In a parallel world, a human called Fibonacci was playing with his computer. Suddenly, he got a *BSoD* (The Blue Screen of Death). While his OS restarted, he realized that a group of immortal rabbits grows as shown: 0, 1, 1, 2, 3, 5, 8, 13, 21, ... , where each number is the number of pairs that were alive each month, and because of their immortality, the series could get really big in a short period of time. In other words, the number of pairs of those rabbits is the sum of the pairs of the previous two months. More formally:

$$f_n = \begin{cases} 0 & \text{if} \quad n = 0 \\ 1 & \text{if} \quad n = 1 \\ f_{n-1} + f_{n-2} & \text{if} \quad n \geq 1 \end{cases}$$

Trying to make a general statement, Fibonacci changed the root values (0 and 1) to variables $A$ and $B$, so $f_0 = A$ and $f_1 = B$.

When the computer was finally ready, and his IDE loaded, he decided to make a program with the cool Java's BigInteger-class to calculate how many rabbits will exist in the month $m$. After coding the algorithm he realized it wasn't as fast as he expected, so he decided to send telepathic messages to people from other dimensions asking for a fast algorithm of big numbers to solve this problem. Our problemsetters at RPC got that message so they want to commend you that **non-trivial** task!

## Input

The input starts with a line that contains an integer $T \leq 1000$ that represents the number of calculations that Fibonacci wants to do. The following $T$ lines will contain 3 integers $A$, $B$, $m$ ($0 \leq A, B \leq 1000000$, $0 \leq m \leq 100000$).

## Output

For each test case print the number of rabbits in the month $m$ with an end of line.

## Sample Input

```
1
100 100 50
```

## Sample Output

```
2036501107400
```