행렬의 거듭제곱

시간 제한	0.5초
메모리 제한	512 MB
사용 가능 언어	C, C++
C++ 허용 STL	전체

C++의 경우 main 함수 내의 시작 지점에 다음 내용을 추가하여 cin, cout 의 입출력 속도를 개선할 수 있다.

ios_base::sync_with_stdio(false);

cin.tie(nullptr);

cout.tie(nullptr);

단, 위 내용을 추가할 경우 cin, cout 만 사용해야 하며, scanf, printf 등 C 입출력과 혼용해서 사용하면 안 된다.
C++ std::endl의 경우 출력 속도가 느리므로, cout<<endl; 대신 **cout<<"₩n";**을 사용하는 것을 권장한다.

음이 아닌 정수 N과 양의 정수 A, K에 대해서 $A^N \ mod \ K$을 구하는 방법을 생각해보자.
지수의 정의 대로 $A^N = \underbrace{A \times \cdots \times A}_{N개} \ mod \ K$ 로 계산할 수도 있으나, 다음과 같이 분할정복을 적용하면 더 빠른 계산이 가능하다.

$A^0 \ mod \ K = 1 \ mod \ K$

For $N \geq 1$, $A^N \ mod \ K = \begin{cases} A^{\frac{N}{2}} \times A^{\frac{N}{2}} \ mod \ K & \text{if } N \text{ is even} \\ A \times A^{N-1} \ mod \ K & \text{if } N \text{ is odd} \end{cases}$

재귀적으로 호출하는 N이 1 또는 0 이 될 때까지 해당 식을 반복하여 A^N을 구할 수 있다.

기존의 단순한 접근 방법이 $N-1$번의 곱셈을 통해서 문제를 해결했다면, 위의 분할정복 방법은 $O(\log N)$ 번의 곱셈을 통해 문제를 해결할 수 있다.

이 아이디어를 정수의 거듭제곱에서 2×2 행렬의 거듭제곱으로 확장해보자.

먼저 두 개의 2×2 행렬의 곱셈은 다음과 같이 정의된다.

$$\begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \times \begin{pmatrix} b_1 & b_2 \\ b_3 & b_4 \end{pmatrix} = \begin{pmatrix} a_1 b_1 + a_2 b_3 & a_1 b_2 + a_2 b_4 \\ a_3 b_1 + a_4 b_3 & a_3 b_2 + a_4 b_4 \end{pmatrix}$$

2×2행렬 M에 대해 다음과 같이 거듭제곱을 정의한다고 하자. 단, I는 2×2 항등행렬 $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$이다.

$$M^N = \begin{cases} I & \text{if } N = 0 \\ M \times M^{N-1} & \text{if } N \geq 1 \end{cases}$$

행렬 M과 거듭제곱할 횟수 N이 주어졌을 때, 위와 같이 분할정복을 적용하여 M^N을 구하는 프로그램을 작성하라

입력

첫 번째 줄에 테스트 케이스의 수 T $(1 \le T \le 100{,}000)$가 주어진다.

각 테스트 케이스는 다음과 같이 구성되어 있다.

- 첫 번째 줄에 행렬 M의 1행을 구성하는 두 원소가 공백으로 구분되어 주어지며, 두 번째 줄에 행렬 M의 2행을 구성하는 두 원소가 공백으로 구분되어 주어진다. 각 원소들은 0부터 1,000,000 사이의 값이다.
- 세 번째 줄에 행렬 M을 거듭제곱할 횟수 N $(1 < \mathrm{N} \le 1{,}000{,}000{,}000)$이 주어진다.

출력

각 테스트 케이스마다 M^N 의 각 원소들을 1,000,000,007 으로 나눈 나머지로 구성된 행렬을 두 줄에 걸쳐 출력한다. 첫 번째 줄은 해당 행렬의 1 행을 구성하는 두 원소를 공백으로 구분하여 출력하며, 두 번째 줄은 해당 행렬의 2 행을 구성하는 두 원소를 공백으로 구분하여 출력한다.

예제 입출력

예제 입력 1	예제 출력 1
2 3 1 4 2 31 2 0 0 1 1000000000	669976888 28621621 114486484 641355267 140625001 0 0 1