팀 선정 문제 – 쇠막대기

문제를 풀긴 했지만 개인적으로 접근했을 때 굉장히 헷갈렸던 문제이다.

스택을 활용하여 여는 괄호 '(' 닫는 괄호 ')'를 통해 레이저인지, 쇠막대기인지 확인 하는게 가장 큰 포인트이며 쇠막대기일 경우 쇠막대기의 시작과 끝을 판별 하는게 매우 헷갈렸다.

1. 쇠막대기

10799번: 쇠막대기

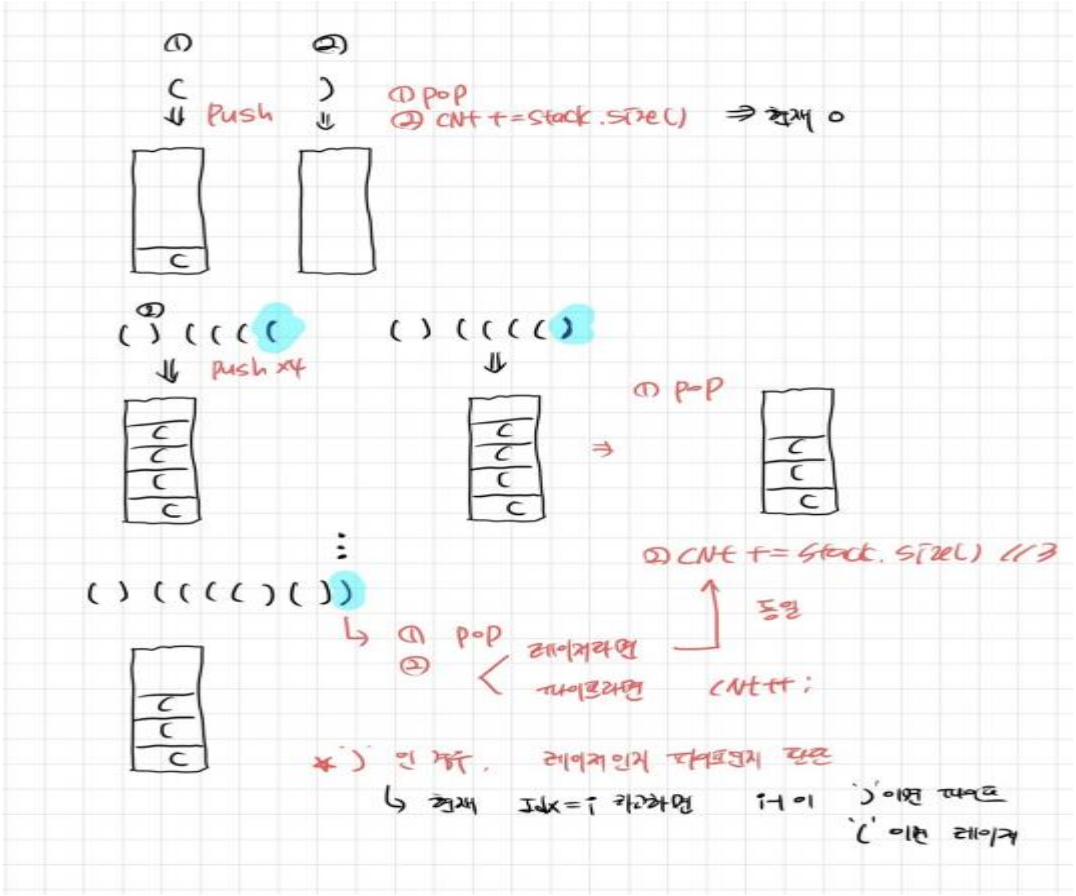
여러 개의 쇠막대기를 레이저로 절단하려고 한다. 효율적인 작업을 위해서 쇠막대기를 아래에서 위로 겹쳐 놓고, 레이저를 위에서 수직으로 발사하여 쇠막대기들을 자른다. 쇠막대기와 레이저의 배치는 다음 조건을 만

https://www.acmicpc.net/problem/10799

BAE/KJOON ONLINE JUDGE

▼ 접근방법

- (: 스택에 push
-) : 스택 값을 pop하고, ')' 레이저인지, 파이프인지 판단해줘야함.
  - 현 위치:i 그 이전 위치: i-1 모두가) 인 경우 (즉 ')) ' 이라면) 현재 i는 파이프이다. 파이프 끝이므로, cnt+1해줘야한다.
  - 현 위치:i 그 이전 위치: i-1 이 다른 경우, (즉 ' () ' 이라면) 현재 i는 레이저이다. 현재 스택에 쌓여진 사이즈만큼 cnt+=stack.size()해주어야함.



```
  public static void main(String[] args) throws IOException {
    // TODO Auto-generated method stub

    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    Stack<Character> stack = new Stack<>();

    String str = br.readLine();
    char[] arr = str.toCharArray();

    int cnt = 0;
    stack.push(arr[0]);
    for (int i = 1; i < arr.length; i++) {
      char current = arr[i];

      if (current == '(')
        stack.push(current);

      if (current == ')') { // )인 경우, 그 앞전의 값을 비교하여 지금 )이 파이프인지 레이저인지 판단해야함
        char before = arr[i - 1];
        stack.pop();

        if (before == ')') { // 그 앞전에 레이저였음. 지금은 파이프가 닫힌것
          cnt++; // 막대기 하나가 끝나서 끝 부분만 남은 상태이므로 +1 필수
        }
        else { // 그 앞 (이었으며 현재 )인 상태
          cnt += stack.size(); // 스택사이즈만큼 현재 쇠막대기 수를 카운트 해줌
        }
      }
    }
    System.out.println(cnt);
  }
}
```

초급자 – Stack 20001번 고무오리 디버깅

문제

백준 문제 풀이에 힘들어하는 수진이를 위해 민우는 문제해결에 도움이 되는 고무오리를 준비했다. 민우가 준비한 고무오리는 신비한 능력이 존재하는데, 최근에 풀던 백준 문제를 해결해주는 능력이다. 신비한 고무오리와 함께 수진이의 백준 풀이를 도와주자!

고무오리의 사용법은 다음과 같다.

- "고무오리 디버깅 시작" 이라고 외친다
- 문제들을 풀기 시작한다
- 고무오리를 받으면 최근 풀던 문제를 해결한다
- "고무오리 디버깅 끝" 이라고 외치면 문제풀이를 종료한다.

하지만 고무오리에는 치명적인 문제가 있는데, 풀 문제가 없는데 사용한다면 고무오리는 체벌로 두 문제를 추가한다는 점이다.

입력

첫 번째 줄에 "고무오리 디버깅 시작"이라고 주어진다. 두 번째 줄부터 "고무오리" 또는 "문제"가 주어진다. 이는 "고무오리 디버깅 끝"이 주어질 때까지 반복한다. 최대 102줄이 입력으로 주어진다.

출력

고무오리 디버깅이 끝날 때, 주어진 문제를 수진이가 해결하였는지 여부에 따라 남은 문제 없이 모든 문제를 해결하였으면 "고무오리야 사랑해"을 출력하고 하나라도 문제가 남았다면 "힝구"를 출력하라.

스택을 활용하여 고무오리와 문제를 입력 받으면 상쇄시켜 없애는 문제이다.

예외 조건은 스택에 아무것도 없을 때 고무오리가 들어온다면 문제2개와 같은 효과인 것이다.

1) Stack.size 스택의 크기(길이)

2) stact.push 스택에 값 입력(후입선출)

3) stack.pop 스택에 값 삭제(후입선출)

4) stack.isEmpty 스택이 비어 있는지 true false 반환

```java
package day0812;

import java.io.BufferedReader;

public class BaekJoon_고무오리_디버깅 {
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        Stack<Integer> stack = new Stack<Integer>();
        while (true) {
            String str = br.readLine();
            if (str.equals("고무오리")) {     //  입력에 고무오리가 들어오지만
                if (stack.size() == 0) {    //  스택에 아무것도 없을때 고무오리가 들어오면 문제 2개 추가
                    stack.push(1);
                    stack.push(1);
                } else {                    //  스택에 무언가 있을때 고무오리가 들어오면 문제 1개 삭제(pop)
                    stack.pop();
                }
            } else if (str.equals("문제")) {  //  입력에 문제가 들어오면 문제 1개 추가
                stack.push(1);
            } else if (str.equals("고무오리 디버깅 끝")) {
                break;
            }
        }
        System.out.println(stack.isEmpty() ? "고무오리야 사랑해" : "힝구");
    }
}
```

초급자 – Queue 2161번 카드 1

문제

N장의 카드가 있다. 각각의 카드는 차례로 1부터 N까지의 번호가 붙어 있으며, 1번 카드가 제일 위에, N번 카드가 제일 아래인 상태로 순서대로 카드가 놓여 있다.

이제 다음과 같은 동작을 카드가 한 장 남을 때까지 반복하게 된다. 우선, 제일 위에 있는 카드를 바닥에 버린다. 그 다음, 제일 위에 있는 카드를 제일 아래에 있는 카드 밑으로 옮긴다.

예를 들어 N=4인 경우를 생각해 보자. 카드는 제일 위에서부터 1234 의 순서로 놓여있다. 1을 버리면 234가 남는다. 여기서 2를 제일 아래로 옮기면 342가 된다. 3을 버리면 42가 되고, 4를 밑으로 옮기면 24가 된다. 마지막으로 2를 버리고 나면, 버린 카드들은 순서대로 1 3 2가 되고, 남는 카드는 4가 된다.

N이 주어졌을 때, 버린 카드들을 순서대로 출력하고, 마지막에 남게 되는 카드를 출력하는 프로그램을 작성하시오.

입력

첫째 줄에 정수 N(1 ≤ N ≤ 1,000)이 주어진다.

출력

첫째 줄에 버리는 카드들을 순서대로 출력한다. 제일 마지막에는 남게 되는 카드의 번호를 출력한다.

큐에 입력 값을 넣고 while 문을 활용하여 큐가 빌 때 까지 작업을 반복시킨다.

가장 먼저 넣은 값을 변수에 따로 저장해 놓고 값을 삭제 시킨다.

값을 따로 저장한 변수는 출력해주고 두번째 값도 똑같이 변수에 저장하고 값을 삭제한다.

두번째 값은 다시 큐에 넣어야 하기 때문에 큐에 변수 값을 삽입시켜준다.

1) Queue.offer 값을 큐에 넣어준다(선입선출)

2) queue.poll 가장 먼저 넣은 값을 삭제 시킨다

3) queue.isEmpty 큐가 비어 있는지 true false 반환

```java
package day0812;

import java.io.BufferedReader;

public class BaekJoon_카드1 {

    public static void main(String[] args) throws NumberFormatException, IOException {

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        int N = Integer.parseInt(br.readLine());
        Queue<Integer> queue = new LinkedList<Integer>();
        for (int i = 1; i <= N; i++) {

            queue.offer(i); // 차례대로 1~N의 수를 큐에 넣는다.

        }

        while (!queue.isEmpty()) {

            int out = queue.poll(); // 가장 먼저 넣은 값을 빼준다. 이 값을 변수에 저장

            System.out.print(out + " ");

            if (queue.isEmpty()) // 큐가 비어있는지 판별

                break;

            int in = queue.poll(); // 두번째로 넣었던 값을 빼준다.

            queue.offer(in); // 두번째 값은 뒤에 다시 넣어야 하기 때문에 다시 offer하여 넣는다.

        }

    }

}
```