1-1). 20001 : 고무오리 디버깅

```java
public class Main1 {

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

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); // 입력을 문자열을 받기 때문에 bufferedreader 사용

        Stack<Boolean> stack = new Stack<>(); // "고무오리", "문제" 가 입력되었을 때 관리할 스택

        String str = br.readLine();

        if (str.equals("고무오리 디버깅 시작")) { // "고무오리 디버깅 시작" 문자열이 입력되면 반복문 시작
            while (true) {

                str = br.readLine();

                if (str.equals("고무오리")) { // "고무오리" 문자열이 들어올 경우 스택이 비어있으면 스택 2개 생성, 스택이 1개 이상이면 스택 한 개 삭제
                    if (stack.size() == 0) {
                        stack.push(true);
                        stack.push(true);
                    } else {
                        stack.pop();
                    }

                } else if (str.equals("문제")) { // "문제" 문자열이 입력이 되면 스택 한개 생성
                    stack.push(true);
                } else if (str.equals("고무오리 디버깅 끝")) { // 반복문 종료 조건문
                    break;
                }

            }

            if (stack.size() == 0) { // 스택이 비어있을 경우 출력
                System.out.println("고무오리야 사랑해");
            } else { // 스택이 1개 이상 남았을 경우 출력
                System.out.println("힝구");
            }
        }
    }
}
```

1-2) 17608 : 막대기

```java
public class Main2 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in); // 단순 숫자 입력이므로 Scanner을 이용하여 입력을 받음
        Stack<Integer> stack = new Stack<>();

        int num = sc.nextInt(); // 들어올 숫자의 개수 입력

        for (int i = 0; i < num; i++) {

            int height = sc.nextInt(); // 제일 왼쪽부터 층이 쌓이기 시작

            if (stack.size() == 0) { // 처음 쌓이는 경우에는 무조건 스택에 push
                stack.push(height);
            } else if (stack.peek() <= height) { // 앞선 층보다 들어오는 층이 더 높을 경우
                while (stack.peek() <= height) { // 이때까지 쌓였던 스택중에 들어오는 층보다 낮은 스택은 다 삭제
                    stack.pop();
                    if (stack.size() == 0) // 계속 삭제를 할 경우에는 예외가 발생하므로 남는게 아무 것도 없을 경우 반복문 종료
                        break;
                }
                stack.push(height);
            } else if (stack.peek() > height) { // 앞선 층보다 들어오는 층이 더 낮을 경우 스택에 push
                stack.push(height);
            }

        }
        System.out.println(stack.size());
    }

}
```

1-3) 2161 : 카드1

```java
public class Main3 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in); // 단순 숫자 입력이므로 Scanner을 통하여 입력 받기
        Queue<Integer> queue = new LinkedList<Integer>(); // 큐 자료구조 생성

        int n = sc.nextInt(); // 1~n까지의 숫자카드 범위 지정

        for (int i = 1; i <= n; i++) { // queue에 1부터 n까지 순서대로 넣기
            queue.offer(i);
        }

        while (!(queue.size() == 1)) { // queue 사이즈가 1이 될때까지 반복
            System.out.println(queue.poll()); // 제일 윗 장에 있는 카드 번호 출력
            queue.offer(queue.poll()); // 그 다음에 있는 카드를 삭제함과 동시에 제일 마지막 카드로 넣기
        }

        System.out.println(queue.poll()); // 반복문 탈출후 마지막 남은 한 장 숫자 출력

    }

}
```

1-4) 2164 : 카드2

```java
public class Main4 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in); // 단순 숫자 입력이므로 Scanner을 통하여 입력 받기
        Queue<Integer> queue = new LinkedList<Integer>(); // 큐 자료구조 생성

        int n = sc.nextInt(); // 1~n까지의 숫자카드 범위 지정

        for (int i = 1; i <= n; i++) { // queue에 1부터 n까지 순서대로 넣기
            queue.offer(i);
        }

        while (!(queue.size() == 1)) { // queue 사이즈가 1이 될때까지 반복
            queue.poll(); // 제일 윗 장 카드 버리기
            queue.offer(queue.poll()); // 그 다음에 있는 카드를 삭제함과 동시에 제일 마지막 카드로 넣기
        }

        System.out.println(queue.poll()); // 반복문 탈출후 마지막 남은 한 장 숫자 출력

    }

}
```