Postfix Eval

Write a C code to evaluate a postfix expression using stacks. You are expected to fill in the code for the following functions.

'pop' for popping the elements from the stack.

'push' for pushing elements into the stack.

'eval' for evaluating the given postfix expression.

'disp' for printing the result after evaluating the postfix expression.

Input Format

Postfix expression as a string.

Constraints

Each character in the string should either be numbers from 0-9 or '+' or '-', '*', '/'.

0<= strlen(postfix_expression)<=30.

sizeof(stack) = 30.

It is assumed that the size of the expression is always less than the size of the stack.

In case you are using exit function, use exit(0) and not exit(1) for the submission to get accepted.

Output Format

Resultant value after evaluating the postfix expression.

Print "Stack is empty" when you try to pop from an empty stack.

Print "Stack is full" when you try to push into a stack which is full.

Sample Input 0

```
231*+9-
```

Sample Output 0

```
-4
```

Explanation 0

3*1 is evaluated and gives 3.

2+3 is evaluated and gives 5.

5-9 is evaluated and gives -4.

Submissions: 20
Max Score: 10
Difficulty: Easy

Rate This Challenge:

More

C

```c
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#define MAX 30

void push(int s, int* top, int stack[MAX]);
int pop(int stack[MAX], int* top);
void eval(int stack[MAX], int* top, char postfix[MAX]);
void disp(int stack[MAX],int* top);

int main()
{
    int stack[MAX];
    int top = -1;
    char postfix[30];
    scanf("%s", postfix);
    eval(stack, &top, postfix);
    disp(stack,&top);
    return 0;
}

void push(int s, int* top, int stack[MAX])
{
    if(*top>=MAX){
        printf("Stack is full");
        exit (0);
    }
    else{
        (*top)++;
        stack[*top]=s;
    }
}

int pop(int stack[MAX], int* top)
{
    if(*top==-1){
        printf("Stack is empty");
        exit(0);
    }
    else{
        int res = stack[*top];
        (*top)--;
        return res;
    }
}

void eval(int stack[MAX], int* top, char postfix[MAX])
{
    int op1, op2, res;
    for(int i=0; postfix[i]!='\0'; i++){
        if(isdigit(postfix[i])){
            push(postfix[i]-'0', top, stack);
        }
        else{
            op2=pop(stack, top);
            op1=pop(stack, top);

        switch (postfix[i]){
            case '+' : res=op1+op2;
                break;
            case '-' : res=op1-op2;
```

```
                break;
            case '*' : res=op1*op2;
                break;
            case '/' : res=op1/op2;
                break;
            default:printf("Invalid operation\n");
                break;
            }
        push(res, top, stack);
        }
    }
}

void disp(int stack[MAX],int* top)
{
    if(*top==-1){
        printf("Stack is empty");
        return ;
    }
    else if(*top>=MAX){
        printf("Stack is full");
        return;
    }
    else{
        printf("%d",pop(stack, top));
    }
}
```

Line: 88 Col: 2

Upload Code as File Test against custom input Run Code Submit Code