

# 알고리즘 문제풀이

## 5. 괄호

## 1. 올바른 괄호

### 설명

괄호가 입력되면 올바른 괄호이면 "YES", 올바르지 않으면 "NO"를 출력합니다.


(())() 이것은 괄호의 쌍이 올바르게 위치하는 거지만, (()())은 올바른 괄호가 아니다.

### 입력

첫 번째 줄에 괄호 문자열이 입력됩니다. 문자열의 최대 길이는 30이다.

### 출력

첫 번째 줄에 YES, NO를 출력한다.

예시 입력 1 

((()()))()

예시 출력 1

NO

```
public String solution(String str){  
    String answer="YES";  
    Stack<Character> stack=new Stack<>();  
    for(char x : str.toCharArray()){  
        if(x=='(') stack.push(x);  
        else{  
            if(stack.isEmpty()) return "NO";  
            stack.pop();  
        }  
    }  
    if(!stack.isEmpty()) return "NO";  
    return answer;  
}
```

## 2. 괄호문자제거

### 설명

입력된 문자열에서 소괄호 ( ) 사이에 존재하는 모든 문자를 제거하고 남은 문자만 출력하는 프로그램을 작성하세요.

### 입력

첫 줄에 문자열이 주어진다. 문자열의 길이는 100을 넘지 않는다.

### 출력

남은 문자만 출력한다.

### 예시 입력 1

```
(A(BC)D)EF(G(H)(IJ)K)LM(N)
```

### 예시 출력 1

```
EFLM
```

(A(BC)D)EF(G(H)(IJ)K)LM(N)

```

public String solution(String str){
    String answer="";
    Stack<Character> stack=new Stack<>();
    for(char x : str.toCharArray()){
        if(x==''){
            while(stack.pop()!='(');
        }
        else stack.push(x);
    }
    for(int i=0; i<stack.size(); i++) answer+=stack.get(i);
    return answer;
}

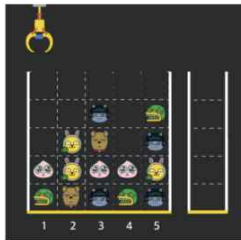
```

### 3. 크레인 인형뽑기(카카오)

#### 설명

게임개발자인 조르디는 크레인 인형뽑기 기계를 모바일 게임으로 만들려고 합니다.

조르디는 게임의 재미를 높이기 위해 화면 구성과 규칙을 다음과 같이 게임 로직에 반영하려고 합니다.



게임 화면은  $1 \times 1$  크기의 칸들로 이루어진  $N \times N$  크기의 정사각 격자이며 위쪽에는 크레인이 있고 오른쪽에는 바구니가 있습니다.

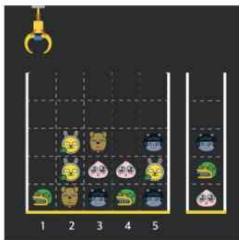
(위 그림은  $5 \times 5$  크기의 예시입니다). 각 격자 칸에는 다양한 인형이 들어 있으며 인형이 없는 칸은 빈칸입니다.

모든 인형은  $1 \times 1$  크기의 격자 한 칸을 차지하며 격자의 가장 아래 칸부터 차곡차곡 쌓여 있습니다.

게임 사용자는 크레인을 좌우로 움직여서 멈춘 위치에서 가장 위에 있는 인형을 집어 올릴 수 있습니다. 집어 올린 인형은 바구니에 쌓이게 되는데,

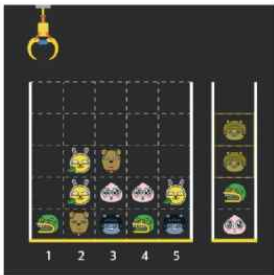
이때 바구니의 가장 아래 칸부터 인형이 순서대로 쌓이게 됩니다.

다음 그림은 [1번, 5번, 3번] 위치에서 순서대로 인형을 집어 올려 바구니에 담은 모습입니다.



만약 같은 모양의 인형 두 개가 바구니에 연속해서 쌓이게 되면 두 인형은 터뜨려지면서 바구니에서 사라지게 됩니다.

위 상태에서 이어서 [5번] 위치에서 인형을 집어 바구니에 쌓으면 같은 모양 인형 두 개가 없어집니다.



크레인 작동 시 인형이 집어지지 않는 경우는 없으나 만약 인형이 없는 곳에서 크레인을 작동시키는 경우에는 아무런 일도 일어나지 않습니다.

또한 바구니는 모든 인형이 들어갈 수 있을 만큼 충분히 크다고 가정합니다. (그림에서는 화면표시 제약으로 5칸만으로 표현하였음)

게임 화면의 격자의 상태가 담긴 2차원 배열 board와 인형을 집기 위해 크레인을 작동시킨 위치가 담긴 배열 moves가 매개변수로 주어질 때, 크레인을 모두 작동시킨 후 터트려져 사라진 인형의 개수를 구하는 프로그램을 작성하세요.



## 입력

첫 줄에 자연수  $N(5 \leq N \leq 30)$ 이 주어집니다.

두 번째 줄부터  $N \times N$  board 배열이 주어집니다.

board의 각 칸에는 0 이상 100 이하인 정수가 담겨있습니다.

0은 빈 칸을 나타냅니다.

1 ~ 100의 각 숫자는 각기 다른 인형의 모양을 의미하며 같은 숫자는 같은 모양의 인형을 나타냅니다.

board배열이 끝난 다음줄에 moves 배열의 길이  $M$ 이 주어집니다.

마지막 줄에는 moves 배열이 주어집니다.

moves 배열의 크기는 1 이상 1,000 이하입니다.

moves 배열 각 원소들의 값은 1 이상이며 board 배열의 가로 크기 이하인 자연수입니다.

## 출력

첫 줄에 터트려져 사라진 인형의 개수를 출력합니다.

## 예시 입력 1

```
5
0 0 0 0 0
0 0 1 0 3
0 2 5 0 1
4 2 4 4 2
3 5 1 3 1
8
1 5 3 5 1 2 1 4
```

## 예시 출력 1

```
4
```

```
5
00000
00103
02501
42442
35131
8
15351214
```



8 >> 8번 뽑는 순서  
1 5 3 5 1 2 1 4

```
public static void main(String[] args){
    Main T = new Main();
    Scanner kb = new Scanner(System.in);
    int n=kb.nextInt();
    int[][] board=new int[n][n];
    for(int i=0; i<n; i++){
        for(int j=0; j<n; j++){
            board[i][j]=kb.nextInt();
        }
    }
    int m=kb.nextInt();
    int[] moves=new int[m];
    for(int i=0; i<m; i++) moves[i]=kb.nextInt();
    System.out.println(T.solution(board, moves));
}
```

```

public int solution(int[][] board, int[] moves){
    int answer=0;
    Stack<Integer> stack = new Stack<>();
    for(int pos : moves){
        for(int i=0; i<board.length; i++){
            if(board[i][pos-1]!=0){
                int tmp=board[i][pos-1];
                board[i][pos-1]=0;
                if(!stack.isEmpty() && tmp==stack.peek()){
                    answer+=2;
                    stack.pop();
                }
                else stack.push(tmp);
                break;
            }
        }
    }
    return answer;
}

```