2019 숭고한 연합

Algorithm Contest

- 초급반 -

대회가 시작되기 전까지 절대 표지를 넘기지 마세요

< Hosted by >

SCCC AlKor ALOHA

< Sponsored by >

NAVER D2 Streami STARTLINK

SAMSUNG SOFTWARE MEMBERSHIP

고려대학교 SW 중심대학사업단

대회는 총 **7문제**이며, 구성은 다음과 같습니다. 총 문제지가 표지를 제외하고 13쪽인지 확인하시길 바랍니다.

A 와글와글 숭고한

B 보너스 점수

C 이건 꼭 풀어야 해!

D 무한부스터

E 우울한 방학

F 다이나믹 롤러

G 핑거 스냅

- 대회 시간은 **4시간**입니다.
- **대회가 진행되는 동안 대회 사이트를 제외한 웹에 접속/열람/검색 등을 할 수 없습니다. 예외로, 공지를 통해 명시된 레퍼런스 사이트는 열람할 수 있습니다.**
- **이전에 저장해두었던 자료/코드 등을 열람 및 사용할 수 없습니다. 예외로, 주최진 측에서 제공하는 '템플릿 코드 문서'는 열람할 수 있습니다.**
- 그 외에 운영진이 판단하기에 악의적인 제출을 진행하거나 부정 행위를 저지를 경우 경고 없이 대회 참가 자격이 박탈될 수 있습니다.
- 컴파일 환경은 BOJ의 기본 환경을 따릅니다.
- 대회 중 '질문' 탭을 통해 주최진에게 물을 수 있습니다. 예/아니오로 답이 나오는 질문을 해주시기 바랍니다.
- **문제지는 참가자의 편의를 위해 제공되었을 뿐, 웹사이트의 내용이 공식 문제입니다. 인쇄본과 사이트의 내용 중 상이한 부분이 있을 경우, 사이트에 기재된 설명을 참고하시길 바랍니다.**

문제 A. 와글와글 숭고한

시간 제한: 1 초
메모리 제한: 512 MB

숭고한 알고리즘 캠프가 다가오고 있고 방학이 되어서까지도 각 대학들의 협업은 계속되고 있다. 그럼에도 불구하고 운영진들과 강사진들이 각자의 일정 때문에 바빠 계획에 차질이 조금씩 생기고 있다. 숭고한 알고리즘 캠프의 대표인 창호는 효율적인 일처리를 위해 엄정한 평가를 내리기로 하였다.

창호는 숭고한 알고리즘 캠프의 구성원인 숭실대학교(Soongsil University), 고려대학교(Korea University), 한양대학교(Hanyang University)의 참여도를 수치화하였다. 창호가 보기에 세 대학교의 참여도의 합이 100 이상이면 일처리가 잘 되고 있기에 안심할 수 있지만, 100 미만이면 창호는 참여도가 가장 낮은 대학의 동아리에게 무언의 압박을 넣을 예정이다. 숭고한 알고리즘 캠프의 성공을 빌며 창호의 고민을 해결해주자.

입력 형식

입력의 첫 줄에는 숭실대학교의 참여도, 고려대학교의 참여도, 한양대학교의 참여도를 의미하는 세 자연수 S, K, H가 공백으로 구분되어 주어진다. ($0 \leq S,\ K,\ H \leq 100$)

세 대학의 참여도는 모두 다르다.

출력 형식

첫 번째 줄에 일처리가 잘 되고 있어 무언의 압박이 필요가 없으면 (따옴표를 제외하고) "OK"를 출력한다.

그 외에는 첫 번째 줄에 무언의 압박이 필요한 동아리가 속한 대학의 영문 이름의 첫 단어를 출력한다. 영문 이름 표기는 지문에 나온 것을 따른다.

예제

표준 입력	표준 출력
31 41 59	OK
1 2 3	Soongsil
19 8 9	Korea
45 33 21	Hanyang

문제 B. 보너스 점수

시간 제한: 1 초
메모리 제한: 512 MB

숭고한 알고리즘 캠프 퀴즈 타임이 시작되었다! PS 기초, 동적 계획법, 파라메트릭 서치, 욱제의 생일, 탐색, 그리디, 최단경로 알고리즘, 구데기컵, 서로소 집합, 최소 신장 트리, 최소 공통 조상, 세그먼트 트리, 코드포스에서 C++로 높은 수준의 난수를 생성하는 방법, 최대 유량, 볼록 껍질, 스타트링크 사무실에 있는 게임용 컴퓨터의 RAM의 총 용량 등등 수많은 주제를 총망라하고 있는 이 미니 대회는 수많은 참가자들의 도전으로 오늘도 빛나고 있고, 제출된 OX표의 개수는 셀 수 없을 정도이다.

운영진들은 이 OX표들을 채점하고, 점수를 계산해낸 다음, 시상식을 진행하며 화기애애하게 행사를 마무리해야 한다. 숭고한 알고리즘 캠프 퀴즈 타임에서 OX표의 점수는 다음과 같이 계산된다.

- OX표에 N개의 문제들이 있을 때, 1번 문제, 2번 문제, $\cdots$, N번 문제 순으로 채점된다.
- 문제는 뒤로 갈수록 어려워지기 때문에, i번 문제의 기본 점수는 i점이다.
  - 문제를 맞히면 그 문제의 기본 점수(즉 i번 문제의 경우 i점)를 획득하며, 틀리면 얻지 못한다.
- 기본 점수와 별개로, '보너스 점수'라는 값이 존재한다. 이는 처음에는 0점이다.
  - 문제를 맞히면 그 때의 '보너스 점수'를 획득하고, '보너스 점수'의 값이 1점 증가한다.
  - 문제를 틀리면 '보너스 점수'를 얻지 못하고, '보너스 점수'의 값이 0점으로 초기화된다.

민성이는 얼떨결에 숭고한 알고리즘 캠프 퀴즈 타임의 OX표를 채점해야 하는 업무를 맡게 되었다. 수많은 OX표를 볼 생각에 머리가 지끈거리는 민성이는 프로그램을 세워 이를 자동화하려고 한다. 시상식까지 4시간밖에 남지 않은 민성이를 도와 점수를 계산해주자.

입력 형식

첫 번째 줄에 OX표의 길이인 자연수 N이 주어진다. $(1 \le N \le 10,000)$

두 번째 줄에 OX표를 의미하는 문자열 S가 주어진다. S는 `O`(알파벳 대문자 O, ASCII 코드 79)와 `X`(알파벳 대문자 X, ASCII 코드 88)로만 구성되어 있으며, 길이는 N이다.

문자열 S의 i번째 글자가 `O`이면 해당 참가자가 i번째 문제를 맞혔음을 의미하고, `X`이면 틀렸음을 의미한다.

출력 형식

첫 번째 줄에 입력으로 들어온 OX표의 점수를 출력한다.

예제

표준 입력	표준 출력
8 XOOOXOOX	26

설명

민성이는 다음과 같이 총 26점을 받게 된다.

- 1번 문제를 틀렸으므로 점수를 얻지 못하고 보너스 점수가 0점으로 초기화된다. (총 점수 0점)
- 2번 문제를 맞혔으므로 기본 점수 2점과 보너스 점수 0점을 획득하며, 보너스 점수가 1점으로 증가한다. (총 점수 2점)
- 3번 문제를 맞혔으므로 기본 점수 3점과 보너스 점수 1점을 획득하며, 보너스 점수가 2점으로 증가한다. (총 점수 6점)
- 4번 문제를 맞혔으므로 기본 점수 4점과 보너스 점수 2점을 획득하며, 보너스 점수가 3점으로 증가한다. (총 점수 12점)
- 5번 문제를 틀렸으므로 점수를 얻지 못하고 보너스 점수가 0점으로 초기화된다. (총 점수 12점)
- 6번 문제를 맞혔으므로 기본 점수 6점과 보너스 점수 0점을 획득하며, 보너스 점수가 1점으로 증가한다. (총 점수 18점)
- 7번 문제를 맞혔으므로 기본 점수 7점과 보너스 점수 1점을 획득하며, 보너스 점수가 2점으로 증가한다. (총 점수 26점)
- 8번 문제를 틀렸으므로 점수를 얻지 못하고 보너스 점수가 0점으로 초기화된다. (총 점수 26점)

문제 C. 이건 꼭 풀어야 해!

시간 제한: 1 초
메모리 제한: 512 MB

숭실골 높은 언덕 깊은 골짜기에 출제로 고통 받는 욱제가 살고 있다!

욱제는 또 출제를 해야 해서 단단히 화가 났다. 그래서 욱제는 길이 N짜리 수열 A를 만들고, A를 비내림차순으로 정렬해서 수열 B를 만들어 버렸다!! 여기서 B를 출력하기만 하면 문제가 너무 쉬우니까 하나만 더 하자. 아래와 같은 질문이 무려 Q개나 주어진다!! (ㅎㅎ;; ㅈㅅ.. ㅋㅋ!!)

- **L R**: $B_L + B_{L+1} + \cdots + B_{R-1} + B_R$ 을 출력한다.

Figure 1: 모든 참가자가 문제를 풀 수 있을 것이라고 기대하는 욱제의 표정

욱제의 질문에 답하고 함께 엠티를 떠나자!!

입력 형식

첫 번째 줄에 수열 A의 길이 N과 질문의 개수 Q가 공백으로 구분되어 주어진다. $(1 \le N,\ Q \le 300,000)$

두 번째 줄에 N개의 정수 $A_1,\ A_2,\ \cdots,\ A_N$ 이 공백으로 구분되어 주어진다. A_i는 수열 A의 i번째 수이다. $(1 \le A_i \le 1,000)$

세 번째 줄부터 Q개의 줄에 걸쳐 욱제의 질문을 의미하는 두 수 L, R이 공백으로 구분되어 주어진다. $(1 \le L \le R \le N)$

주어지는 모든 입력은 자연수이다.

출력 형식

Q개의 줄에 걸쳐, 질문의 답을 순서대로 각각 출력한다.

예제

표준 입력	표준 출력
5 6 2 5 1 4 3 1 5 2 4 3 3 1 3 2 5 4 5	15 9 3 6 14 9
5 3 2 5 1 2 3 1 3 2 3 1 5	5 4 13

설명

첫 번째 예제에서 [2, 5, 1, 4, 3]을 비내림차순으로 정렬하면 [1, 2, 3, 4, 5]이다.

두 번째 예제에서 [2, 5, 1, 2, 3]을 비내림차순으로 정렬하면 [1, 2, 2, 3, 5]이다.

참고 사항

비내림차순은 원소가 감소하지 않는 (같거나 증가하는) 순서를 말한다.

```
while (Q--) {
    int sum = 0, L, R;
    scanf("%d %d", &L, &R);
    for (int i = L; i <= R; i++) {
        sum += a[i];
    }
    printf("%d\n", sum);
}
```

위와 같이 각 질문마다 반복문을 매번 돌려서 답을 계산하면, 시간복잡도가 $O(QN)$ 이 되므로 시간 초과를 받게 된다. 다른 방법을 이용해 문제를 해결해야 한다.

문제 D. 무한부스터

시간 제한: 1 초
메모리 제한: 512 MB

카트라이더를 처음 시작하는 카린이 정범이는 어려운 조작법에 실망감이 커져가고 있다. 드리프트, 순간 부스터, 커팅, 톡톡이 등등 어려운 테크닉에 질린 정범이는 그나마 쉬운 '숭고한 무한부스터 모드'에 도전해보려고 한다.

'숭고한 무한부스터 모드'는 크기 $N \times M$ 의 직사각형 모양의 맵에서 진행되며, 맵 전체가 단위 격자로 구성되어 있다. 기존의 '무한부스터 모드'와는 다르게, 모든 격자 안에는 특정 개수의 부스터 아이템이 위치한다. 이 모드에서 플레이의 방식은 다음과 같다.

처음에 플레이어의 카트바디는 출발지점인 1행 1열에 위치하며, 멈춰 있는 상태이고, 보유하고 있는 부스터 아이템의 개수는 0개이다. 목표는 도착지점인 N 행 M 열의 격자에 도달하는 것이며, 도달하는 즉시 게임이 종료된다. 카트바디가 격자에 멈추어 있을 때, 격자에 놓여있는 부스터 아이템을 자동으로 전부 습득하게 된다. 이 과정에서 x 개를 습득했다면 한 방향을 정해 오른쪽으로 최대 x 칸을 가거나, 아래쪽으로 최대 x칸을 이동할 수 있으며, 1칸 단위로 이동하게 된다. 예를 들어 부스터 아이템을 3개 습득했을 때, 오른쪽으로 2칸 이동이나 아래쪽으로 3칸 이동은 가능하지만, 오른쪽으로 1칸 이동 후 아래로 2칸 이동이나 왼쪽으로 1칸 이동이나 아래쪽으로 2.718칸 이동은 불가능하다. **이동 후 멈추면서 보유하고 있던 부스터 아이템은 모두 소진된다**.

이동중에 멈추지 않고 지나치는 격자의 부스터 아이템은 습득할 수 없으며, 카트바디는 맵을 벗어나는 방향으로는 움직일 수 없다.

정범이는 '숭고한 무한부스터 모드'에서 출발지점부터 도착지점까지 주행하면서 부스터 아이템을 획득하게 되는 격자의 개수를 최소화하고 싶다. 카린이 정범이를 도와주도록 하자.

입력 형식

첫 번째 줄에 맵의 세로 길이와 가로 길이를 나타내는 양의 정수 N 과 M 이 공백으로 구분되어 주어진다. $(1 \le N, M \le 300)$

두 번째 줄부터 N 개의 줄에 걸쳐 각 격자에 있는 부스터 아이템 개수인 M 개의 양의 정수 a_{ij} 가 공백으로 구분되어 주어진다. $(1 \le a_{ij} \le \max(N, M))$ a_{ij} 는 i 행 j 열의 격자에 있는 부스터 아이템 개수이다.

출발지점과 도착지점은 다르다.

출력 형식

첫 번째 줄에 정범이가 맵의 출발지점부터 도착지점까지 이동하면서 부스터 아이템을 획득하게 되는 격자의 최소 개수를 출력한다.

예제

표준 입력	표준 출력
4 5 1 1 4 1 3 3 4 1 3 2 1 1 5 3 2 5 3 1 1 1	3
5 4 2 4 2 3 1 1 1 3 2 1 2 2 1 4 4 1 1 2 2 1	3

설명

첫 번째 예제에서 3번 만에 이동하는 최적 경로는 다음과 같다.

- 1행 1열에서 1개의 부스터 아이템을 획득하고, 아래쪽으로 1칸 이동하여 2행 1열에서 멈춘다.
- 2행 1열에서 3개의 부스터 아이템을 획득하고, 아래쪽으로 2칸 이동하여 4행 1열에서 멈춘다.
- 4행 1열에서 5개의 부스터 아이템을 획득하고, 오른쪽으로 4칸 이동하여 4행 5열에서 멈춘다. 도착지점에 도달했기 때문에 게임이 종료된다.

두 번째 예제에서 3번 만에 이동하는 최적 경로는 다음과 같다.

- 1행 1열에서 2개의 부스터 아이템을 획득하고, 오른쪽으로 1칸 이동하여 1행 2열에서 멈춘다.
- 1행 2열에서 4개의 부스터 아이템을 획득하고, 아래쪽으로 4칸 이동하여 5행 2열에서 멈춘다.
- 5행 2열에서 2개의 부스터 아이템을 획득하고, 오른쪽으로 2칸 이동하여 5행 4열에서 멈춘다. 도착지점에 도달했기 때문에 게임이 종료된다.

문제 E. 우울한 방학

시간 제한: 1 초
메모리 제한: 512 MB

방학동안 기숙사에 홀로 남겨진 인호는 우울하고 고독하다. 다행히 인호는 M일의 방학 동안 N개의 약속이 잡혀있기에, 약속 날짜의 효율적인 배치를 통해 방학 내에 느낄 우울함의 합을 최소화하려고 한다.

인호의 기분은 정수로 표현 가능하며, 기분이 0 미만인 날에 $(\text{기분})^2$ 만큼 우울함을 느낀다. 인호의 기분은 오늘 약속이 있다면 약속의 기대행복 값인 H_i이며, 약속이 없으면 어제의 기분에서 1을 뺀 값이다.

인호는 하루에 최대 한 개의 약속을 소화할 수 있으며, N개의 약속들의 순서는 주어진 순서대로여야 한다.

방학은 내일부터 시작이며, 오늘 인호의 기분은 0일 때, 약속을 적절히 배치하여 인호가 방학 동안 느낄 우울함의 합을 최소화하자.

입력 형식

첫 번째 줄에는 인호의 약속 개수인 자연수 N과 방학의 일수인 자연수 M이 공백으로 구분되어 주어진다. $(0 \leq N < M < 1000)$

두 번째 줄에는 N개의 정수 $H_1, H_2, \cdots, H_N$이 공백으로 구분되어 주어진다. H_i는 i번째 약속의 기대행복 값이다. $(1 \leq H_i < 100)$

출력 형식

첫 번째 줄에 인호가 방학 동안 느낄 우울함의 합의 최솟값을 출력한다.

예제

표준 입력	표준 출력
3 10 2 2 1	2

설명

1일, 5일, 8일에 약속을 순서대로 배치하면, 4일과 10일에 각각 1만큼의 우울함을 느끼게 되어, 총 2만큼의 우울함을 느끼게 된다. 이보다 덜 우울함을 느끼게 만드는 방법은 존재하지 않는다.

문제 F. 다이나믹 롤러

시간 제한: 2 초
메모리 제한: 512 MB

페인팅 전문 회사 부키치 암즈는 거대한 페인팅용 롤러 "다이나믹 롤러"를 출시하였다. 이 신제품은 평범한 페인팅 롤러와 마찬가지로 굴려서 칠할 수 있지만, 손잡이를 세로로 휘둘러 잉크를 한번에 흩뿌릴 수도 있도록 설계되었다. 이러한 새로운 사용방법은 거대한 몸집과 맞물려 매우 역동적으로 보였기 때문에, 이 롤러의 이름은 다이나믹 롤러가 되었다. 평소 페인팅에 관심이 많던 멩미는 다이나믹 롤러의 매력에 흠뻑 빠져, 단숨에 다이나믹 롤러를 구매했다. 지금 당장 롤러를 시험해 보고 싶었던 멩미는 통로 일부분을 칠해보기로 했다.

통로는 $1 \times N$ 길이의 일자 형태를 가지고 있고, 통로의 바닥은 1×1 타일로 가득 차있다. 각 타일은 잉크지수 A_i와 점도지수 B_i를 가지고 있다. 타일이 제각각 다른 특성을 가지고 있기 때문에, 멩미는 세심하게 롤러를 휘둘러야만 한다. 멩미가 i번째 타일 위에 서 있을 때, 멩미는 다이나믹 롤러로 현재 위치보다 오른쪽에 있으면서 점도지수가 서 있는 칸의 잉크지수 A_i 이하인 칸을 칠할 수 있다.

통로는 기본적인 관리가 되고 있기 때문에, 각 칸의 잉크지수 A_i는 점도지수 B_i 이상이다. 그러나 깊숙한 통로는 관리에 어려움이 있기 때문에, 점도지수 B_i는 항상 오름차순이다. 이런 상황 속에서 멩미가 통로의 각 타일에서 서 있을 때 다이나믹 롤러로 칠할 수 있는 최대의 칸 수를 구해보자.

입력 형식

첫 번째 줄에 통로의 길이인 자연수 N이 주어진다. $(1 \le N \le 5 \times 10^5)$

두 번째 줄에 N개의 정수 $A_1,\ A_2,\ \cdots, A_N$이 공백으로 구분되어 주어진다. A_i는 i번째 칸의 잉크지수를 의미한다. $(1 \le A_i \le 10^{18})$

세 번째 줄에 N개의 정수 $B_1,\ B_2,\ \cdots, B_N$이 공백으로 구분되어 주어진다. B_i는 i번째 칸의 점도지수를 의미한다. $(1 \le B_i \le 10^{18}, A_i \ge B_i)$

$B_1,\ B_2,\ \cdots,\ B_N$은 오름차순이다. 즉, $1 \le i \le j \le N$을 만족하는 모든 정수 순서쌍 (i, j)에 대해 $B_i \le B_j$가 성립한다.

출력 형식

첫 번째 줄에 N개의 정수 $x_1,\ x_2,\ \cdots,\ x_N$을 공백으로 구분하여 출력한다. x_i는 i번째 칸에 서서 다이나믹 롤러를 사용할 때 최대로 칠할 수 있는 칸의 개수이다.

예제

표준 입력	표준 출력
5 10 10 10 10 10 10 10 10 10 10	4 3 2 1 0
6 60 90 100 88 90 99 60 70 80 85 90 90	0 4 3 0 1 0

설명

첫 번째 예제 설명은 다음과 같다.

- 첫 번째 칸에서는 시작 지점의 잉크지수가 10이다. 두 번째 칸부터 다섯 번째 칸의 점도지수가 10이기에 다섯 번째 칸까지 칠할 수 있다. 그렇기에 총 4칸을 칠할 수 있다.
- 두 번째 칸도 똑같이 3칸, 세 번째 칸은 2칸, 네 번째 칸은 1칸, 다섯 번째 칸은 0칸 칠할 수 있다.

두 번째 예제 설명은 다음과 같다.

- 첫 번째 칸의 잉크지수는 60이다. 60보다 작거나 같은 점도지수를 가진 가장 오른쪽 칸이 첫 번째 칸이므로 답은 0칸이다.
- 두 번째 칸의 잉크지수는 90이다. 여섯 번째이자 마지막 칸의 점도지수가 90이므로 여섯 번째 칸까지 칠할 수 있으므로 답은 4칸이다.
- 세 번째 칸의 잉크지수는 100이다. 마찬가지로 여섯 번째 칸까지 칠할 수 있으므로 답은 3칸이다.
- 네 번째 칸의 잉크지수는 88이다. 88보다 작거나 같은 점도지수를 가진 가장 오른쪽 칸은 네 번째 칸이므로 답은 0칸이다.
- 다섯 번째 칸의 잉크지수는 90이다. 90보다 작거나 같은 점도지수를 가진 칸은 가장 오른쪽 칸은 여섯 번째 칸이므로 답은 1칸이다.
- 여섯 번째 칸의 잉크지수는 99이며, 99보다 작거나 같은 점도지수를 가진 칸은 가장 오른쪽 칸은 여섯 번째 칸이므로 답은 0칸이다.

문제 G. 핑거 스냅

시간 제한: 2 초
메모리 제한: 512 MB

[어벤져스] 시리즈를 보지 않은 사람이라도 '인피니티 건틀렛'이 무엇인지는 다들 알 것이다. 그래도 모르는 사람들을 위해 설명을 하자면, 인피니티 스톤이 모두 모인 인피니티 건틀렛을 끼고 손가락을 튕기면, 사용자가 원하는 것을 할 수 있다. 그러나 반동이 매우 심하기 때문에 그리 많이는 사용할 수 없다.

정신 나간 수학자 Sonaht는 우연히 이 인피니티 건틀렛을 손에 넣게 된다. 그러나 이 인피니티 건틀렛에는 약간의 하자가 있어서, 핑거 스냅으로 할 수 있는 일이 몇가지 없다. 다음은, 핑거 스냅으로 할 수 있는 일을 나열한 것이다.

1. 전 우주의 생명체 수를 현재의 절반으로 한다.
2. 전 우주의 생명체 수를 현재의 1/3로 한다.
3. 전 우주의 생명체 수를 현재보다 하나 늘린다.
4. 전 우주의 생명체 수를 현재보다 하나 줄인다. 이미 전 우주의 생명체 수가 0이라면 할 수 없다.

첫 두 경우에서, 나누어 떨어지지 않으면 몫만 남기고, 나머지는 버린다.

Sonaht는 전 우주의 생명체 수를 목표치 A 이상 B 이하로 줄이려 한다. 그러나 역시나 정신 나간 수학자답게, A 이상 B 이하인 수 중 소수로 만들려 한다 (**어쩌면 A와 B 사이에 소수가 없을지도 모르지만 말이다.**) 소수란, 서로 다른 약수가 1과 자기 자신밖에 없는 수를 의미한다. 그러나 인피니티 건틀렛은 반동이 심하기에, Sonaht는 최대한 적은 수의 핑거 스냅으로 이 목표를 달성하고자 한다. Sonaht가 최소 몇 번의 핑거 스냅을 해야 할지 구해보자.

입력 형식

첫 번째 줄에 테스트 케이스의 개수 T가 주어진다. ($1 \le T \le 10$)

두 번째 줄부터 T개의 줄에 걸쳐, 현재 전 우주의 생명체 수인 자연수 N과, Sonaht의 목표 범위인 자연수 A, B가 공백으로 구분되어 주어진다. ($2 \le N \le 1,000,000,\ 2 \le A \le B \le 100,000$)

출력 형식

매 줄마다, 각 테스트 케이스에서 Sonaht가 전 우주의 생명체 수를 목표범위 내의 소수로 만드는 데 필요한 최소한의 핑거 스냅의 횟수를 출력한다.

만약 목표범위 내의 소수로 만들 수 없다면, `-1`을 출력한다.

매 테스트 케이스는 독립적으로 고려되어야 한다.

예제

표준 입력	표준 출력
5 9 2 4 100000 605 610 300 14 16 7 7 10 98765 500 550	1 15 -1 0 12

설명

첫 번째 테스트 케이스에서는, 2번 동작 (전 우주의 생명체를 1/3로 줄임)을 하면 전 우주의 생명체의 수가 3이 되어, 2 이상 4 이하 소수가 된다. 그러므로 필요 횟수는 1이다.

세 번째 테스트 케이스에서는, 14 이상 16 이하 소수가 존재하지 않으므로, 아무리 핑거 스냅을 해도 목표를 달성할 수 없다. 따라서 -1을 출력한다.

네 번째 테스트 케이스에서는 핑거 스냅을 하지 않고도 목표가 달성되으므로 필요 횟수는 0이다.