Kick Start 2017 - Round D

Sherlock and Matrix Game

Problem

Today, Sherlock and Watson attended a lecture in which they were introduced to matrices. Sherlock is one of those programmers who is not really interested in linear algebra, but he did come up with a problem involving matrices for Watson to solve.

Sherlock has given Watson two one-dimensional arrays A and B; both have length **N**. He has asked Watson to form a matrix with **N** rows and **N** columns, in which the j^{th} element in the i^{th} row is the product of the i-th element of A and the j-th element of B.

Let (x, y) denote the cell of the matrix in the x-th row (numbered starting from 0, starting from the top row) and the y-th column (numbered starting from 0, starting from the left column). Then a submatrix is defined by bottom-left and top-right cells `(a, b)` and `(c, d)` respectively, with a ≥ c and d ≥ b, and the submatrix consists of all cells (i, j) such that c ≤ i ≤ a and b ≤ j ≤ d. The sum of a submatrix is defined as sum of all of the cells of the submatrix.

To challenge Watson, Sherlock has given him an integer **K** and asked him to output the $\mathbf{K}^{th}$ largest sum among all submatrices in Watson's matrix, with **K** counting starting from 1 for the largest sum. (It is possible that different values of **K** may correspond to the same sum; that is, there may be multiple submatrices with the same sum.) Can you help Watson?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of one line with nine integers **N**, **K**, $\mathbf{A_1}$, $\mathbf{B_1}$, **C**, **D**, $\mathbf{E_1}$, $\mathbf{E_2}$ and **F**. **N** is the length of arrays A and B; **K** is the rank of the submatrix sum Watson has to output, $\mathbf{A_1}$ and $\mathbf{B_1}$ are the first elements of arrays A and B, respectively; and the other five values are parameters that you should use to generate the elements of the arrays, as follows:

First define $x_1 = \mathbf{A_1}$, $y_1 = \mathbf{B_1}$, $r_1 = 0$, $s_1 = 0$. Then, use the recurrences below to generate x_i and y_i for i = 2 to **N**:

- x_i = ($\mathbf{C}$*x_{i-1} + $\mathbf{D}$*y_{i-1} + $\mathbf{E_1}$) modulo **F**.
- y_i = ($\mathbf{D}$*x_{i-1} + $\mathbf{C}$*y_{i-1} + $\mathbf{E_2}$) modulo **F**.

Further, generate r_i and s_i for i = 2 to **N** using following recurrences:

- r_i = ($\mathbf{C}$*r_{i-1} + $\mathbf{D}$*s_{i-1} + $\mathbf{E_1}$) modulo 2.
- s_i = ($\mathbf{D}$*r_{i-1} + $\mathbf{C}$*s_{i-1} + $\mathbf{E_2}$) modulo 2.

We define $\mathbf{A_i} = (-1)^{r_i} * x_i$ and $\mathbf{B_i} = (-1)^{s_i} * y_i$, for all i = 2 to **N**.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is the $\mathbf{K^{th}}$ largest submatrix sum in the matrix defined in the statement.

Limits

1 ≤ **T** ≤ 20.
Memory limit: 1GB.
1 ≤ **K** ≤ min(10^5, total number of submatrices possible).
0 ≤ $\mathbf{A_1}$ ≤ 10^3.
0 ≤ $\mathbf{B_1}$ ≤ 10^3.
0 ≤ **C** ≤ 10^3.
0 ≤ **D** ≤ 10^3.
0 ≤ $\mathbf{E_1}$ ≤ 10^3.
0 ≤ $\mathbf{E_2}$ ≤ 10^3.
1 ≤ **F** ≤ 10^3.

Small dataset (Test set 1 - Visible)

Time limit: 40 seconds.
1 ≤ **N** ≤ 200.

Large dataset (Test set 2 - Hidden)

Time limit: 200 seconds.
1 ≤ **N** ≤ 10^5.

Sample

Sample Input

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

Sample Output

```
Case #1: 6
Case #2: 4
Case #3: 1
```

In case 1, using the generation method, the generated arrays A and B are [1, -3] and [1, -3], respectively. So, the matrix formed is
[1, -3]
[-3, 9]
All possible submatrix sums in decreasing order are [9, 6, 6, 4, 1, -2, -2, -3, -3]. As **K = 3**, answer is 6.

In case 2, using the generation method, the generated arrays A and B are [2] and [2], respectively. So, the matrix formed is
[4]
As **K = 1**, answer is 4.

In case 3, using the generation method, the generated arrays A and B are [1, 0] and [2, -1] respectively. So, the matrix formed is
[2, -1]
[0, 0]
All possible submatrix sums in decreasing order are [2, 2, 1, 1, 0, 0, 0, -1, -1]. As **K = 3**, answer is 1.