# Problem 2 - Survival Message

You are involved in a warfare, of course you are not a typical soldier, instead you are the squad programmer. Your forces were surrounded by enemies, but you are not discovered yet since you are in a deep cover inside a building. You have to send message to the headquarters, but it has to be special, should look more like an interference, because otherwise it will be detected by the enemy.

The message is formed by five elements **two numbers from [1…99] and three letters**. The elements should be in the following order:

* **Element 1**: **number from N to 99**
* **Element 2**: **uppercase letter from {firstUpper} to 'Z'**
* **Element 3**: **lowercase letter from** **{lowerLetter} to 'z'**
* **Element 4**: **uppercase letter from {secondUpper} to** **'**Z**'**
* **Element 5**: **number from M to 10**.

In order for the message to be readable by the allied forces **it has to** **match the following criteria:**

* **The first number** **should end in 2**
* **The second number should end in 5**
* **There will be total count of allowed valid messages {count}**

Your task is to generate all the valid messages and when **you reach the maximum number of messages {count}, you have to print on the console the last one which will be the survival message.**

## Input

The input consists of exactly **six lines:**

* **First line: N - number between** **[10...99]**
* **Second line**: **{firstUpper} -** **uppercase letter between [A…Z]**
* **Third line: {lowerLetter} - lowercase letter between [a…z]**
* **Fourth line**: **{secondUpper} - uppercase letter between [A…Z]**
* **Fifth line: M -** **number between** **[10...99]**
* **Sixth line: {count} - number between [1...100]**

## Output

Print on the console **the last message when you reach the maximum count of valid combinations.**

## Constraints

* There will be **no invalid input lines**.
* Allowed working **time** / **memory**: **100ms** / **16MB**.

## Example

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Обяснение** |
| 10  Y  y  Y  20  4 | 12YzZ15 | The first generated message is: 10YyY20, **which is invalid according to the validation criteria the** first number (10) doesn’t end in 2, also the second number (20) doesn’t end in 5.  The second message generated - 10YyY19 is again invalid, because the first number doesn’t end in 2 and the second number doesn’t end in 5.  We **count only the valid generated messages** so we continue to generate more combinations. We generate **the first valid** message: **12YyY15 – this time the first number (12) ends in 2 and the seocnd (15) ends in 5.**  So we increase the count by 1.  Then we generate **the second valid** message: 12YyZ15.  And **the** **third valid** message is: 12YzY15.  The **final valid message is the fourth:** 12YzZ15.  The total count of valid messages is **now 4**, which means we have reached the **maximum count (4)** of valid messages so we print the **fourth message:** **12YzZ15.**  The program stops. |
| 12  A  b  C  45  8 | 12AbD15 |  |
| 45  H  r  M  34  10 | 52HrQ15 |  |