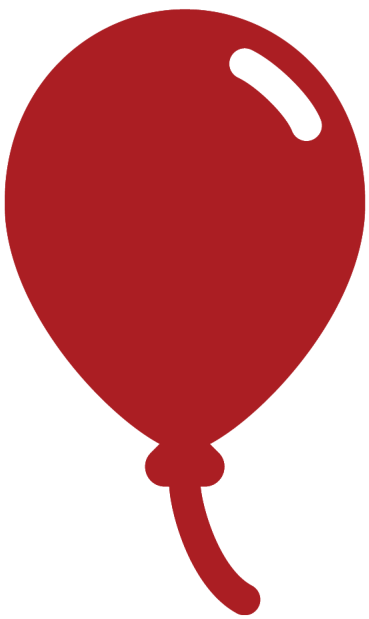
제 2회 청정수컵 에디토리얼



Sogang ACM-ICPC Team

2019년 6월 6일

15시 00분 – 17시 00분

개요

이번 대회는 초급 스터디를 수강하시는 학우 분들의 프로그래밍 대회 맛보기와 기초공학설계 기말고사 대비 목적으로 개최되었으며, 모든 문제는 초급 스터디에서 배운 내용만으로 풀 수 있는 문제로 구성하였습니다.
난이도별로 문제 **A**–**C**는 무난한 문제들, **D**–**F**는 다소 생각이 필요한 문제들, **G**–**I**는 어려운 문제들로 구성하였으며, 알고리즘별로 크게 **A**와 **D**, **F**, **I**는 배열 사용, **B**와 **G**는 재귀 함수 활용, **C**와 **E**, **H**는 문자열 처리로 구성하였습니다. 특히 **H**는 작년(2018년) 기초공학설계 기말고사 **2**번 문제와 거의 같은 알고리즘이 필요한 문제였습니다.
첫 번째 제출은 대회 시작 2분만에 나왔고, 문제 **A**, **B**는 참가자 전원께서 풀어 주셨습니다. 안타깝게도 **H**와 **I**는 정답자가 없었습니다. 가장 많이 푸신 분께서는 9문제 중 7문제를 풀어 주셨습니다.
시간 내어 참가해 주신 모든 분들께 감사드립니다!

대회 운영진

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

Problem A
평균 점수
시간 제한: 1 초
메모리 제한: 128 MB

상현이가 가르치는 아이폰 앱 개발 수업의 수강생은 원섭, 세희, 상근, 숭, 강수이다.
어제 이 수업의 기말고사가 있었고, 상현이는 지금 학생들의 기말고사 시험지를 채점하고 있다. 기말고사 점수가 40점 이상인 학생들은 그 점수 그대로 자신의 성적이 된다. 하지만, 40점 미만인 학생들은 보충학습을 듣는 조건을 수락하면 40점을 받게 된다. 보충학습은 거부할 수 없기 때문에, 40점 미만인 학생들은 항상 40점을 받게 된다.
학생 5명의 점수가 주어졌을 때, 평균 점수를 구하는 프로그램을 작성하시오.

입력
입력은 총 5줄로 이루어져 있고, 원섭이의 점수, 세희의 점수, 상근이의 점수, 숭이의 점수, 강수의 점수가 순서대로 주어진다.
점수는 모두 0점 이상, 100점 이하인 5의 배수이다. 따라서, 평균 점수는 항상 정수이다.

출력
첫째 줄에 학생 5명의 평균 점수를 출력한다.

예제 입력	예제 출력
10 65 100 30 95	168

힌트
숭과 원섭이는 40점 미만이고, 보충학습에 참여할 예정이기 때문에 40점을 받게 된다. 따라서, 점수의 합은 340점이고, 평균은 68점이 된다.

해설 1

구성원들의 처음 점수를 각각 x_1, x_2, x_3, x_4, x_5라고 하자. 보충학습은 거부할 수 없기 때문에 보충학습을 들은 이후의 점수 $y_k = \max(x_k, 40)$이다. 마지막으로 평균은 $\frac{y_1+y_2+y_3+y_4+y_5}{5}$이다.

이 문제는 크기 5의 1차원 배열을 만들어 해결할 수 있다. $a[0]$부터 $a[5]$를 순서대로 입력받고, 각각의 원소에 대해 만약 $a[i] < 40$이라면 $a[i]$를 40으로 만들어 준다. 마지막에는 정수 $s = 0$을 선언한 뒤 $a[i]$를 s에 누적해 주고, $\frac{s}{5}$를 출력하면 정답이다.

정답 소스 1

```c
#include <stdio.h>

int main() {
    int a[5];
    int s = 0;
    for (int i = 0; i < 5; i++) {
        scanf("%d", &a[i]);
    }
    for (int i = 0; i < 5; i++) {
        if (a[i] < 40) a[i] = 40;
        s += a[i];
    }
    printf("%d", s / 5);
    return 0;
}
```

해설 2

굳이 배열을 사용하지 않고도 문제를 풀 수 있다. 임시로 입력을 받는 정수 하나를 선언하고, 입력을 5번 받으면서 합에 누적해 주면 된다.

정답 소스 2

```c
#include <stdio.h>

```

```c
int main() {
    int s = 0, x;
    for (int i = 0; i < 5; i++) {
        scanf("%d", &x);
        if (x < 40) x = 40;
        s += x;
    }
    printf("%d", s / 5);
    return 0;
}
```

문제 출처

第13回日本情報オリンピック(JOI 2013/2014), 予選 1번

문제 통계

정답: 6명 중 6명 (100%)

첫 정답자: ilsole1025 (이수빈) / 대회 시작 후 2분

Problem B

이항 계수 1

시간 제한: 1 초

메모리 제한: 256 MB

자연수 N과 정수 K가 주어졌을 때 이항 계수 $\binom{N}{K}$를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 N과 K가 주어진다. ($1 \leq N \leq 10, 0 \leq K \leq N$)

출력

$\binom{N}{K}$를 출력한다.

예제 입력	예제 출력
5 2	10

힌트

$\binom{N}{K} = {}_N\mathrm{C}_K$

해설 1

$n! = n(n-1)!$ 혹은 $\binom{N}{K} = \binom{N-1}{K-1} + \binom{N-1}{K}$ 임을 이용해 재귀 함수를 구성할 수 있다.

정답 소스 1

```c
#include <stdio.h>

int binom(int n, int k) {
    if (n == k) return 1;
    if (k == 0) return 1;
    return binom(n - 1, k - 1) + binom(n - 1, k);
}

int main() {
    int N, K;
    scanf("%d%d", &N, &K);
    printf("%d", binom(N, K));
}
```

해설 2

위의 점화식을 이용하지만 재귀 함수를 사용하지 않는 방법도 있다. 2차원 배열을 이용하면 된다.

정답 소스 2

```c
#include <stdio.h>

int main() {
    int binom[11][11];
    int N, K;
    scanf("%d%d", &N, &K);

    binom[0][0] = 0;
```

```
    for (int i = 1; i <= N; i++) {
        binom[i][0] = 1;
        for (int j = 1; j < i; j++) {
            binom[i][j] = binom[i - 1][j - 1] + binom[i - 1][j];
        }
        binom[i][i] = 1;
    }

    printf("%d", binom[N][K]);
}
```

문제 출처

백준 온라인 저지, 11050번

문제 통계

정답: 6명 중 6명 (100%)

첫 정답자: ihatbuet (김준서), ilsole1025 (이수빈) / 대회 시작 후 5분

Problem C
알파벳 거리
시간 제한: 1 초
메모리 제한: 128 MB

길이가 같은 두 단어가 주어졌을 때, 각 단어에 포함된 모든 글자의 알파벳 거리를 구하는 프로그램을 작성하시오.

두 글자 x와 y 사이의 알파벳 거리를 구하려면, 먼저 각 알파벳에 숫자를 할당해야 한다. 'A' $= 1$, 'B' $= 2, \cdots,$ 'Z' $= 26$. 그 다음 $y \geq x$인 경우에는 $y - x$, $y < x$인 경우에는 $(y + 26) - x$가 알파벳 거리가 된다. 예를 들어, 'B'와 'D' 사이의 거리는 $4 - 2 = 2$이고, 'D'와 'B' 사이의 거리는 $(2 + 26) - 4 = 24$이다.

입력

첫째 줄에 테스트 케이스의 수(< 100)가 주어진다. 각 테스트 케이스는 한 줄로 이루어져 있고, 두 단어가 공백으로 구분되어져 있다. 단어의 길이는 4보다 크거나 같고, 20보다 작거나 같으며, 알파벳 대문자로만 이루어져 있다.

출력

각 테스트 케이스마다 각 글자의 알파벳 거리를 공백으로 구분해 출력한다.

예제 입력

```
5
AAAA ABCD
ABCD AAAA
DARK LOKI
STRONG THANOS
DEADLY ULTIMO
```

예제 출력

```
Distances: 0 1 2 3
Distances: 0 25 24 23
Distances: 8 14 19 24
Distances: 1 14 9 25 1 12
Distances: 17 7 19 5 1 16
```

해설

C에서 문자와 문자열을 다룬다면 다음과 같은 사항들을 사용할 수 있다.

- 문자는 정수형으로 다룰 수 있다.
- 문자열의 끝은 '\0'로 나타낸다.

첫 번째 사항을 이용해 알파벳 두 개의 거리를 구하는 함수를 작성하면 다음과 같다.

$$f(x,y) = \begin{cases} y-x & \text{if } y \geq x \\ y+26-x & \text{if } y < x \end{cases}$$

이는 간단한 if 문으로 구현할 수 있다.
단어는 20글자까지 들어올 수 있으므로 단어를 저장하는 문자 배열의 크기는 맨 뒤에 '\0'가 붙는 것을 고려하여 21로 정한다. 이제 각 문자열의 0번째 위치부터 비교하여 미리 만들어 둔 함수로 알파벳 거리를 계산해 출력한다. 문자가 '\0'라면 출력을 종료한다.

정답 소스

```c
#include <stdio.h>

int dist(char x, char y) {
    if (y >= x) return y - x;
    else return y + 26 - x;
}

int main() {
    int n;
    char word1[21], word2[21];
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%s %s", word1, word2);
        printf("Distances: ");
        for (int j = 0; j < 21; j++) {
            if (word1[j] == '\0') break;
```

```
            printf("%d ", dist(word1[j], word2[j]));
        }
        printf("\n");
    }
    return 0;
}
```

문제 출처

University of Maryland High School Programming Contest (HSPC), 2012, P2번

문제 통계

정답: 6명 중 5명 (83%)

첫 정답자: ilsole1025 (이수빈) / 대회 시작 후 8분

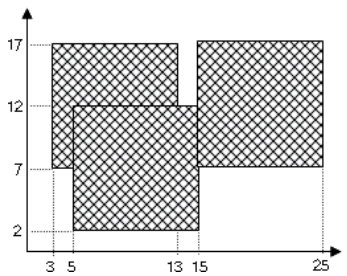
Problem D

색종이

시간 제한: 1 초

메모리 제한: 128 MB

가로, 세로의 크기가 각각 100인 정사각형 모양의 흰색 도화지가 있다. 이 도화지 위에 가로, 세로의 크기가 각각 10인 정사각형 모양의 검은색 색종이를 색종이의 변과 도화지의 변이 평행하도록 붙인다. 이러한 방식으로 색종이를 한 장 또는 여러 장 붙인 후 색종이가 붙은 검은 영역의 넓이를 구하는 프로그램을 작성하시오.



예를 들어 흰색 도화지 위에 세 장의 검은색 색종이를 그림과 같은 모양으로 붙였다면 검은색 영역의 넓이는 260이 된다.

입력

첫째 줄에 색종이의 수가 주어진다. 이어 둘째 줄부터 한 줄에 하나씩 색종이를 붙인 위치가 주어진다. 색종이를 붙인 위치는 두 개의 자연수로 주어지는데 첫 번째 자연수는 색종이의 왼쪽 변과 도화지의 왼쪽 변 사이의 거리이고, 두 번째 자연수는 색종이의 아래쪽 변과 도화지의 아래쪽 변 사이의 거리이다. 색종이의 수는 100 이하이며, 색종이가 도화지 밖으로 나가는 경우는 없다.

출력

첫째 줄에 색종이가 붙은 검은 영역의 넓이를 출력한다.

예제 입력	예제 출력
3 3 7 15 7 5 2	260

해설

도화지를 100×100의 2차원 배열로 생각하자. 각 칸은 1일 경우 색종이가 있는 상태이고, 0일 경우 없는 상태라고 생각하자. 그러면 도화지를 처음에 전부 0으로 초기화해 두고, 색종이의 좌표가 들어올 때마다 색종이가 존재하는 영역을 1로 색칠해 줄 수 있다. 마지막에는 도화지의 모든 칸의 합을 구해 출력한다.

정답 소스

```c
#include <stdio.h>

int main() {
    int filled[100][100];
    for (int i = 0; i < 100; i++) {
        for (int j = 0; j < 100; j++) {
            filled[i][j] = 0;
        }
    }

    int n;
    scanf("%d", &n);
    for (int t = 0; t < n; t++) {
        int x, y;
        scanf("%d%d", &x, &y);
        for (int i = x; i < x + 10; i++) {
            for (int j = y; j < y + 10; j++) {
                filled[i][j] = 1;
            }
        }
    }

    int s = 0;
    for (int i = 0; i < 100; i++) {
        for (int j = 0; j < 100; j++) {
```

```
26              s += filled[i][j];
27          }
28      }
29
30      printf("%d", s);
31  }
```

문제 출처

한국정보올림피아드 시·도 지역본선, 2007, 초등부 2번

문제 통계

정답: 6명 중 3명 (50%)

첫 정답자: ilsole1025 (이수빈) / 대회 시작 후 15분

Problem E
잃어버린 괄호
시간 제한: 2 초
메모리 제한: 128 MB

세준이는 양수와 +, −, 그리고 괄호를 가지고 길이가 최대 50인 식을 만들었다. 그리고 나서 세준이는 괄호를 모두 지웠다.

그리고 나서 세준이는 괄호를 적절히 쳐서 이 식의 값을 최소로 만들려고 한다.

괄호를 적절히 쳐서 이 식의 값을 최소로 만드는 프로그램을 작성하시오.

입력

첫째 줄에 식이 주어진다. 식은 '0'–'9', '+', 그리고 '−'만으로 이루어져 있고, 가장 처음과 마지막 문자는 숫자이다. 그리고 연속해서 두 개 이상의 연산자가 나타나지 않고, 5자리보다 많이 연속되는 숫자는 없다. 수는 0으로 시작할 수 있다.

출력

첫째 줄에 정답을 출력한다.

예제 입력	예제 출력
55-50+40	-35

해설

아래 예시를 참고하면 '−' 기호 뒤에 나오는 모든 '+' 기호들은 괄호를 적절히 쳐서 모두 '−'로 만들 수 있다.

$$\begin{aligned} & 34+27+56-3+97+23-10+5 \\ \Rightarrow & 34+27+56-(3+97+23)-(10+5) \\ = & 34+27+56-3-97-23-10-5 \end{aligned}$$

따라서 처음 '−' 기호가 나온 뒤의 모든 기호를 '−'로 취급해 식을 계산하면 된다. '−'가 등장했다면 1이고 아직 등장하지 않았다면 0인 플래그를 설정해 구현할 수 있다.

정답 소스

```c
#include <stdio.h>

int main() {
    int sum = 0;
    int flag = 0;
    int x;
    char next_sign;
    while (1) {
        scanf("%d%c", &x, &next_sign);
        if (flag) sum -= x; else sum += x;
        if (next_sign == '\n') break;
        if (next_sign == '-') flag = 1;
    }
    printf("%d", sum);
}
```

문제 출처

백준 온라인 저지, 1541번

문제 통계

정답: 6명 중 3명 (50%)

첫 정답자: kevink1113 (강상원) / 대회 시작 후 28분

Problem F

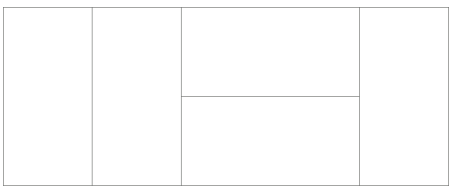
$2 \times n$ 타일링

시간 제한: 1 초

메모리 제한: 256 MB

$2 \times n$ 크기의 직사각형을 1×2, 2×1 타일로 채우는 방법의 수를 구하는 프로그램을 작성하시오.

아래 그림은 2×5 크기의 직사각형을 채운 한 가지 방법의 예이다.



입력

첫째 줄에 n이 주어진다. ($1 \le n \le 1{,}000$)

출력

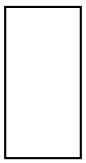
첫째 줄에 $2 \times n$ 크기의 직사각형을 채우는 방법의 수를 10,007로 나눈 나머지를 출력한다.

예제 입력 1	예제 출력 1
2	2

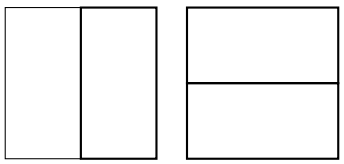
예제 입력 2	예제 출력 2
9	55

해설

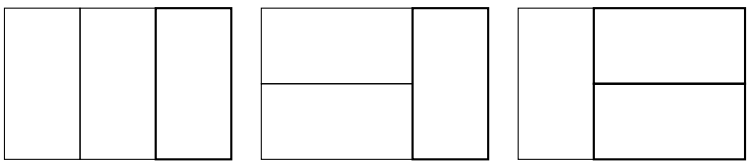
$n = 1$일 때는 아래와 같이 1가지 경우만 가능하다.



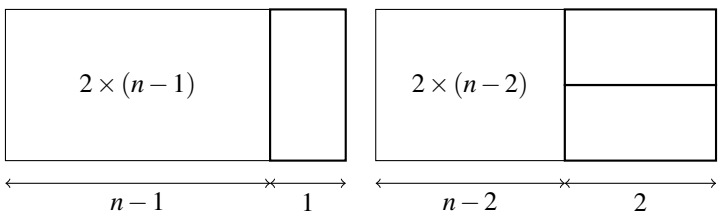
$n = 2$일 때는 2가지 경우가 가능하다.



$n = 3$일 때는 3가지 경우가 가능하다.



이를 바탕으로 생각해 보면, $2 \times n$ 크기의 직사각형을 채우는 것은 $2 \times (n-1)$ 크기의 직사각형을 채운 후 오른쪽에 2×1 타일을 하나 붙이거나 $2 \times (n-2)$ 크기의 직사각형을 채운 후 오른쪽에 1×2 타일을 두 개 붙이는 경우로 나누어지는 것을 알 수 있다.



따라서 $2 \times n$ 크기의 직사각형을 채우는 경우의 수를 a_n이라고 할 때

$$a_n = \begin{cases} 1 & \text{if } n = 1 \\ 2 & \text{if } n = 2 \\ a_{n-2} + a_{n-1} & \text{otherwise} \end{cases}$$

임을 알 수 있다. 이는 피보나치 수열이라고도 한다.

하지만 이 문제를 단순히 재귀 함수 작성으로 접근하면 시간 초과를 받게 된다. 예를 들어 a_7은

$$\begin{aligned}
a_7 &= a_5 + a_6 \\
&= a_3 + a_4 + a_4 + a_5 \\
&= a_1 + a_2 + a_2 + a_3 + a_2 + a_3 + a_3 + a_4 \\
&= 1 + 2 + 2 + a_1 + a_2 + 2 + a_1 + a_2 + a_1 + a_2 + a_2 + a_3 \\
&= 1 + 2 + 2 + 1 + 2 + 2 + 1 + 2 + 1 + 2 + 2 + a_1 + a_2 \\
&= 1 + 2 + 2 + 1 + 2 + 2 + 1 + 2 + 1 + 2 + 2 + 1 + 2 \\
&= 21
\end{aligned}$$

으로 계산되는데, 계산 과정을 보면 이미 계산한 a_3이나 a_4 등을 다시 계산하고 있다. 문제의 조건에서 $n \leq 1{,}000$인데, 단순히 재귀로 a_{1000}을 계산하면 함수가 4.35×10^{199}번 호출되게 된다. 이는 상당히 비효율적이다. 따라서 다른 방법을 생각해야 한다.

a_{n-2}와 a_{n-1}이 구해져 있으면 a_n도 구할 수 있다는 점에서 착안해, a를 배열로 만들면 아래와 같이 대략 n번만의 연산에 a_n을 구할 수 있다.

정답 소스

```c
#include <stdio.h>

int main() {
    int a[1001];
    a[1] = 1;
    a[2] = 2;

    int n;
    scanf("%d", &n);
    for (int i = 3; i <= n; i++) {
        a[i] = (a[i - 2] + a[i - 1]) % 10007;
    }
    printf("%d", a[n]);
}
```

문제 출처

백준 온라인 저지, 11726번

문제 통계

정답: 6명 중 1명 (17%)

첫 정답자: kevink1113 (강상원) / 대회 시작 후 33분

Problem G

책 페이지

시간 제한: 2 초

메모리 제한: 128 MB

지민이는 N쪽인 책이 한권 있다. 첫 페이지는 1쪽이고, 마지막 페이지는 N쪽이다. 각 숫자가 모두 몇 번이 나오는지 출력하는 프로그램을 작성하시오.

입력

첫째 줄에 N이 주어진다. N은 1,000,000,000보다 작거나 같은 자연수이다.

출력

첫째 줄에 0이 총 몇 번 나오는지, 1이 총 몇 번 나오는지, ⋯, 9가 총 몇 번 나오는지를 출력한다.

예제 입력	예제 출력
11	1 4 1 1 1 1 1 1 1 1

해설

1–n까지에 등장하는 숫자의 개수를 세기 위해 문제를 여러 구간으로 쪼개 더한다고 생각하자. 이를 위해 구간 a–b에 숫자가 몇 번씩 등장하는지 살펴보고자 한다.

일단 a를 0으로 끝나게 만들고, b가 9로 끝나게 만들 수 있다. 만약 처음에 a가 338, b가 981이었다면 a–b는

$$338,\ 339,\ 340\text{–}979,\ 980,\ 981$$

이므로 338, 339, 980, 981에 대해서만 등장하는 숫자의 개수를 더해 줄 수 있다.

a를 0으로 끝나게 만들고, b가 9로 끝나게 만들면 a–b의 1의 자리에는 0–9가 각각 $\dfrac{b-9-a}{10}+1$번씩 등장하게 된다.

$$\left.\begin{array}{cccccccccc} 340 & 341 & 342 & 343 & 344 & 345 & 346 & 347 & 348 & 349 \\ & & & & \vdots & & & & & \\ 970 & 971 & 972 & 973 & 974 & 975 & 976 & 977 & 978 & 979 \end{array}\right\} \frac{979-9-340}{10}+1\text{개}$$

1의 자리에 있는 숫자들은 전부 세었으므로, 이제 a와 b의 마지막 자리를 지우고 위 과정을 반복해서 각각 a', b'를 얻자. 예를 들어 a가 340, b가 979였다면 마지막 자리를 지워 34, 97이 되고

$$34,\ \cdots,\ 39,\ 40\text{–}89,\ 90,\ \cdots,\ 97$$

으로 생각할 수 있는데, 34, $\cdots$, 39, 90, $\cdots$, 97에 등장하는 숫자의 개수를 α리 두자. 그러면 a'–b'의 1의 자리에는 0–9가 각각 $\dfrac{b'-9-a'}{10}+1$번씩 등장하며 원래 범위 a–b에서는 10의 자리에 0–9가 각각 $10\left(\dfrac{b'-9-a'}{10}+1+\alpha\right)$번씩 등장하게 된다.

$$\left.\begin{array}{cccccccccc} 40\times & 41\times & 42\times & 43\times & 44\times & 45\times & 46\times & 47\times & 48\times & 49\times \\ & & & & \vdots & & & & & \\ 80\times & 81\times & 82\times & 83\times & 84\times & 85\times & 86\times & 87\times & 88\times & 89\times \end{array}\right\} \frac{89-9-40}{10}+1\text{개}$$

이를테면 1–346194에서 등장하는 숫자의 개수를 구한다면

- (1, $\cdots$, 9, **10–346189**, 346190, $\cdots$, 346194)의 1의 자리에서 등장하는 숫자의 개수
- (1, $\cdots$, 9, **10–34609**, 34610, $\cdots$, 34618)의 1의 자리에서 등장하는 숫자의 개수 $\times 10^1$

- (1, ⋯, 9, **10–3459**, 3460)의 1의 자리에서 등장하는 숫자의 개수 $\times 10^2$
- (1, ⋯, 9, **10–339**, 340, ⋯, 345)의 1의 자리에서 등장하는 숫자의 개수 $\times 10^3$
- (1, ⋯, 9, **10–29**, 30, ⋯, 33)의 1의 자리에서 등장하는 숫자의 개수 $\times 10^4$
- (1, 2)의 1의 자리에서 등장하는 숫자의 개수 $\times 10^5$

로 구하게 되는 것이다. 이렇게 문재를 재귀적으로 해결할 수 있다.

정답 소스

이 소스 코드의 저작권은 스타트링크에 있다.

```c
#include <stdio.h>

void calc(int n, int ten, int *ans) {
    while (n > 0) {
        ans[n % 10] += ten;
        n = n / 10;
    }
}

void go(int start, int end, int ten, int *ans) {
    while (start % 10 != 0 && start <= end) {
        calc(start, ten, ans);
        start += 1;
    }
    if (start > end) return;
    while (end % 10 != 9 && start <= end) {
        calc(end, ten, ans);
        end -= 1;
    }
    int cnt = (end / 10 - start / 10 + 1);
    for (int i = 0; i <= 9; i++) {
        ans[i] += cnt * ten;
    }
```

```
24      go(start / 10, end / 10, ten * 10, ans);
25  }
26
27  int main() {
28      int n;
29      int ans[10];
30      scanf("%d", &n);
31      for (int i = 0; i < 10; i++) {
32          ans[i] = 0;
33      }
34
35      go(1, n, 1, ans);
36      for (int i = 0; i <= 9; i++) {
37          printf("%d ", ans[i]);
38      }
39
40      return 0;
41  }
```

문제 출처

백준 온라인 저지, 1019번

문제 통계

정답: 6명 중 2명 (33%)

첫 정답자: kevink1113 (강상원) / 대회 시작 후 41분

Problem H

Eakspay igpay atinlay?

시간 제한: 1 초

메모리 제한: 256 MB

당신은 영어를 돼지 라틴어로 번역하는 작업을 맡게 되었다. 영어를 돼지 라틴어로 번역하려면 두 가지 규칙만 알고 있으면 된다. 만약에 어떤 단어가 자음으로 시작한다면, 단어 앞부분에 등장하는 연속된 자음들을 떼서 단어 뒤에 붙이고 끝에 “a”, “y”를 붙인다. 만약에 어떤 단어가 모음으로 시작한다면, 단순히 단어 뒤에 “y”, “a”, “y”를 붙인다. 이 문제에서 “y”는 모음으로 취급한다. 단어들은 모두 소문자로 이루어져 있다고 생각해도 좋다.

각 줄은 200자를 넘지 않는다.

입력

첫 번째 줄에는 번역해야 하는 문장의 수 N이 주어진다. 다음 N개의 줄에는 줄마다 번역해야 하는 문장이 주어진다.

출력

줄마다 영어를 돼지 라틴어로 번역한 결과를 출력한다.

예제 입력

```
2
twist of fate
hotter than the fourth of july
```

예제 출력

```
isttway ofyay atefay
otterhay anthay ethay ourthfay ofyay ulyjay
```

해설

우선 모음을 판단하는 데 매번 긴 식을 쓰는 것은 번거로우므로 어떤 문자가 모음인지 아닌지를 쉽게 판단할 수 있는 함수를 하나 만들도록 하자.
C에서는 `fgets` 함수를 사용해 한 줄을 문자열로 전부 읽을 수 있다. 혹은 '\n'이 나올 때까지 한 글자씩 읽는 방법을 사용할 수도 있다. 이 방법을 이용해 문자열 배열에 입력 문자열을 저장한다. 코드에서는 `original` 배열에 저장했다.
답을 저장할 `ans` 배열을 만들고, 단어마다 문자열을 처리하도록 하자. 이는 while 루프로 가능하다. 만약 현재 확인하는 문자가 '\n'이라면 이는 줄의 끝을 의미하므로 while 루프에서 탈출하도록 하며, 현재 확인하는 문자가 공백 문자라면 단어의 끝을 의미하므로 적절한 처리를 해 주도록 한다.
현재 원래 문자열에서 확인할 문자 인덱스를 `o_idx`, 답을 저장할 문자열에서의 현재 커서 위치를 `a_idx`라고 하면,

- 단어의 첫 글자가 **모음**일 때 코드의 19–21번째 줄은 단어가 끝날 때까지 원래 문자열의 `o_idx`에 있는 문자를 답 문자열의 `a_idx`로 옮기는 역할을 하고, 23–30번째 줄은 답 문자열 마지막에 'yay'와 공백 문자 하나를 적는 역할을 한다.
- 단어의 첫 글자가 **자음**일 때 코드의 32번째 줄은 단어의 시작 인덱스를 저장하고, 33–38번째 줄은 첫 번째 모음이 나오는 인덱스를 계산한다. 이후에 39–43번째 줄에서는 `o_idx`는 첫 번째 모음이 나오는 인덱스로 설정되어 있으므로 첫 글자가 모음일 때처럼 단어가 끝날 때까지 글자들을 옮기고, 44–47번째 줄에서는 미리 계산해 둔 인덱스들을 이용해 단어 처음에 등장한 자음들을 뒤로 옮겨 준다. 마지막으로 48–50번째 줄은 답 문자열 마지막에 'ay'와 공백 문자 하나를 적는 역할을 한다.

이 과정이 끝나면 답 문자열은 공백 문자로 끝날 것이다. 문자열이 끝났음을 명시하기 위해 56–57번째 줄에서는 마지막 공백 문자를 지우고 '\0'으로 대체한다.
이와 같은 과정을 각 테스트 케이스마다 반복하기 쉽게 하기 위해 이 과정 전체를 `solve()`라는 함수로 만들 수 있겠다.

정답 소스

```
1  #include <stdio.h>
2
3  int is_vowel(char c) {
```

```
    return (c == 'a' || c == 'e' || c == 'i' ||
            c == 'o' || c == 'u' || c == 'y');
}

void solve() {
    char original[201], ans[1000];
    int o_idx = 0, a_idx = 0;
    fgets(original, 201, stdin);

    while (1) {
        if (original[o_idx] == ' ') o_idx++;
        if (original[o_idx] == '\n') break;

        if (is_vowel(original[o_idx])) {
            while (original[o_idx] != ' ' && original[o_idx] != '\n') {
                ans[a_idx] = original[o_idx];
                a_idx++;
                o_idx++;
            }
            ans[a_idx] = 'y';
            a_idx++;
            ans[a_idx] = 'a';
            a_idx++;
            ans[a_idx] = 'y';
            a_idx++;
            ans[a_idx] = ' ';
            a_idx++;
        } else {
            int o_word_start_idx = o_idx;
            while (!is_vowel(original[o_idx]) &&
                   original[o_idx] != ' ' &&
                   original[o_idx] != '\n') {
```

```
                o_idx++;
            }
            int o_word_vowel_idx = o_idx;
            while (original[o_idx] != ' ' && original[o_idx] != '\n') {
                ans[a_idx] = original[o_idx];
                a_idx++;
                o_idx++;
            }
            for (int o = o_word_start_idx; o < o_word_vowel_idx; o++) {
                ans[a_idx] = original[o];
                a_idx++;
            }
            ans[a_idx] = 'a';
            a_idx++;
            ans[a_idx] = 'y';
            a_idx++;
            ans[a_idx] = ' ';
            a_idx++;
        }
    }
    a_idx--;
    ans[a_idx] = '\0';
    printf("%s\n", ans);
}

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

문제 출처

2nd Annual Cornell University High School Programming Contest, 2번

문제 통계

정답: 6명 중 0명 (0%)

Problem I
sqrt log sin
시간 제한: 1 초
메모리 제한: 128 MB

도현이는 수학 숙제를 하고 있다. 문제는 다음과 같다.
다음과 같이 재귀적으로 정의된 수열이 있다.

$$x_0 = 1$$

$$x_i = x_{\lfloor i-\sqrt{i} \rfloor} + x_{\lfloor \ln(i) \rfloor} + x_{\lfloor i \sin^2(i) \rfloor}$$

i가 주어졌을 때, x_i를 구하는 프로그램을 작성하시오.

입력
입력은 여러 개의 테스트 케이스로 이루어져 있으며, 한 줄에 하나씩 주어진다.
각 줄에는 i가 주어지며, 이 수는 0보다 작지 않고, 1,000,000보다 크지 않다.
입력의 마지막 줄에는 -1이 주어지며, 이 수는 입력의 마지막을 나타내는 수이다.

출력
입력으로 주어진 i마다 x_i를 1,000,000로 나눈 나머지를 출력한다.

예제 입력	예제 출력
0 -1	1

힌트
$\lfloor x \rfloor$는 x보다 작거나 같은 정수들 중 가장 큰 정수를 나타내는 x에 대한 함수이다.
C의 경우 `math.h` 헤더에서 다음 함수들을 사용할 수 있다.

- `double sin(double x)` $\rightarrow \sin x$
- `double log(double x)` $\rightarrow \ln x$
- `double sqrt(double x)` $\rightarrow \sqrt{x}$
- `double floor(double x)` $\rightarrow \lfloor x \rfloor$

해설

$i \geq 1$의 경우 $\sin^2(i) \leq 1$이므로 $\lfloor i - \sqrt{i} \rfloor$, $\lfloor \ln(i) \rfloor$, $\lfloor i \sin^2(i) \rfloor$은 전부 i보다 작다. 따라서 문제 **F**처럼 $x_{\lfloor i-\sqrt{i} \rfloor}$, $x_{\lfloor \ln(i) \rfloor}$, $x_{\lfloor i \sin^2(i) \rfloor}$가 모두 구해져 있다면 x_i도 구할 수 있으며, 따라서 i를 1에서부터 증가시켜 가면서 모든 x_i를 구할 수 있다.

다만 이 문제의 경우 입력이 여러 개의 테스트 케이스로 주어진다. 이런 경우 테스트 케이스가 들어올 때마다 x_i를 새로 계산하는 것은 비효율적일 것이다. 따라서 $x_{1000000}$까지를 전부 계산해 두고, 필요할 때마다 계산되어 있는 x_i를 출력하는 방식으로 접근하는 것이 바람직하다.

정답 소스

```c
#include <stdio.h>
#include <math.h>

int main() {
    int x[1000001];
    x[0] = 1;

    for (int i = 1; i <= 1000000; i++) {
        x[i] = (
            x[(int) floor(i - sqrt(i))] +
            x[(int) floor(log(i))] +
            x[(int) floor(i * sin(i) * sin(i))]
        ) % 1000000;
    }

    int i;
    while (1) {
        scanf("%d", &i);
        if (i == -1) break;
        printf("%d\n", x[i]);
    }
```

```
22
23      return 0;
24  }
```

문제 출처

Waterloo’s local Programming Contests, 2009. 8. 3, E번

문제 통계

정답: 6명 중 0명 (0%)