

# 에꼴캐스톤디자인 코딩 테스트 리뷰

컴퓨터 공학과 김민성

# 목차

01 수 정렬하기 (2750)

02 숫자의 합 구하기(11720)

03 구간 합 구하기 (11659)

04 연속된 자연수의 합 구하기 (2018)

05 '좋은 수' 구하기(1253)

06 최솟값 구하기 (11003)

07 스택으로 오름차순  
수열 만들기 (1874)

08 카드게임 (2164)

# 01 수 정렬하기 (2750)

## 문제

N개의 수가 주어졌을 때, 이를 오름차순으로 정렬하는 프로그램을 작성하시오.

## 입력

첫째 줄에 수의 개수  $N(1 \leq N \leq 1,000)$ 이 주어진다. 둘째 줄부터 N개의 줄에는 수가 주어진다. 이 수는 절댓값이 1,000보다 작거나 같은 정수이다. 수는 중복되지 않는다.

## 출력

첫째 줄부터 N개의 줄에 오름차순으로 정렬한 결과를 한 줄에 하나씩 출력한다.

### 예제 입력 1 [복사](#)

```
5
5
2
3
4
1
```

### 예제 출력 1 [복사](#)

```
1
2
3
4
5
```

## 출처

- 문제의 오타를 찾은 사람: lazy\_ren

```
1 n = int(input())
2 arr = [int(input()) for _ in range(n)]
3
4 arr.sort() # 오름차순 정렬
5
6 for num in arr:
7     print(num)
```

파이썬에서 `list.sort()` 나 `sorted()` 를 호출했을 때 쓰이는 알고리즘은 **Timsort**(팀소트) 입니다.

### 🌀 Timsort 정렬 알고리즘

- 혼합형 정렬(hybrid sorting)
  - 삽입 정렬(Insertion Sort) + 병합 정렬(Merge Sort)을 결합한 방식
- 실제 데이터에서 자주 나타나는 **부분적으로 정렬된 패턴**을 빠르게 처리할 수 있도록 설계됨.

### 🕒 시간 복잡도

- 최선:  $O(n)$  (이미 정렬에 가까울수록 빠름, 삽입 정렬 성질 활용)
- 평균:  $O(n \log n)$
- 최악:  $O(n \log n)$



```
1 import sys
2 input = sys.stdin.readline
3
4 n = int(input())
5 arr = [int(input()) for _ in range(n)]
6
7 for i in range(n - 1):
8     for j in range(0, n - 1 - i):
9         if arr[j] > arr[j + 1]:
10             arr[j], arr[j + 1] = arr[j + 1], arr[j]
11
12 print('\n'.join(map(str, arr)))
```

## Java (BufferedReader 사용)

java

📄 코드 복사

```
import java.io.*;
import java.util.*;

public class Main {
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int n = Integer.parseInt(br.readLine().trim());
        int[] arr = new int[n];
        for (int i = 0; i < n; i++) {
            arr[i] = Integer.parseInt(br.readLine().trim());
        }

        // Bubble Sort
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - 1 - i; j++) {
                if (arr[j] > arr[j + 1]) {
                    int tmp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = tmp;
                }
            }
        }

        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < n; i++) {
            sb.append(arr[i]).append('\n');
        }
        System.out.print(sb.toString());
    }
}
```



## JavaScript (Node.js, 백준 스타일 입출력)

javascript

📄 코드 복사

```
const fs = require('fs');

const input = fs.readFileSync(0, 'utf8').trim().split('\n');
const n = parseInt(input[0].trim(), 10);
const arr = [];

for (let i = 1; i <= n; i++) {
    arr.push(parseInt(input[i].trim(), 10));
}

// Bubble Sort
for (let i = 0; i < n - 1; i++) {
    for (let j = 0; j < n - 1 - i; j++) {
        if (arr[j] > arr[j + 1]) {
            const tmp = arr[j];
            arr[j] = arr[j + 1];
            arr[j + 1] = tmp;
        }
    }
}

console.log(arr.join('\n'));
```

필요하면 시간복잡도  $O(n^2)$ 인 버블정렬 대신, 같은 입출력으로 **내장 정렬** 사용한 버전도 만들어 드릴게요.

## 02 숫자의 합 구하기(11720)

### 문제

N개의 숫자가 공백 없이 쓰여있다. 이 숫자를 모두 합해서 출력하는 프로그램을 작성하시오.

### 입력

첫째 줄에 숫자의 개수  $N$  ( $1 \leq N \leq 100$ )이 주어진다. 둘째 줄에 숫자 N개가 공백없이 주어진다.

### 출력

입력으로 주어진 숫자 N개의 합을 출력한다.

#### 예제 입력 1 복사

```
1
1
```

#### 예제 출력 1 복사

```
1
```

#### 예제 입력 2 복사

```
5
54321
```

#### 예제 출력 2 복사

```
15
```

### 출처


- 문제를 만든 사람: baekjoon
- 데이터를 추가한 사람: jh05013

```
1  import sys
2  input = sys.stdin.readline
3
4  n = int(input())
5  nums = input().strip()
6
7  ans = sum(map(int, nums))
8  print(ans)
```



## Java

java

 코드 복사

```
import java.io.*;

public class Main {
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));


        int n = Integer.parseInt(br.readLine().trim()); // 숫자의 개수
        String nums = br.readLine().trim();           // 숫자 문자열

        int ans = 0;
        for (int i = 0; i < n; i++) {
            ans += nums.charAt(i) - '0'; // 문자 → 숫자 변환
        }

        System.out.println(ans);
    }
}
```

## JavaScript (Node.js)

javascript

 코드 복사

```
const fs = require('fs');
const input = fs.readFileSync(0, 'utf8').trim().split('\n');

const n = parseInt(input[0].trim(), 10);
const nums = input[1].trim();

let ans = 0;
for (let i = 0; i < n; i++) {
    ans += parseInt(nums[i], 10); // 문자 → 숫자 변환
}

console.log(ans);
```

# 03 구간 합 구하기 (11659)

## 문제

수  $N$ 개가 주어졌을 때,  $i$ 번째 수부터  $j$ 번째 수까지 합을 구하는 프로그램을 작성하시오.

## 입력

첫째 줄에 수의 개수  $N$ 과 합을 구해야 하는 횟수  $M$ 이 주어진다. 둘째 줄에는  $N$ 개의 수가 주어진다. 수는  $1,000$ 보다 작거나 같은 자연수이다. 셋째 줄부터  $M$ 개의 줄에는 합을 구해야 하는 구간  $i$ 와  $j$ 가 주어진다.

## 출력

총  $M$ 개의 줄에 입력으로 주어진  $i$ 번째 수부터  $j$ 번째 수까지 합을 출력한다.

## 제한

- $1 \leq N \leq 100,000$
- $1 \leq M \leq 100,000$
- $1 \leq i \leq j \leq N$

## 예제 입력 1 [복사](#)

```
5 3
5 4 3 2 1
1 3
2 4
5 5
```

## 예제 출력 1 [복사](#)

```
12
9
1
```

## 출처

- 문제를 만든 사람: baekjoon
- 데이터를 추가한 사람: djm03178

```
1  import sys
2  input = sys.stdin.readline
3
4  n, m = map(int, input().split())
5  arr = list(map(int, input().split()))
6
7  # 누적 합(prefix sum) 배열 만들기
8  prefix = [0] * (n + 1)
9  for i in range(1, n + 1):
10     |    prefix[i] = prefix[i - 1] + arr[i - 1]
11
12  # m개의 구간 합 출력
13  for _ in range(m):
14     |    i, j = map(int, input().split())
15     |    print(prefix[j] - prefix[i - 1])
```

## 간결한 Java 버전

java

코드 복사

```
import java.io.*;
import java.util.*;

public class Main {
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st;

        st = new StringTokenizer(br.readLine());
        int n = Integer.parseInt(st.nextToken());
        int m = Integer.parseInt(st.nextToken());

        int[] prefix = new int[n + 1];
        st = new StringTokenizer(br.readLine());
        for (int i = 1; i <= n; i++) {
            prefix[i] = prefix[i - 1] + Integer.parseInt(st.nextToken());
        }

        StringBuilder sb = new StringBuilder();
        while (m-- > 0) {
            st = new StringTokenizer(br.readLine());
            int i = Integer.parseInt(st.nextToken());
            int j = Integer.parseInt(st.nextToken());
            sb.append(prefix[j] - prefix[i - 1]).append('\n');
        }
        System.out.print(sb);
    }
}
```

## JavaScript (Node.js)

javascript

코드 복사

```
const fs = require('fs');
const input = fs.readFileSync(0, 'utf8').trim().split('\n');

const [n, m] = input[0].split(' ').map(Number);
const arr = input[1].split(' ').map(Number);

// 누적 합(prefix sum) 배열
const prefix = Array(n + 1).fill(0);
for (let i = 1; i <= n; i++) {
    prefix[i] = prefix[i - 1] + arr[i - 1];
}

let output = [];
for (let k = 2; k < 2 + m; k++) {
    const [i, j] = input[k].split(' ').map(Number);
    output.push(prefix[j] - prefix[i - 1]);
}

console.log(output.join('\n'));
```

# 04 연속된 자연수의 합 구하기 (2018)

## 문제

어떠한 자연수  $N$ 은, 몇 개의 연속된 자연수의 합으로 나타낼 수 있다. 당신은 어떤 자연수  $N(1 \leq N \leq 10,000,000)$ 에 대해서, 이  $N$ 을 몇 개의 연속된 자연수의 합으로 나타내는 가지수를 알고 싶어한다. 이때, 사용하는 자연수는  $N$ 이하여야 한다.

예를 들어, 15를 나타내는 방법은 15, 7+8, 4+5+6, 1+2+3+4+5의 4가지가 있다. 반면에 10을 나타내는 방법은 10, 1+2+3+4의 2가지가 있다.

$N$ 을 입력받아 가지수를 출력하는 프로그램을 작성하시오.

## 입력

첫 줄에 정수  $N$ 이 주어진다.

## 출력

입력된 자연수  $N$ 을 몇 개의 연속된 자연수의 합으로 나타내는 가지수를 출력하시오

### 예제 입력 1 복사

15

### 예제 출력 1 복사

4

## 출처

[Olympiad](#) > [USA Computing Olympiad](#) > [2002-2003 Season](#) > [USACO March 2003 Contest](#) > [Orange 2번](#)

- 데이터를 추가한 사람: momiens
- 문제의 오타를 찾은 사람: yukariko

```
1  N = int(input())
2
3  count = 0
4  start, end = 1, 1
5  current_sum = 1
6
7  while start <= N:
8      if current_sum == N:
9          count += 1
10         current_sum -= start
11         start += 1
12     elif current_sum < N:
13         end += 1
14         current_sum += end
15     else:
16         current_sum -= start
17         start += 1
18
19  print(count)
```

## Java

java

 코드 복사

```
import java.io.*;

public class Main {
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int N = Integer.parseInt(br.readLine().trim());

        int count = 0;
        int start = 1, end = 1, currentSum = 1;

        while (start <= N) {
            if (currentSum == N) {
                count++;
                currentSum -= start;
                start++;
            } else if (currentSum < N) {
                end++;
                currentSum += end;
            } else {
                currentSum -= start;
                start++;
            }
        }

        System.out.println(count);
    }
}
```

## JavaScript (Node.js)

javascript

 코드 복사

```
const fs = require('fs');
const N = parseInt(fs.readFileSync(0, 'utf8').trim(), 10);

let count = 0;
let start = 1, end = 1, currentSum = 1;

while (start <= N) {
    if (currentSum === N) {
        count++;
        currentSum -= start;
        start++;
    } else if (currentSum < N) {
        end++;
        currentSum += end;
    } else {
        currentSum -= start;
        start++;
    }
}

console.log(count);
```

# 05 ‘좋은 수’ 구하기(1253)

## 문제

N개의 수 중에서 어떤 수가 다른 수 두 개의 합으로 나타낼 수 있다면 그 수를 “좋다(GOOD)”고 한다.

N개의 수가 주어지면 그 중에서 좋은 수의 개수는 몇 개인지 출력하라.

수의 위치가 다르면 값이 같아도 다른 수이다.

## 입력

첫째 줄에는 수의 개수  $N(1 \leq N \leq 2,000)$ , 두 번째 줄에는 i번째 수를 나타내는  $A_i$ 가 N개 주어진다. ( $|A_i| \leq 1,000,000,000$ ,  $A_i$ 는 정수)

## 출력

좋은 수의 개수를 첫 번째 줄에 출력한다.

### 예제 입력 1 [복사](#)

```
10
1 2 3 4 5 6 7 8 9 10
```

### 예제 출력 1 [복사](#)

```
8
```

## 힌트

3,4,5,6,7,8,9,10은 좋다.

## 출처

- 데이터를 추가한 사람: BaaaaaaaaaarkingDog, baggomsoon96, jame0313, kevin0928, kiwiyou
- 문제의 오타를 찾은 사람: jh05013
- 빠진 조건을 찾은 사람: tongnamuu
- 문제를 만든 사람: xhark



```
1  import sys
2  input = sys.stdin.readline
3
4  n = int(input().strip())
5
6  arr = []
7  while len(arr) < n:
8      line = input().strip()
9      if not line:
10         continue
11     arr.extend(map(int, line.split()))
12 arr = arr[:n]
13
14 arr.sort()
15 ans = 0
16
17 for i in range(n):
18     target = arr[i]
19     left, right = 0, n - 1
```

```
20     while left < right:
21         if left == i:
22             left += 1
23             continue
24         if right == i:
25             right -= 1
26             continue
27
28         s = arr[left] + arr[right]
29         if s == target:
30             ans += 1
31             break
32         elif s < target:
33             left += 1
34         else:
35             right -= 1
36
37 print(ans)
```

## Java

java

 코드 복사

```
import java.io.*;
import java.util.*;

public class Main {
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int n = Integer.parseInt(br.readLine().trim());

        List<Integer> list = new ArrayList<>();
        while (list.size() < n) {
            String line = br.readLine();
            if (line == null || line.trim().isEmpty()) continue;
            StringTokenizer st = new StringTokenizer(line);
            while (st.hasMoreTokens() && list.size() < n) {
                list.add(Integer.parseInt(st.nextToken()));
            }
        }

        int[] arr = list.stream().mapToInt(Integer::intValue).toArray();
        Arrays.sort(arr);

        int ans = 0;
        for (int i = 0; i < n; i++) {
            int target = arr[i];
            int left = 0, right = n - 1;
            while (left < right) {
                if (left == i) {
                    left++;
                    continue;
                }
                if (right == i) {
                    right--;
                    continue;
                }

                int s = arr[left] + arr[right];
                if (s == target) {
                    ans++;
                    break;
                } else if (s < target) {
                    left++;
                } else {
                    right--;
                }
            }
        }

        System.out.println(ans);
    }
}
```




```
    }
    if (right == i) {
        right--;
        continue;
    }

    int s = arr[left] + arr[right];
    if (s == target) {
        ans++;
        break;
    } else if (s < target) {
        left++;
    } else {
        right--;
    }
}

System.out.println(ans);
}
```

## 간단한 JavaScript 버전

javascript

 코드 복사

```
const fs = require('fs');
const data = fs.readFileSync(0, 'utf8').trim().split(/\s+/).map(Number);

const n = data[0];
const arr = data.slice(1).sort((a, b) => a - b);

let ans = 0;

for (let i = 0; i < n; i++) {
  let target = arr[i];
  let l = 0, r = n - 1;
  while (l < r) {
    if (l === i) { l++; continue; }
    if (r === i) { r--; continue; }

    let sum = arr[l] + arr[r];
    if (sum === target) { ans++; break; }
    sum < target ? l++ : r--;
  }
}

console.log(ans);
```

# 06 최솟값 구하기 (11003)

## 문제

N개의 수  $A_1, A_2, \dots, A_N$ 과 L이 주어진다.

$D_i = A_{i-L+1} \sim A_i$  중의 최솟값이라고 할 때, D에 저장된 수를 출력하는 프로그램을 작성하시오. 이때,  $i \leq 0$  인  $A_i$ 는 무시하고 D를 구해야 한다.

## 입력

첫째 줄에 N과 L이 주어진다. ( $1 \leq L \leq N \leq 5,000,000$ )

둘째 줄에는 N개의 수  $A_i$ 가 주어진다. ( $-10^9 \leq A_i \leq 10^9$ )

## 출력

첫째 줄에  $D_i$ 를 공백으로 구분하여 순서대로 출력한다.

### 예제 입력 1 [복사](#)

```
12 3
1 5 2 3 6 2 3 7 3 5 2 6
```

### 예제 출력 1 [복사](#)

```
1 1 1 2 2 2 2 2 3 3 2 2
```

## 출처

- 문제를 만든 사람: baekjoon
- 데이터를 추가한 사람: doju

```
1  import sys
2  from collections import deque
3
4  input = sys.stdin.readline
5
6  n, l = map(int, input().split())
7  arr = list(map(int, input().split()))
8
9  dq = deque()
10 result = []
11
12 for i in range(n):
13     while dq and dq[-1][0] > arr[i]:
14         dq.pop()
15     dq.append((arr[i], i))
16
17     if dq[0][1] <= i - l:
18         dq.popleft()
19
20     result.append(dq[0][0])
21
22 print(*result)
```

java

```
import java.io.*;
import java.util.*;

public class Main {
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st = new StringTokenizer(br.readLine());
        int n = Integer.parseInt(st.nextToken());
        int l = Integer.parseInt(st.nextToken());

        st = new StringTokenizer(br.readLine());
        int[] arr = new int[n];
        for (int i = 0; i < n; i++) arr[i] = Integer.parseInt(st.nextToken());

        Deque<int[]> dq = new ArrayDeque<>();
        StringBuilder sb = new StringBuilder();

        for (int i = 0; i < n; i++) {
            // 현재 값보다 큰 값들은 제거
            while (!dq.isEmpty() && dq.peekLast()[0] > arr[i]) dq.pollLast();
            dq.offerLast(new int[]{arr[i], i});

            // 범위 벗어난 값 제거
            if (dq.peekFirst()[1] <= i - l) dq.pollFirst();

            // 현재 윈도우의 최소값
            sb.append(dq.peekFirst()[0]).append(' ');
        }

        System.out.println(sb.toString().trim());
    }
}
```

📄 코드 복사



## 🌐 JavaScript (Node.js)

javascript

```
const fs = require('fs');
const input = fs.readFileSync(0, 'utf8').trim().split(/\s+/).map(Number);

const n = input[0], l = input[1];
const arr = input.slice(2);

const dq = []; // [value, index]
const result = [];

for (let i = 0; i < n; i++) {
    // 현재 값보다 큰 값들은 dq에서 제거
    while (dq.length && dq[dq.length - 1][0] > arr[i]) dq.pop();
    dq.push([arr[i], i]);

    // 윈도우 범위에서 벗어난 값 제거
    if (dq[0][1] <= i - l) dq.shift();

    // 현재 윈도우 최소값은 맨 앞
    result.push(dq[0][0]);
}

console.log(result.join(' '));
```

📄 코드 복사

# 07 스택으로 오름차순 수열 만들기 (1874)

## 문제

스택 (**stack**)은 기본적인 자료구조 중 하나로, 컴퓨터 프로그램을 작성할 때 자주 이용되는 개념이다. 스택은 자료를 넣는 (**push**) 입구와 자료를 뽑는 (**pop**) 입구가 같아 제일 나중에 들어간 자료가 제일 먼저 나오는 (**LIFO, Last in First out**) 특성을 가지고 있다.

1부터  $n$ 까지의 수를 스택에 넣었다가 뽑아 늘어놓음으로써, 하나의 수열을 만들 수 있다. 이때, 스택에 **push**하는 순서는 반드시 오름차순을 지키도록 한다고 하자. 임의의 수열이 주어졌을 때 스택을 이용해 그 수열을 만들 수 있는지 없는지, 있다면 어떤 순서로 **push**와 **pop** 연산을 수행해야 하는지를 알아낼 수 있다. 이를 계산하는 프로그램을 작성하라.

## 입력

첫 줄에  $n$  ( $1 \leq n \leq 100,000$ )이 주어진다. 둘째 줄부터  $n$ 개의 줄에는 수열을 이루는 1이상  $n$ 이하의 정수가 하나씩 순서대로 주어진다. 물론 같은 정수가 두 번 나오는 일은 없다.

## 출력

입력된 수열을 만들기 위해 필요한 연산을 한 줄에 한 개씩 출력한다. **push**연산은 **+**로, **pop** 연산은 **-**로 표현하도록 한다. 불가능한 경우 **NO**를 출력한다.

예제 입력 1 복사

```
8
4
3
6
8
7
5
2
1
```

예제 출력 1 복사

```
+
```

```
+
```

```
+
```

```
+
```

```
-
```

```
-
```

```
+
```

```
+
```

```
-
```

```
+
```

```
+
```

```
-
```

```
-
```

```
-
```

```
-
```

```
-
```

```
-
```

```
-
```

```
1  import sys
2  input = sys.stdin.readline
3
4  n = int(input())
5  target = []
6  for _ in range(n):
7      target.append(int(input()))
8
9  stack = []
10 ops = []
11 next_num = 1
12
13 for t in target:
14     while next_num <= t:
15         stack.append(next_num)
16         ops.append('+')
17         next_num += 1
18
19
20
21     if stack and stack[-1] == t:
22         stack.pop()
23         ops.append('-')
24     else:
25         print("NO")
26         sys.exit(0)
27
28
29 print('\n'.join(ops))
```



## 간소화된 Java 버전

java

 코드 복사

```
import java.io.*;
import java.util.*;

public class Main {
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int n = Integer.parseInt(br.readLine().trim());

        Deque<Integer> stack = new ArrayDeque<>();
        StringBuilder sb = new StringBuilder();
        int next = 1;

        for (int i = 0; i < n; i++) {
            int t = Integer.parseInt(br.readLine().trim());

            while (next <= t) {
                stack.push(next++);
                sb.append("+\n");
            }

            if (!stack.isEmpty() && stack.peek() == t) {
                stack.pop();
                sb.append("-\n");
            } else {
                System.out.println("NO");
                return;
            }
        }

        System.out.print(sb);
    }
}
```

## JavaScript (Node.js)

javascript

 코드 복사

```
const fs = require('fs');
const input = fs.readFileSync(0, 'utf8').trim().split('\n').map(Number);

const n = input[0];
const target = input.slice(1);

const stack = [];
const ops = [];
let nextNum = 1;
let possible = true;

for (const t of target) {
    while (nextNum <= t) {
        stack.push(nextNum++);
        ops.push('+');
    }

    if (stack.length && stack[stack.length - 1] === t) {
        stack.pop();
        ops.push('-');
    } else {
        possible = false;
        break;
    }
}

if (!possible) console.log("NO");
else console.log(ops.join('\n'));
```



# 08 카드게임 (2164)

## 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, 남는 카드는 4가 된다.

N이 주어졌을 때, 제일 마지막에 남게 되는 카드를 구하는 프로그램을 작성하시오.

## 입력

첫째 줄에 정수 N( $1 \leq N \leq 500,000$ )이 주어진다.

## 출력

첫째 줄에 남게 되는 카드의 번호를 출력한다.

### 예제 입력 1 [복사](#)

6

### 예제 출력 1 [복사](#)

4

```
1  # num 2164.py
2  def remove_odd_index(arr):
3      j = 0
4      for i in range(0, len(arr), 2):
5          arr[j] = arr[i]
6          j += 1
7      del arr[j:]
8      return arr
9
10 def remove_even_index(arr):
11     j = 0
12     for i in range(1, len(arr), 2):
13         arr[j] = arr[i]
14         j += 1
15     del arr[j:]
16     return arr
17
18 n = int(input())
19 arr = list(range(1, n + 1))
20
21 while len(arr) > 1:
22     if len(arr) % 2 == 0:
23         arr = remove_even_index(arr)
24     else:
25         keep = arr[1]
26         tail = arr[2:]
27         arr = remove_even_index(tail)
28         arr.append(keep)
29
30 print(arr[0])
```

## Java (큐 이용한 효율적 구현)

java

📄 코드 복사

```
import java.io.*;
import java.util.*;

public class Main {
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int n = Integer.parseInt(br.readLine().trim());

        Deque<Integer> q = new ArrayDeque<>();
        for (int i = 1; i <= n; i++) q.addLast(i);

        while (q.size() > 1) {
            q.pollFirst();           // 맨 앞 버리기
            q.addLast(q.pollFirst()); // 다음 카드 뒤로 보내기
        }

        System.out.println(q.peek());
    }
}
```

## JavaScript (Node.js, 큐 활용)

javascript

📄 코드 복사

```
const fs = require('fs');
const n = parseInt(fs.readFileSync(0, 'utf8').trim(), 10);

const q = [];
for (let i = 1; i <= n; i++) q.push(i);

let front = 0;
while (q.length - front > 1) {
    front++;           // 맨 앞 제거
    q.push(q[front]);  // 다음 카드 뒤로 이동
    front++;
}

console.log(q[front]);
```