
void push(char symbol)
{
	stack[++top]=symbol;
}

void display()
{
	int jm;
	char stk_str[20];
	for(jm=0;jm<=top;jm++)
		printf("%c",stack[jm]);
}

void getLine(FILE *fp)
{
	int j=0;
	char ch;
	scanf("%d",&lim);
	while(i<lim)
	{
		ch=getchar();
		if(ch==EOF)
			break;

		if(ch=='\n')
		{
			i++,j=0;
		}
		else
		{
			grammar[i][j]=ch;
			j++;
		}
```

```
	}
	scanf("%s",input);
}

int main(void)
{
	precedence['+']['+']=1;
	precedence['+']['-']=1;
	precedence['+']['*']=-1;
	precedence['+']['/']=-1;
	precedence['+']['$']=1;

	precedence['-']['+']=1;
	precedence['-']['-']=1;
	precedence['-']['*']=-1;
	precedence['-']['/']=-1;
	precedence['-']['$']=1;

	precedence['*']['+']=1;
	precedence['*']['-']=1;
	precedence['*']['*']=1;
	precedence['*']['/']=1;
	precedence['*']['$']=1;

	precedence['/']['+']=1;
	precedence['/']['-']=1;
	precedence['/']['*']=1;
	precedence['/']['/']=1;
	precedence['/']['$']=1;
	getLine(stdin);
	int m,n;
	strcat(input,"$");
	int index=0;
	int len= strlen(input);
	push('$');
	int flag=1;
	printf("STACK\t  INPUT BUFFER  \tACTION\n");
		while(index<len && flag)
	{

		char current= input[index];

		//printf("Current %c\n", current);

		if(stack[top]=='$' && current!='$')
		{
			printf("\n");
			display();
			printf("\t\t%s\t\tSHIFT %c\n",input+index,current);
			push(current);
			index++;
		}

		else if(top>0)
		{
```

```
                        if(top==1 && stack[top]==grammar[1][0] && current=='$')
                        {
                                //accept
                                display();
                                printf("\t\t%s\t\tACCEPT\n",input+index);
                                break;
                        }

                        else if((top%2==1) && stack[top]>='a' && stack[top]<='z')
                                {
                                        display();
                                        char val_on_top = pop();
                                        //find left-hand side of the production
                                        int temp;
                                        int f=0;
                                        for(temp=0; temp<lim;temp++)
                                        {
                                                if(strlen(grammar[temp])==3 && grammar[temp]
[2]==val_on_top)
                                                        {
                                                                push(grammar[temp][0]);
                                                                printf("\t\t%s\t\tREDUCE BY
%s\n",input+index, grammar[temp]);
                                                                f=1;
                                                                break;
                                                        }
                                        }
                                        if(f==0)
                                        {
                                                flag=0;
                                                printf("\t\t%s\t\t",input+index);
                                                printf("REJECTED\n");
                                                break;
                                        }

                                }

                        else if(top%2==0 && current!='$')
                        {
                                display();
                                printf("\t\t%s\t\tSHIFT %c\n",input+index, current);
                                push(current);
                                index++;
                        }

                        else if((top>=3&& top%2==1) && stack[top]>='A' && stack[top]<='Z' &&
precedence[stack[top-1]][current]==1)
                        {
                                //printf("PRECEDENCE of %c and %c : %d \n",stack[top-1], current,
precedence[stack[top-1]][current] );
                                display();
                                char val1= pop();
                                char val2= pop();
                                char val3= pop();
                                        //find left-hand side of the production
```

```
                                        int temp;
                                        int f=0;
                                        for(temp=0; temp<lim;temp++)
                                        {
                                                if(strlen(grammar[temp])==5 && grammar[temp]
[2]==val1 && grammar[temp][3]==val2 && grammar[temp][4]==val3)
                                                        {
                                                                push(grammar[temp][0]);
                                                                printf("\t\t%s\t\tREDUCE BY
%s\n",input+index,grammar[temp]);
                                                                f=1;
                                                                break;
                                                        }
                                        }
                                        if(f==0)
                                        {
                                                flag=0;
                                                printf("\t\t%s\t\t",input+index);
                                                printf("REJECTED\n");
                                                break;
                                        }

                        }
                        else if(top==1 && stack[top]>='A' && stack[top]<='Z' || (top==3 &&
precedence[stack[top-1]][current]==-1))
                        {
                                //printf("Here: \n");
                                display();
                                printf("\t\t%s\t\tSHIFT %c\n",input+index,current);
                                push(current);
                                index++;
                        }

                        else if(current=='$' && stack[top]!=grammar[1][0])
                        {
                                display();
                                printf("\t\t%s\t\tREJECT\n",input+index);
                                break;
                        }
                }
        }

}
```

INPUT:

```
6
E=E+E
E=E-E
E=E*E
E=E/E
E=a
a+a-a+a*a/a-a+a/a
```

OUTPUT:

STACK	INPUT BUFFER	ACTION
$	a+a-a+a*a/a-a+a/a$	SHIFT a
$a	+a-a+a*a/a-a+a/a$	REDUCE BY E=a
$E	+a-a+a*a/a-a+a/a$	SHIFT +
$E+	a-a+a*a/a-a+a/a$	SHIFT a
$E+a	-a+a*a/a-a+a/a$	REDUCE BY E=a
$E+E	-a+a*a/a-a+a/a$	REDUCE BY E=E+E
$E	-a+a*a/a-a+a/a$	SHIFT -
$E-	a+a*a/a-a+a/a$	SHIFT a
$E-a	+a*a/a-a+a/a$	REDUCE BY E=a
$E-E	+a*a/a-a+a/a$	REDUCE BY E=E-E
$E	+a*a/a-a+a/a$	SHIFT +
$E+	a*a/a-a+a/a$	SHIFT a
$E+a	*a/a-a+a/a$	REDUCE BY E=a
$E+E	*a/a-a+a/a$	SHIFT *
$E+E*	a/a-a+a/a$	SHIFT a
$E+E*a	/a-a+a/a$	REDUCE BY E=a
$E+E*E	/a-a+a/a$	REDUCE BY E=E*E
$E+E	/a-a+a/a$	SHIFT /
$E+E/	a-a+a/a$	SHIFT a
$E+E/a	-a+a/a$	REDUCE BY E=a
$E+E/E	-a+a/a$	REDUCE BY E=E/E
$E+E	-a+a/a$	REDUCE BY E=E+E
$E	-a+a/a$	SHIFT -
$E-	a+a/a$	SHIFT a
$E-a	+a/a$	REDUCE BY E=a
$E-E	+a/a$	REDUCE BY E=E-E
$E	+a/a$	SHIFT +
$E+	a/a$	SHIFT a
$E+a	/a$	REDUCE BY E=a
$E+E	/a$	SHIFT /
$E+E/	a$	SHIFT a
$E+E/a	$	REDUCE BY E=a
$E+E/E	$	REDUCE BY E=E/E
$E+E	$	REDUCE BY E=E+E
$E	$	ACCEPT