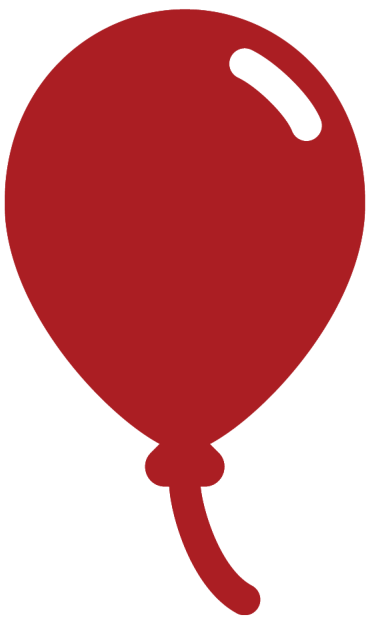
제 1회 청정수컵 에디토리얼



Sogang ACM-ICPC Team

2019년 4월 13일

15시 00분 – 17시 00분

개요

이번 대회는 2010년대 초반의 ICPC 팀 선발전을 제외하고는 학회에서 학회원들을 대상으로 개최한 대회로서는 처음이었습니다. 초급 스터디를 수강하시는 학우 분들의 프로그래밍 대회 맛보기와 기초 공학설계 중간고사 대비 목적으로 개최되었으며, 모든 문제는 초급 스터디에서 배운 내용만으로 풀 수 있는 문제로 구성하였습니다.
첫 번째 제출은 대회가 시작하자마자 나왔고, 문제 **A**, **B**, **F**는 참가자 전원께서 풀어 주셨습니다. 가장 어려웠던 문제는 각각 1명, 0명께서 푸신 **G**와 **H**였고, 가장 많이 푸신 분께서는 8문제 중 7문제를 풀어 주셨습니다.
시간 내어 참가해 주신 모든 분들께 감사드립니다!

대회 운영진

박수현 - 대회 기획, 문제 선정, 해설 집필
임지환 - 대회 기획, 해설 감수
이용욱 - 대회 기획
박한나 - 대회 기획
이태한 - 대회 기획

Problem A

N 찍기

시간 제한: 1 초

메모리 제한: 128 MB

자연수 N이 주어졌을 때, 1부터 N까지 한 줄에 하나씩 출력하는 프로그램을 작성하시오.

입력

첫째 줄에 100,000보다 작거나 같은 자연수 N이 주어진다.

출력

첫째 줄부터 N번째 줄 까지 차례대로 출력한다.

예제 입력	예제 출력
5	1 2 3 4 5

해설

단순한 출력 문제이다. n을 입력받아 `for` 루프 안에서 하나씩 출력하면 된다.

정답 소스

```c
#include <stdio.h>

int main() {
	int n;
	scanf("%d", &n);
	for (int i = 1; i <= n; i++) {
		printf("%d\n", i);
	}
	return 0;
```

```
}
```

문제 출처

백준 온라인 저지 #2741, 만든 사람 baekjoon

문제 통계

정답: 7명 중 7명 (100%)

첫 정답자: fakingate (김진수) / 대회 시작 후 0분

Problem B

A+B - 3

시간 제한: 1 초

메모리 제한: 256 MB

두 정수 A와 B를 입력받은 다음, $A+B$를 출력하는 프로그램을 작성하시오.

입력

첫째 줄에 테스트 케이스의 개수 T가 주어진다.

각 테스트 케이스는 한 줄로 이루어져 있으며, 각 줄에 A와 B가 주어진다. $(0 < A, B < 10)$

출력

각 테스트 케이스마다 $A+B$를 출력한다.

예제 입력

```
5
1 1
2 3
3 4
9 8
5 2
```

예제 출력

```
2
5
7
17
7
```

해설

단순한 덧셈 문제이다. 테스트케이스에 제한이 없었는데, 입력 스트림과 출력 스트림은 별개이므로, 별도로 들어오는 입력을 전부 저장해 두고 나중에 전부 계산해 출력할 필요는 없다. 보통 프로그래밍 대회에서는 입력이 들어온 직후에 출력해도 괜찮다.

정답 소스

```c
#include <stdio.h>

int main() {
        int n;
        scanf("%d", &n);
        for (int i = 1; i <= n; i++) {
```

```
        int x, y;
        scanf("%d%d", &x, &y);
        printf("%d\n", x + y);
    }
    return 0;
}
```

문제 출처

백준 온라인 저지 #10950, 만든 사람 baekjoon, 빠진 조건을 찾은 사람 djm03178

문제 통계

정답: 7명 중 7명 (100%)

첫 정답자: alluo2 (김세영), whysoserious (임수민) / 대회 시작 후 5분

Problem C
주차의 신
시간 제한: 1 초
메모리 제한: 128 MB

선영이는 쇼핑하러 긴 도로에 자주 간다. 선영이는 주차를 세상에서 가장 귀찮아 하기 때문에, 아무데나 주차를 한다. 주차를 한 후에는 가려고 했던 상점으로 걸어 간다.

어느날, 선영이는 다리가 너무 아파서 병원에 갔다. 의사는 선영이에게 되도록 조금 걷거나, 쇼핑을 하지 말라고 했다. 선영이는 쇼핑을 버릴 수 없다. 그녀의 특기를 발휘해서 가장 좋은 주차 자리를 찾으려고 한다.

긴 도로는 일직선이다. 또, 모든 상점의 위치는 정수 좌표를 가지고 있다. 주차장은 모든 정수 좌표마다 하나씩 있으며, 선영이를 위해 항상 비어있다. 선영이는 주차비를 아끼기 위해서 쇼핑을 마치기 전까지는 주차한 차를 이동시키지 않을 것이다. 선영이는 힘이 매우 세기 때문에, 자신이 쇼핑한 물건을 모두 들지 못하는 경우는 없다. 가려고 계획한 상점은 모두 방문해야 한다.

입력

첫째 줄에 테스트 케이스의 개수 t가 주어진다. ($1 \le t \le 100$) 모든 테스트 케이스는 두 줄로 이루어져 있다. 첫째 줄에는 선영이가 방문할 상점의 수 n이 주어지며 ($1 \le n \le 20$), 둘째 줄에는 상점의 위치가 주어진다. ($0 \le x_i \le 99$)

출력

선영이가 가려고 했던 모든 상점을 방문하고 차로 돌아오기 위해 걸어야 하는 거리의 최솟값을 출력한다.

예제 입력	예제 출력
2 4 24 13 89 37 6 7 30 41 14 39 42	152 70

해설

주차 위치와 상관없이, 맨 왼쪽의 상점과 맨 오른쪽의 상점은 꼭 방문해야 한다. 맨 왼쪽 상점과 맨 오른쪽 상점 사이 어느 위치에 주차하더라도 (주차 위치에서 맨 왼쪽 상점까지 가는 거리) + (맨 왼쪽 상점에서 맨 오른쪽 상점으로 가는 거리) + (맨 오른쪽 상점에서 다시 주차 위치로 돌아오는 거리)는 일

정함을 알 수 있다. 따라서 걸어야 하는 거리는 항상 $2 \times$ (맨 왼쪽 상점부터 맨 오른쪽 상점까지의 거리)이다.

수 n개의 최댓값과 최솟값은 아래 정답 소스와 같이 구할 수 있다. 충분히 큰 수 min과 충분히 작은 수 max를 변수로 선언해 두고, 수가 들어올 때마다 min보다 들어온 수가 더 작으면 min을 들어온 수로 바꿔 주고, max보다 들어온 수가 더 크면 max를 들어온 수로 바꿔 준다. 이 문제의 경우 들어오는 좌표는 $0 \le x_i \le 99$이었으므로, 초기에 $min = 0$, $max = 99$로 잡으면 적당하겠다.

정답 소스

```c
#include <stdio.h>

int main() {
	int t, n, x;
	scanf("%d", &t);
	for (int i = 1; i <= t; i++) {
		scanf("%d", &n);
		int min = 99, max = 0;
		for (int j = 1; j <= n; j++) {
			scanf("%d", &x);
			if (max < x) max = x;
			if (min > x) min = x;
		}
		printf("%d\n", 2 * (max - min));
	}
	return 0;
}
```

문제 출처

ACM-ICPC Nordic Collegiate Programming Contest (NCPC), 2007, C번

문제 통계

정답: 5명 중 7명 (71.4%)

첫 정답자: alluo2 (김세영) / 대회 시작 후 21분

Problem D
3의 배수
시간 제한: 0.1 초
메모리 제한: 128 MB

윤영이는 3의 배수 마니아이다. 그는 모든 자연수를 3개의 3의 배수의 자연수로 분해하는 것을 취미로 가지고 있다. 문득 그는 자신에게 주어진 수를 3개의 3의 배수로 분리하는 경우의 수가 몇 개인지 궁금해졌다. 하지만 윤영이는 마지막 학기이기 때문에 이런 계산을 하기에는 너무 게을러졌다. 그래서 당신에게 이 계산을 부탁했다.

즉, 임의의 3의 배수 자연수 n이 주어졌을 때, 해당 수를 3의 배수의 자연수 3개로 분리하는 방법의 개수를 출력해라. 단 분해한 수의 순서가 다르면 다른 방법으로 간주한다. 예를 들어 $12 = 3 + 6 + 3$ 과 $12 = 3 + 3 + 6$ 은 다른 방법이다.

입력
임의의 3의 배수 자연수 n이 주어진다. ($3 \leq n \leq 3000$)

출력
자연수 n을 분해하는 방법의 개수를 출력하라.

예제 입력 1
```
9
```

예제 출력 1
```
1
```

예제 입력 2
```
12
```

예제 출력 2
```
3
```

해설 1

3의 배수를 여러 3의 배수로 분해해야 하므로, $\frac{n}{3}$을 자연수로 분해하는 방법의 수로 생각할 수도 있겠다.

분해하는 순서는 중요하지 않으므로, 2중 `for` 루프를 사용해 해결할 수 있다. 첫 번째 수 i와 두 번째 수 j를 루프에서 고르면 세 번째 수는 $\frac{n}{3} - i - j$이고, 이 수가 1보다 크거나 같다면 $\frac{n}{3}$을 세 개의 자연수로 분할할 수 있는 하나의 방법이 되는 것이다.

정답 1 소스

```c
#include <stdio.h>

int main() {
	int n;
	scanf("%d", &n);
	int x = n / 3, s = 0;
	for (int i = 1; i <= x; i++) {
		for (int j = 1; j <= x; j++) {
			int k = x - i - j;
			if (k >= 1) s++;
		}
	}
	printf("%d", s);
	return 0;
}
```

해설 2

자연수 x를 순서를 신경쓰지 않고 자연수 3개로 분할하는 방법의 수는 길이 x짜리 선분을 길이 1 이상의 선분 3개로 나누는 것으로도 생각할 수 있는데, 이 때 선분을 나누는 방법의 개수는 선분을 자를 수 있는 $x-2$개의 지점 중 두 지점을 선택하는 방법의 수 $\binom{x-2}{2} = \frac{(x-1)(x-2)}{2}$와 같다.

정답 2 소스

```c
#include <stdio.h>

int main() {
	int n;
	scanf("%d", &n);
	int x = n / 3;
	printf("%d", (x - 1) * (x - 2) / 2);
```

```
        return 0;
}
```

문제 출처

2018 서강 프로그래밍 대회 (Master), A번

문제 통계

정답: 5명 중 7명 (71.4%)

첫 정답자: kevink1113 (강상원) / 대회 시작 후 26분

Problem E
Euler's Number
시간 제한: 2 초
메모리 제한: 512 MB

오일러 상수(e로 더 알려진 상수)는 수학에서 특별한 역할을 담당하고 있다. 아마 미적분학이나 경제학(특히 복리 계산에서) 분야에서, 또는 계산기에 있는 자연로그 ln의 밑 등으로 e를 접해본 적 있을 것이다.

극한으로 e를 계산할 수는 있지만, 이산적으로 e를 근사할 수도 있다. $0! = 1$이라고 하면 e는

$$e = \sum_{i=0}^{n} \frac{1}{i!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots$$

이다. n이 ∞에 다가가면 오른쪽의 급수는 e로 다가가게 된다. 여기서 n이 양의 정수라면 위의 식은 e의 실제 값에 대한 근사식으로 사용할 수 있다. (예를 들어, $n = 10$일 때 이 식은 소숫점 아래 7자리까지 정확한 값을 계산해낼 수 있다.)

n의 값을 유일한 입력으로 받아서 n의 값에 대한 e의 근삿값을 계산하는 프로그램을 작성하여라.

입력

정수 n이 주어진다. ($0 \leq n \leq 10000$)

출력

n의 값에 대한 e의 근삿값을 출력하라. 절대/상대 오차는 10^{-12}까지 허용한다.

예제 입력 1

```
3
```

예제 출력 1

```
2.6666666666666665
```

예제 입력 2

```
15
```

예제 출력 2

```
2.718281828458995
```

해설

단순히 $\frac{1}{x!}$를 계산해서 더하면 되는 문제이지만, 10^{-12} 이하의 오차가 요구됨에 주의해야 한다. `float`은 꽤 부정확하므로 이 문제를 해결하는 데 부적합하고, `double` 혹은 `long double`을 사용하여 계산해야 한다.

$\frac{1}{x!}$를 계산하는 함수를 만드는 것도 괜찮은 방법이나, $x!$를 계산하는 함수를 만들어서 결과값에 역수를 취해 더해 풀었다면 틀릴 수도 있다. 13!부터는 int 범위 밖이고, 21!부터는 long long 범위 밖이기 때문이다.

정답 소스

```
#include <stdio.h>

int main() {
	int n;
	scanf("%d", &n);
	double sum = 1;
	double curr = 1;
	for (int i = 1; i <= n; i++) {
		curr /= i;
		sum += curr;
	}
	printf("%.20lf", sum);
	return 0;
}
```

문제 출처

ACM-ICPC North Central North America Regional (NCNA), 2018, E번

문제 통계

정답: 4명 중 7명 (57.1%)

첫 정답자: v010309 (강승구) / 대회 시작 후 55분

Problem F
돌 게임 2
시간 제한: 1 초
메모리 제한: 128 MB

돌 게임은 두 명이서 즐기는 재밌는 게임이다.

탁자 위에 돌 N개가 있다. 상근이와 창영이는 턴을 번갈아가면서 돌을 가져가며, 돌은 1개 또는 3개 가져갈 수 있다. 마지막 돌을 가져가는 사람이 게임을 지게 된다.

두 사람이 완벽하게 게임을 했을 때, 이기는 사람을 구하는 프로그램을 작성하시오. 게임은 상근이가 먼저 시작한다.

입력

첫째 줄에 N이 주어진다. ($0 \le n \le 1000$)

출력

상근이가 게임을 이기면 SK를, 창영이가 게임을 이기면 CY을 출력한다.

예제 입력	예제 출력
4	SK

해설

돌이 1개나 3개 있다면 상근이가 처음에 다 가져가 버리면 이긴다는 것을 알 수 있다. 그리고 돌이 2개나 4개 있다면 상근이가 어떻게 게임을 해도 이길 수 없다는 것을 알 수 있다.

돌이 홀수 개 주어진다면, 창영이가 1개를 가져갔다면 상근이는 3개를, 창영이가 3개를 가져갔다면 상근이는 1개를 가져가는 방법으로 상근이가 항상 이길 수 있다. 같은 방법으로 돌이 짝수 개 주어진다면 창영이가 항상 이길 수 있다.

따라서 n이 짝수라면 CY를, 홀수라면 SK를 출력하면 된다.

정답 소스

```c
#include <stdio.h>

int main() {
```

```
        int n;
        scanf("%d", &n);
        if (n % 2 == 0) {
                printf("CY");
        } else {
                printf("SK");
        }
        return 0;
}
```

문제 출처

백준 온라인 저지 #9656, 만든 사람 baekjoon

문제 통계

정답: 7명 중 7명 (100%)

첫 정답자: v010309 (강승구) / 대회 시작 후 63분

Problem G
유진수
시간 제한: 2 초
메모리 제한: 128 MB

유진수는 어떤 수를 10진수로 표현한 뒤 그 수를 두 부분으로 나눴을 때, 앞부분 자리수의 곱과 뒷부분 자리수의 곱이 같을 때를 말한다.
예를 들어, 1221은 유진수이다. 12와 21로 나눴을 때, 앞부분 자리수의 곱 1×2는 뒷부분 자리수의 곱 2×1과 같기 때문이다. 1236도 마찬가지로 유진수이다. 하지만, 1234는 아니다. 수를 나눌 때 항상 연속된 자리수를 나눠야 하고, 각 부분에 적어도 한 자리는 있어야 한다.
예를 들어, 12345는 총 4가지 방법으로 나눌 수 있다. 1-2345, 12-345, 123-45, 1234-5
어떤 수 N이 주어질 때, 이 수가 유진수인지 아닌지 구하는 프로그램을 작성하시오.

입력
첫째 줄에 수 N이 주어진다. 이 수는 2,147,483,647보다 작거나 같은 자연수이다.

출력
첫째 줄에 N이 유진수이면 `YES`, 아니면 `NO`를 출력한다.

예제 입력	예제 출력
1236	YES

해설

주어진 수를 10의 거듭제곱인 수로 나누면 몫과 나머지는 각각 앞부분과 뒷부분이 된다. 예를 들면,

$$1234 \Rightarrow 123 \times 10^1 + 4$$
$$1234 \Rightarrow 12 \times 10^2 + 34$$
$$1234 \Rightarrow 1 \times 10^3 + 234$$

이다. `for` 루프를 이용해 $n = l \times 10^i + r$의 꼴로 n을 왼쪽 수 l과 오른쪽 수 r로 분할하자. 이제 10의 거듭제곱수를 계산하는 함수와 정수의 모든 자릿수를 곱하는 함수를 만들면 문제를 해결할 수 있다. 하단의 `product` 함수와 같이, 먼저 $p = 1$을 선언해 둔다. 이제 n을 10씩 나눠가면서 n의 1의 자리(즉, n을 10으로 나눈 나머지)를 p에 곱해나간다. 이렇게 하면 n의 가장 오른쪽 숫자부터 가장 왼쪽

숫자까지를 차례로 p에 곱할 수 있다. n이 0이 될 때까지 이 과정을 반복한다.

단, 이와 같이 알고리즘을 짜면 오른쪽 수 l 앞부분에 0이 있는 경우를 제대로 처리할 수 없다. 예를 들어

$$123006 = 123 \times 10^3 + 006$$

으로 분할했을 경우에는 오른쪽의 수의 각 자리수의 곱이 0이 되어야 하는데, `product(6) = 6`이 되어 버린다.

$n = l \times 10^x + r$로 오른쪽 수를 얻었다면 l은 x자리 수, 즉 $10^{x-1} \leq l < 10^x$이어야 하므로 $l < 10^{x-1}$이라면 l의 맨 앞자리가 0임을 의미하게 된다. 이 경우만 따로 처리해 주면 된다.

정답 소스

```c
#include <stdio.h>

int pow10(int n) {
	int p = 1;
	for (int i = 1; i <= n; i++) p *= 10;
	return p;
}

int product(int n) {
	if (n == 0) return 0;
	int p = 1;
	while (n > 0) {
		p *= (n % 10);
		n /= 10;
	}
	return p;
}

int main() {
	int n;
	scanf("%d", &n);
```

```
        int flag = 0;
        for (int i = 1; i <= 9; i++) {
                int pow = pow10(i);
                if (pow > n) break;

                int left = n / pow;
                int right = n % pow;
                if (right < pow / 10) right = 0;

                if (product(left) == product(right)) {
                        flag = 1;
                        break;
                }
        }

        if (flag) {
                printf("YES");
        } else {
                printf("NO");
        }
        return 0;
}
```

문제 출처

백준 온라인 저지 #1356, 만든 사람 baekjoon

문제 통계

정답: 1명 중 7명 (14.2%)

첫 정답자: v010309 (강승구) / 대회 시작 후 87분

Problem H

서로소

시간 제한: 1 초

메모리 제한: 128 MB

양의 정수 n이 주어졌을 때, n보다 작은 양의 정수 중에서 n과 서로소인 수 개수를 구하는 프로그램을 작성하시오.

두 정수 a와 b가 서로소가 되려면 $x > 1, y > 0, z > 0$이면서, $a = xy, b = xz$를 만족하는 정수가 없어야 한다.

입력

입력은 여러 개의 테스트 케이스로 이루어져 있으며, 각 테스트 케이스는 $n \leq 1{,}000{,}000{,}000$으로 이루어져 있다.

입력의 마지막 줄에는 0이 주어진다.

출력

입력으로 주어진 n마다 n보다 작으면서 서로소인 양의 정수의 수를 출력한다.

예제 입력	예제 출력
7 12 0	6 4

해설

1부터 n까지의 수 중 n과 서로소인 수의 개수를 $\phi(n)$이라고 하자.

일단 n이 소수라면 자명히 $\phi(n) = n - 1$이다. 또한 n이 소수 p의 거듭제곱 p^k라면 $\phi(n) = n^{k-1}(n-1)$이다.

특히 서로소인 두 정수 n과 m에 대해 n과의 공약수가 1이면서 m과의 공약수가 1이면 mn과의 공약수도 1임을 알 수 있고, 따라서 $\phi(m)\phi(n) = \phi(mn)$임을 알 수 있다. 그러므로 n의 소인수 p_i들에 대해

$$n = p_1^{a_1} p_2^{a_2} p_3^{a_3} \times \cdots \times p_k^{a_k}$$

이면

$$\begin{aligned}\phi(n) &= p_1^{a_1-1}(p_1-1)\,p_2^{a_2-1}(p_2-1)\,p_3^{a_3-1}(p_3-1)\times\cdots\times p_k^{a_k-1}(p_k-1)\\ &= \left[p_1^{a_1-1}p_2^{a_2-1}p_3^{a_3-1}\times\cdots\times p_k^{a_k-1}\right]\times[(p_1-1)(p_2-1)(p_3-1)\times\cdots\times(p_k-1)]\\ &= \frac{n}{p_1p_2p_3\times\cdots\times p_k}\times(p_1-1)(p_2-1)(p_3-1)\times\cdots\times(p_k-1)\\ &= n\left(\frac{p_1-1}{p_1}\right)\left(\frac{p_2-1}{p_2}\right)\left(\frac{p_3-1}{p_3}\right)\times\cdots\times\left(\frac{p_k-1}{p_k}\right)\\ &= n\prod_{1\le i\le k}\left(\frac{p_i-1}{p_i}\right)\end{aligned}$$

이다.

$n = p_1^{a_1}p_2^{a_2}p_3^{a_3}\times\cdots\times p_k^{a_k}$이므로 n은 p_i로 나눠질 것이다.

n을 res라는 새 변수로 복사해 두자. 우리는 n을 어떤 소수로 나눴을 때 나머지가 0인 소수들에만 관심이 있다. $2\le p\le\sqrt{n}$의 정수 p에 대해 만약 p가 n을 나눈다면, res에 $\frac{p_i-1}{p_i}$을 곱한다. 그리고 n이 p를 소인수로 갖지 않을 때까지 p로 나눈다. 그러면 n은 p의 배수로는 더 이상 나눠지지 않을 것이므로, p가 소수임이 보장된다.

$2\le p\le\sqrt{n}$의 모든 소수 p로 n을 나눠봤는데 n이 1이 아니라면 남은 n 자체가 $\sqrt{n}$보다 큰 소수이다. 이 경우에는 남은 n이 소수라는 점을 이용해 마지막에 res에 $\frac{n-1}{n}$을 곱하면 된다.

res도 p_i로 항상 나눠짐이 보장되므로,

$$res\times\frac{p_i-1}{p_i} = res\left(1-\frac{1}{p_i}\right) = res-\frac{res}{p_i}$$

라는 점에 착안해 $\frac{res}{p_i}$를 계속 빼 줄 수도 있겠다.

$\phi(x)$는 정수론에서 오일러 토션트 함수(Euler's phi function)라고 불리며, 공식 $\phi(x) = x\prod_{p|x}\left(1-\frac{1}{p}\right)$은 오일러 곱 공식(Euler product formula)이라고 불린다.

정답 소스

```c
#include <stdio.h>

int phi(int n) {
    int res = n;
```

```
    for (int p = 2; p * p <= n; p++) {
        if (x % p == 0) {
            while (n % p == 0) n /= p;
            res -= res / p;
        }
    }

    if (n > 1) res -= res / n;
    return res;
}

int main() {
    while (true) {
        int x;
        scanf("%d", &x);
        if (x == 0) break;
                printf("%d\n", phi(x));
    }

    return 0;
}
```

문제 출처

Waterloo's local Programming Contests, 2002년 6월 1일, E번

문제 통계

정답: 0명 중 7명 (0.0%)