Data Structure
Homework #2
Due 24 Oct (Sun), 11:59, 2021

1. **[Check Parentheses Balancing: 40 Pt]** 주어진 식이 다음과 같은 형태로 이루어져 있다고 가정한다: 숫자: 0~9, 알파벳 대문자: A-Z, 4가지 종류의 연산자 (operators): /, * , +, -, 그리고 시작과 닫힘을 표시하는 세가지 종류의 괄호 문자: (,),{,},[,]. 괄호의 시작과 담힘의 밸런싱 (balancing)과 열림-담힘 쌍이 맞는지 체크하는 프로그램을 **배열 기반 스택**을 이용하여 작성하시오.

Input

괄호를 포함하는 중위 표현 수식

Output

입력 문자열 (string)의 괄호가 밸런싱이 맞았을 경우 (열림-닫힘 쌍 검사 포함), 'Yes' 라고 출력하고, 그렇지 않을 경우 'No' 라고 출력한다.

Sample Input

( )
) (
{ 3+(2*5) }
{ }
(3*4+ (6)
({2+5) }

Sample Output

Yes

No

Yes

Yes

No

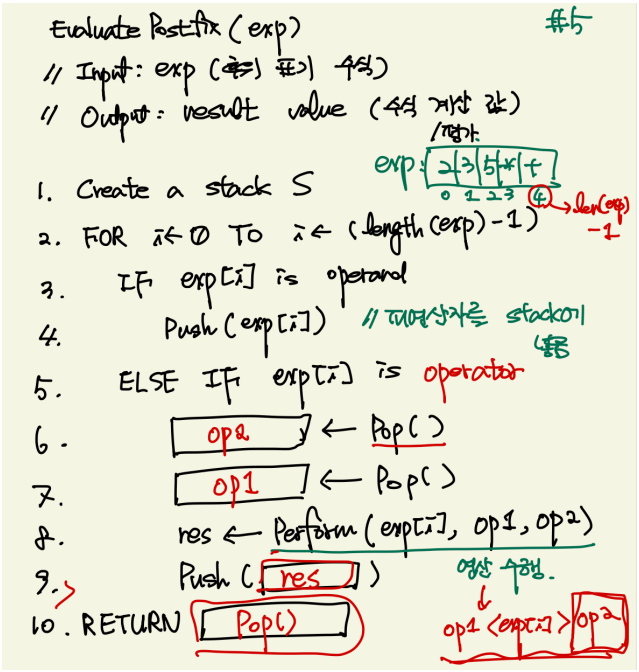
No

2. [Palindrome: 30 Pt] 회문(palindrome)이란 앞뒤 어느 쪽에서 읽어도 같은 단어를 의미한다. 예를 들면 "eye", "madam, I'm Adam", "race car" 등이다 (특수 문자와 공백 문자, 대소문자 등은 무시). 스택을 이용하여 문자열이 회문인지 아닌지를 결정하는 프로그램을 작성하시오 (*Hint: 스택에 Push하기 전에 입력 문자열을 모두 소문자로 바꾸고 공백 문자와 특수 문자 등은 스택에 넣지 않음).

Sample results

>> Enter a string: **madam, I'm Adam**
>> Palindrome!

3. [Infix to Postfix Conversion : 10 Pt] 아래는 후위 표기 수식을 입력으로 받아 이를 평가하는 의사 코드이다. 여기서 만약 입력이 후위 표기가 아닌 전위 표기 수식이라면 의사 코드의 어느 부분을 수정해야 하는가? 수정된 부분만 표시하시오.



4. [Evaluation of Postfix Expression :40 Pt] 후위 표기 수식 (Postfix expression) 으로 주어진 문자열 S를 평가 (evaluate)하는 프로그램을 작성한다. 단, 수식에서 피연산자 (operand)는 0~9 사이의 숫자로만 이루어지며, 4가지 종류의 연산자 (operators)를 포함한다 : **{+,-*,/}**.

Input

0~9 사이의 숫자를 포함하는 후위 표현 수식 문자열 (예: “24+”에서 2와 4는 각각 피연산자이고, +는 연산자이다)이며, 오류가 없는 valid한 수식만 입력으로 들어온다고 가정한다. (단, 나눗셈의 경우 0으로 나누어지는 경우는 예외처리해야 함)

Output

평가된 수식 (소수점 2자리로 표현한 실수).

Sample Input

2

24+

342+/

Sample Output

2.00

6.00

0.50

5.[Print in Circular Queue; 20 pt] 클래스룸에 업로드된 원형 큐 소스 코드 (cir_queue.c)에서 **void queue_print(QueueType *q)** 함수를 완성 후, 소스 코드의 main() 함수를 실행하여 원형 큐의 내용이 잘 출력되는지 확인하시오 (단, 원형 큐의 초기화는 front = rear = -1로 하고, enqueue()와 dequeue() 함수는 교재의 방식이 아닌 주어진 소스 코드를 따른다고 가정).

6. [Count in Circular Queue: 20 pt] 5번 문제의 출력 함수를 수정하여 **int get_count(QueueType *q, int x)** 함수를 추가하시오. get_count() 함수는 원형 큐 내에서 특정 정수 값 x 의 개수를 세어서 반환하는 함수로서, 찾는 정수 값이 없을 경우 0을 반환하고 큐가 비어 있을 경우는 -1을 반환한다. 함수를 구현한 후 다음과 같이 큐에 입력과 출력을 반복한 후, 3과 7의 개수를 구하는 main() 함수를 실행시켜 함수가 잘 동작하는지 확인한다 (각각 2와 0을 반환해야 함).

Enqueue(1) -> Enqueue(2) -> Enqueue(3) -> Enqueue(4) -> Dequeue() -> Enqueue(3) -> Dequeue() -> Enqueue(5) -> Enqueue(6)