2021 겨울 신촌 연합 알고리즘 캠프

스택, 큐, 덱

초급 알고리즘

HI-ARC 김기선

스택

LIFO (Last In First Out)

나중에 들어간 게 처음으로 나온다.

Ex) 프링글스

STL stack 주요 기능

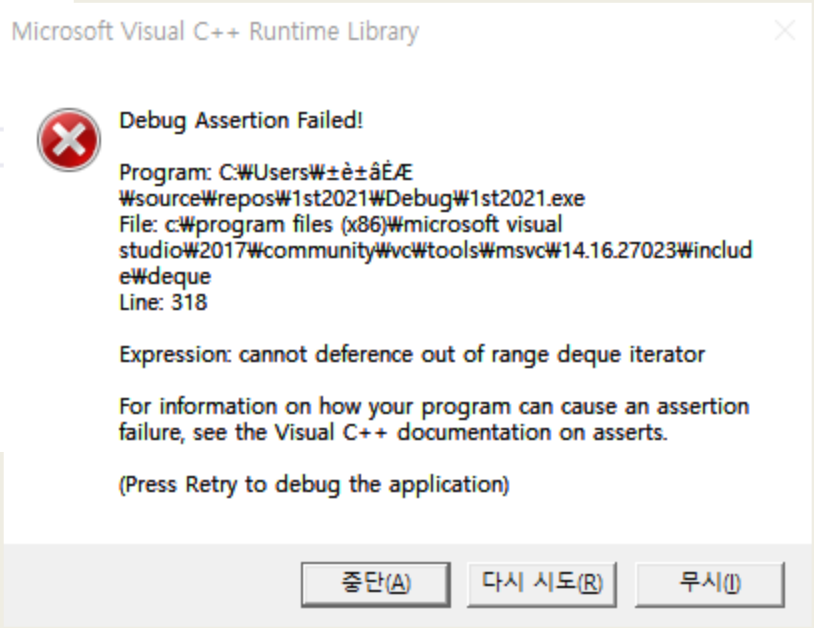
- 삽입 push()
- 삭제 pop()
- 제일 상단의 원소 확인 top()
- 스택의 사이즈 확인 size()
- 스택이 비었는지 확인 empty()

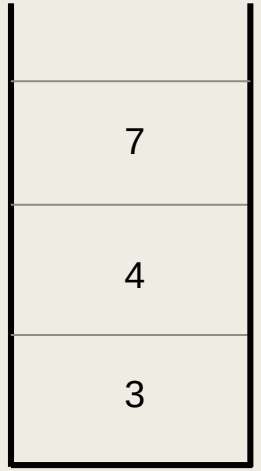
STL stack

```cpp
#include<iostream>
#include<stack>
using namespace std;

stack<int> st;

int main() {
    cout << st.top();
    st.pop();
    cout << st.size() <<'\n';
    st.push(3);
    st.push(4);
    st.push(7);
    st.pop();
    cout << st.top() << '\n';
    cout << st.size() << '\n';
}
```





백준 9012 괄호

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1 초	128 MB	73797	31523	22709	41.543%

문제

괄호 문자열(Parenthesis String, PS)은 두 개의 괄호 기호인 '(' 와 ')' 만으로 구성되어 있는 문자열이다. 그 중에서 괄호의 모양이 바르게 구성된 문자열을 올바른 괄호 문자열(Valid PS, VPS)이라고 부른다. 한 쌍의 괄호 기호로 된 "( )" 문자열은 기본 VPS 이라고 부른다. 만일 x 가 VPS 라면 이것을 하나의 괄호에 넣은 새로운 문자열 "(x)"도 VPS 가 된다. 그리고 두 VPS x 와 y를 접합(concatenation)시킨 새로운 문자열 xy도 VPS 가 된다. 예를 들어 "(())()"와 "((()))" 는 VPS 이지만 "(()(", "(())())" , 그리고 "((()" 는 모두 VPS 가 아닌 문자열이다.

여러분은 입력으로 주어진 괄호 문자열이 VPS 인지 아닌지를 판단해서 그 결과를 YES 와 NO 로 나타내어야 한다.

입력

입력 데이터는 표준 입력을 사용한다. 입력은 T개의 테스트 데이터로 주어진다. 입력의 첫 번째 줄에는 입력 데이터의 수를 나타내는 정수 T가 주어진다. 각 테스트 데이터의 첫째 줄에는 괄호 문자열이 한 줄에 주어진다. 하나의 괄호 문자열의 길이는 2 이상 50 이하이다.

출력

출력은 표준 출력을 사용한다. 만일 입력 괄호 문자열이 올바른 괄호 문자열(VPS)이면 "YES", 아니면 "NO"를 한 줄에 하나씩 차례대로 출력해야 한다.

백준 9012 괄호

() -> O
()() -> O
((()))(()) -> O

(-> X
) -> X
((())) -> X

백준 9012 괄호

1. 여는 괄호가 나오면 스택에 추가
2. 닫는 괄호가 나오면 스택의 top을 확인
   2.1 여는 괄호면 스택의 top을 제거
   2.2 닫는 괄호면 VPS가 아님
   2.3 비어 있으면 VPS가 아님

3. 문자열이 끝날 때까지 비교 후 스택에 괄호가 남아있으면 VPS가 아님
   남아있지 않으면 VPS

백준 9012 괄호

```cpp
#include<iostream>
#include<string>
#include<stack>
using namespace std;

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    int n;
    cin >> n;
    string s;
    for (int i = 0; i < n; i++) {
        cin >> s;
        stack<char> st;
        bool flag = true;
        for (int j = 0; j < s.size(); j++) {
            if (s[j] == '(')st.push('(');
            else if (s[j] == ')') {
                if (!st.empty() && st.top() == '(')
                    st.pop();
                else {
                    flag = false;
                    break;
                }
            }
        }
        if (st.empty() && flag)cout << "YES\n";
        else cout << "NO\n";
    }
}
```

백준 1935 후위 표기식2

A*(B+C) -> ABC+*
A*B+C -> AB*C+
A+B*C+D*E+G -> ABC*+DE*+G+

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
2 초	128 MB	6540	2927	2324	44.917%

문제

후위 표기식과 각 피연산자에 대응하는 값들이 주어져 있을 때, 그 식을 계산하는 프로그램을 작성하시오.

입력

첫째 줄에 피연산자의 개수(1 ≤ N ≤ 26) 가 주어진다. 그리고 둘째 줄에는 후위 표기식이 주어진다. (여기서 피연산자는 A~Z의 영대문자이며, A부터 순서대로 N개의 영대문자만이 사용되며, 길이는 100을 넘지 않는다) 그리고 셋째 줄부터 N+2번째 줄까지는 각 피연산자에 대응하는 값이 주어진다. (3번째 줄에는 A에 해당하는 값, 4번째 줄에는 B에 해당하는값 , 5번째 줄에는 C ...이 주어진다, 그리고 피연산자에 대응 하는 값은 정수이다)

출력

계산 결과를 소숫점 둘째 자리까지 출력한다.

백준 1935 후위 표기식2

예제 입력 1 복사

```
5
ABC*+DE/-
1
2
3
4
5
```

예제 출력 1 복사

```
6.20
```

예제 입력 2 복사

```
1
AA+A+
1
```

예제 출력 2 복사

```
3.00
```

1. 피연산자가 나오면 스택에 삽입
2. 연산자가 나오면 스택에서 상위 두개의 수를 꺼냄
3. 계산(순서를 잘 생각해서)

백준 1935 후위 표기식2

```cpp
#include<iostream>
#include<stack>
#include<string>
using namespace std;

string s;
double num[26];

int main() {
    cout << fixed;
    cout.precision(2);
    int n;
    cin >> n;
    cin >> s;
    for (int i = 0; i < n; i++) cin >> num[i];
    stack<double> st;
    for (int i = 0; i < s.length(); i++) {
        if (s[i] == '*' || s[i] == '/' || s[i] == '+' || s[i] == '-') {
            double a = st.top();
            st.pop();
            double b = st.top();
            st.pop();
            if (s[i] == '*') {
                st.push((1.0)*b*a);
            }
            else if (s[i] == '/') {
                st.push((1.0)*b / a);
            }
            else if (s[i] == '+') {
                st.push(b + a);
            }
            else if (s[i] == '-') {
                st.push(b - a);
            }
        }
        else {
            st.push(num[s[i] - 'A']);
        }
    }
    cout << st.top();
}
```

큐

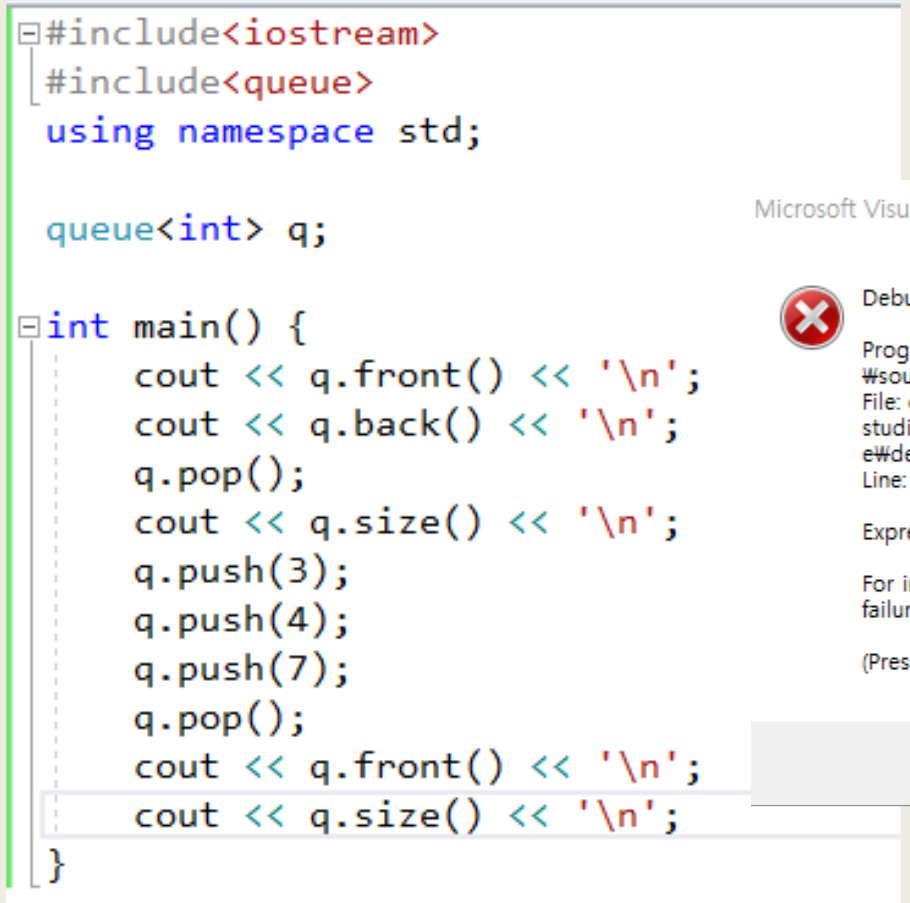
FIFO (First In First Out)

처음에 들어간 게 처음으로 나온다. (우선순위가 같다.)

STL queue 주요 기능

- 삽입 push()
- 삭제 pop()
- 제일 앞/뒤에 있는 원소 확인 front() / back()
- 큐의 사이즈 확인 size()
- 큐가 비었는지 확인 empty()

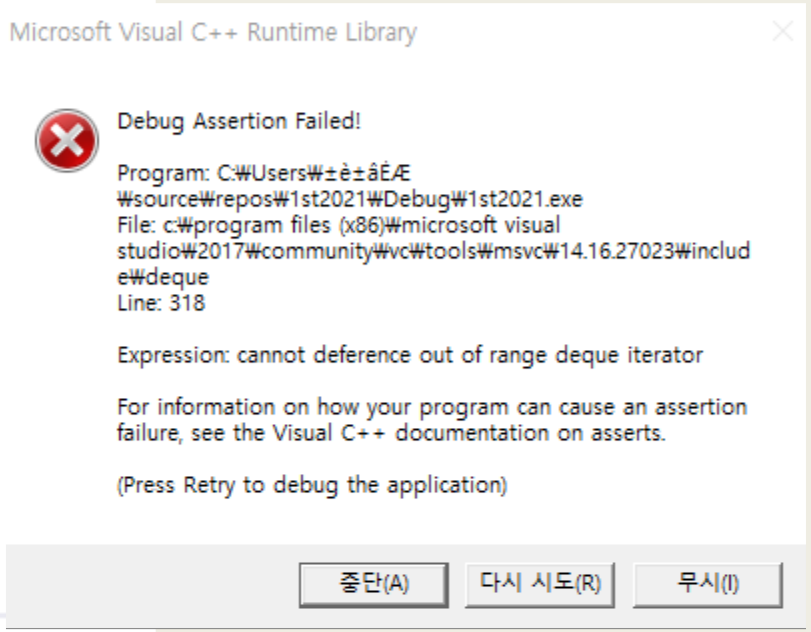
STL queue

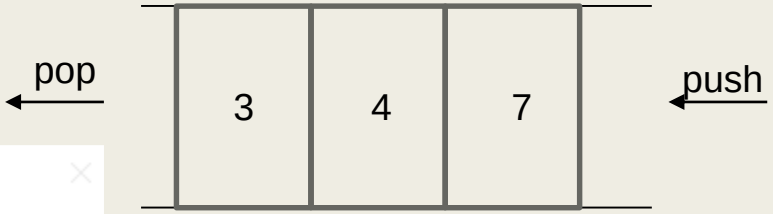


```cpp
#include<iostream>
#include<queue>
using namespace std;

queue<int> q;

int main() {
    cout << q.front() << '\n';
    cout << q.back() << '\n';
    q.pop();
    cout << q.size() << '\n';
    q.push(3);
    q.push(4);
    q.push(7);
    q.pop();
    cout << q.front() << '\n';
    cout << q.size() << '\n';
}
```





백준 1158 요세푸스 문제

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
2 초	256 MB	36333	17647	12819	48.877%

문제

요세푸스 문제는 다음과 같다.

1번부터 N번까지 N명의 사람이 원을 이루면서 앉아있고, 양의 정수 K(≤ N)가 주어진다. 이제 순서대로 K번째 사람을 제거한다. 한 사람이 제거되면 남은 사람들로 이루어진 원을 따라 이 과정을 계속해 나간다. 이 과정은 N명의 사람이 모두 제거될 때까지 계속된다. 원에서 사람들이 제거되는 순서를 (N, K)-요세푸스 순열이라고 한다. 예를 들어 (7, 3)-요세푸스 순열은 <3, 6, 2, 7, 5, 1, 4>이다.

N과 K가 주어지면 (N, K)-요세푸스 순열을 구하는 프로그램을 작성하시오.

입력

첫째 줄에 N과 K가 빈 칸을 사이에 두고 순서대로 주어진다. (1 ≤ K ≤ N ≤ 5,000)

출력

예제와 같이 요세푸스 순열을 출력한다.

예제 입력 1 복사

```
7 3
```

예제 출력 1 복사

```
<3, 6, 2, 7, 5, 1, 4>
```

백준 1158 요세푸스 문제

```cpp
#include<iostream>
#include<queue>
using namespace std;
queue<int> q;
int main() {
    int n, k;
    cin >> n >> k;
    for (int i = 1; i <= n; i++)q.push(i);
    int cnt = 0;
    cout << "<";
    while (!q.empty()) {
        int a = q.front();
        q.pop();
        cnt++;
        if (cnt == k) {
            if (q.size() == 0) {
                cout << a << ">";
            }
            else cout << a << ", ";
            cnt = 0;
        }
        else q.push(a);
    }
}
```

백준 19591 독특한 계산기

당신은 수식을 독특한 방식으로 계산해야 한다. 수식을 계산하는 방식은 다음과 같다.

1. 수식에서 맨 앞의 연산자, 또는 맨 뒤의 연산자 먼저 계산한다. 단, 음수의 부호는 연산자로 취급하지 않는다.
2. 곱셈, 나눗셈을 덧셈, 뺄셈보다 더 먼저 계산한다.
3. 연산자의 우선순위가 같다면 해당 연산자를 계산했을 때 결과가 큰 것부터 계산한다.
4. 계산했을 때 결과 값 또한 같다면 앞에 것을 먼저 계산한다.

예를 들어서 수식이 3 × 2 + 5 − 5 + 7으로 주어진다고 하면 다음과 같이 계산된다.

1. 3 × 2와 5 + 7 중에서 계산 우선순위가 더 높은 ×를 먼저 계산한다. 이후 계산식은 6 + 5 − 5 + 7이다.
2. 앞뒤의 연산자가 같으므로 6 + 5와 5 + 7을 비교했을 때, 5 + 7이 더 크기 때문에 뒤에 있는 +를 먼저 계산한다. 이후 계산식은 6 + 5 − 12이다.
3. 뺄셈과 덧셈의 우선순위가 같으므로 6 + 5와 5 − 12를 비교했을 때, 6 + 5가 더 크기 때문에 +를 먼저 계산한다. 이후 계산식은 11 − 12가 된다.
4. 11 − 12를 계산하면 최종 결과 값은 −1이 된다.

수식은 반드시 수와 연산자가 번갈아 가면서 나온다. 마지막에 연산자가 있는 경우는 존재하지 않으며, 맨 앞을 제외하고 음수가 들어오는 경우도 존재하지 않는다. 즉, −1 − 1 같은 경우는 나올 수 있으나, 2 + −3 같은 경우는 존재하지 않는다고 가정해도 된다. 그리고 불필요한 0이 앞에 있을 수 있다. 즉, 001 + 0002 같은 수식이 나올 수 있다.

또한, 이 문제에서의 나눗셈은 C++에서 정수 간에 정의된 나눗셈으로 생각한다. 즉, 나누어지는 수가 양수면 나머지가 0 이상, 음수면 나머지가 0 이하로 처리가 되는 식으로 진행했을 때 나오는 몫을 계산하는 방식으로 이루어진다. 예를 들어, 3 / 2 = 1, (−3) / 2 = −1, 3 / (−2) = −1, (−3) / (−2) = 1로 계산된다.

이와 같은 계산 과정에 따라 주어진 식을 계산하시오.

입력

숫자, '+', '*', '-', '/'로만 이루어진 길이가 10^6 이하인 수식이 주어진다. 계산 과정 중의 모든 수는 -2^{63} 이상 2^{63} 미만이며, 0으로 나누는 경우는 없다. 숫자 앞에 불필요한 0이 있을 수 있다.

출력

주어진 식을 계산한 결과 값을 출력한다. 불필요한 0은 제거해야 한다.

덱(Double Ended Queue)

- 처음과 끝에서 삽입 삭제가 일어나는 경우에 쓰임
- 중간 원소에 접근이 가능은 하지만 삽입 삭제에 있어서 효율성이 매우 떨어짐

STL deque 주요 기능

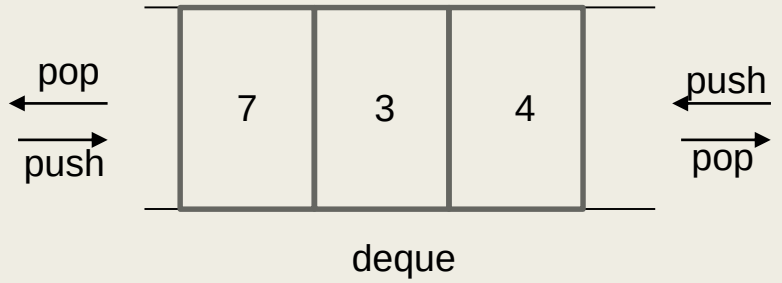
- 삽입 push_front()/push_back()
- 삭제 pop_front()/pop_back()
- 제일 앞/뒤에 있는 원소 확인 front() / back()
- 덱의 사이즈 확인 size()
- 덱이 비었는지 확인 empty()

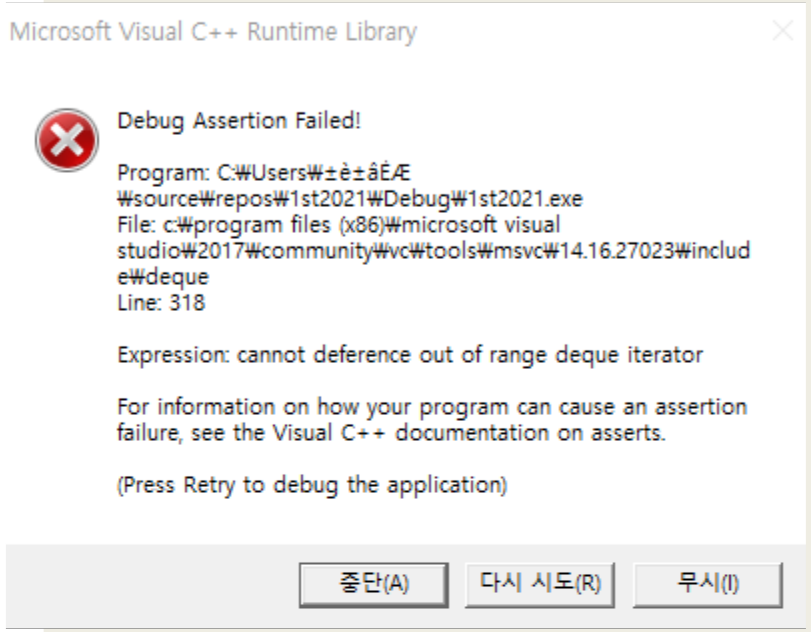
STL deque

```cpp
#include<iostream>
#include<deque>
using namespace std;

deque<int> dq;

int main() {
    cout << dq.front() << '\n';
    cout << dq.back() << '\n';
    dq.pop_front();
    dq.pop_back();
    cout << dq.size() << '\n';
    dq.push_front(3);
    dq.push_back(4);
    dq.push_front(7);
    cout << dq[2] << '\n';
    dq.pop_front();
    cout << dq.front() << '\n';
    cout << dq.size() << '\n';
}
```





백준 19591 독특한 계산기

당신은 수식을 독특한 방식으로 계산해야 한다. 수식을 계산하는 방식은 다음과 같다.

1. 수식에서 맨 앞의 연산자, 또는 맨 뒤의 연산자 먼저 계산한다. 단, 음수의 부호는 연산자로 취급하지 않는다.
2. 곱셈, 나눗셈을 덧셈, 뺄셈보다 더 먼저 계산한다.
3. 연산자의 우선순위가 같다면 해당 연산자를 계산했을 때 결과가 큰 것부터 계산한다.
4. 계산했을 때 결과 값 또한 같다면 앞에 것을 먼저 계산한다.

예를 들어서 수식이 3 × 2 + 5 − 5 + 7으로 주어진다고 하면 다음과 같이 계산된다.

1. 3 × 2와 5 + 7 중에서 계산 우선순위가 더 높은 ×를 먼저 계산한다. 이후 계산식은 6 + 5 − 5 + 7이다.
2. 앞뒤의 연산자가 같으므로 6 + 5와 5 + 7을 비교했을 때, 5 + 7이 더 크기 때문에 뒤에 있는 +를 먼저 계산한다. 이후 계산식은 6 + 5 − 12이다.
3. 뺄셈과 덧셈의 우선순위가 같으므로 6 + 5와 5 − 12를 비교했을 때, 6 + 5가 더 크기 때문에 +를 먼저 계산한다. 이후 계산식은 11 − 12가 된다.
4. 11 − 12를 계산하면 최종 결과 값은 −1이 된다.

수식은 반드시 수와 연산자가 번갈아 가면서 나온다. 마지막에 연산자가 있는 경우는 존재하지 않으며, 맨 앞을 제외하고 음수가 들어오는 경우도 존재하지 않는다. 즉, −1 − 1 같은 경우는 나올 수 있으나, 2 + −3 같은 경우는 존재하지 않는다고 가정해도 된다. 그리고 불필요한 0이 앞에 있을 수 있다. 즉, 001 + 0002 같은 수식이 나올 수 있다.

또한, 이 문제에서의 나눗셈은 C++에서 정수 간에 정의된 나눗셈으로 생각한다. 즉, 나누어지는 수가 양수면 나머지가 0 이상, 음수면 나머지가 0 이하로 처리가 되는 식으로 진행했을 때 나오는 몫을 계산하는 방식으로 이루어진다. 예를 들어, 3 / 2 = 1, (−3) / 2 = −1, 3 / (−2) = −1, (−3) / (−2) = 1로 계산된다.

이와 같은 계산 과정에 따라 주어진 식을 계산하시오.

입력

숫자, '+', '*', '-', '/'로만 이루어진 길이가 10^6 이하인 수식이 주어진다. 계산 과정 중의 모든 수는 -2^{63} 이상 2^{63} 미만이며, 0으로 나누는 경우는 없다. 숫자 앞에 불필요한 0이 있을 수 있다.

출력

주어진 식을 계산한 결과 값을 출력한다. 불필요한 0은 제거해야 한다.

백준 19591 독특한 계산기

3*2+5-5+7

1. 연산자와 피연산자를 두개의 덱에 나눠서 저장
2. 연산자를 앞에서 하나 뒤에서 하나 뽑고, 피연산자도 앞에서 두개 뒤에서 두개 뽑아서 계산(연산자 우선순위, 같으면 크기 비교, 크기 같으면 앞에 꺼 연산)
3. 사용한 연산자는 아예 pop하고, 사용한 피연산자도 pop하고 새로운 값을 push

필수문제

- A - 균형잡힌 세상
- B - 후위 표기식2
- C - 요세푸스 문제
- D - 프린터 큐
- E - 회전하는 큐

연습문제

- A - 괄호
- B - 괄호의 값
- C - 쇠막대기
- D - 후위 표기식
- E - 크게 만들기
- F - 문자열 폭발
- G - 오큰수
- H - 오등큰수
- I - 히스토그램
- J - 카드2
- K - 독특한 계산기
- L - 최솟값 찾기
- M - XML
- N - Brainf*ck

피드백 및 질의응답

감사합니다!