F - Pseudo-random Numbers

Input: `standard input`
Output: `standard output`

Access to high-quality randomness is very important for many applications, especially in cryptography. Radioactive decay is somtimes used as a source of true randomness, but this is a fairly slow procedure for getting random numbers. Also, in many applications it is important that the same random sequence can be produced in two different places. For these reasons one often uses a pseudo-random sequence instead. A pseudo-random sequence is a sequence that is, in fact, not random, but very hard to distinguish from a truly random sequence. A pseudo-random sequence should also be difficult to predict, i.e., given the first few elements of the sequence it should be difficult do determine some later, yet unseen, number in the sequence.

The Association of Cryptographic Machinery (ACM) has devised an algorithm for generating pseudo-random number sequences, but they have no idea how good it really is. Therefore they want you to test it.

The algorithm to generate a sequence of integers, where each integer is between 0 and $B - 1$ inclusive, is as follows:

- Start with any number (the seed) in base B. This number can contain hundreds of base B digits.
- The last digit (least significant) is output as the next element of the sequence.
- Create a new number by writing down the sum of all neighbouring digits from left to right. E.g., with $B = 10$, the number 845 would yield the number 129 (since $8 + 4 = 12$ and $4 + 5 = 9$).
- Repeat steps 2 and 3 as many times as needed, or until the number has only one base B digit. You get one more pseudo-random digit between 0 and $B - 1$ each time.

If we have $B = 10$ and the seed number is 845, then the next numbers will be 129, 311 ($1 + 2 = 3, 2 + 9 = 11$), 42 ($3 + 1 = 4, 1 + 1 = 2$), and 6 ($4 + 2 = 6$). As 6 is a single digit base 10 number, the algorithm terminates. The pseudo-random digits generated are $5, 9, 1, 2$ and 6.

You will be testing the generator as follows. You will be given the first L elements output by the generator and an integer $T > L$. You are supposed to decide if the first T elements are completely determined by the first L elements. To check the robustness of your testing procedure the ACM have slipped in some impossible sequences, i.e. sequences that cannot be generated by any initial seed.

Input

On the first line of the input is a single positive integer N, telling the number of test cases to follow. The first line of each test case consists of one integer B ($2 \leq B \leq 1000$), the base. The second line consists of an integer L ($1 \leq L \leq 100$), followed by the L first elements of some sequence (the elements are written in base 10 and are between 0 and $B - 1$ inclusive). The third line consists of an integer T, ($L < T \leq 100000$), the element of the sequence to predict.

Output

For each test case output, on a line of its own:

- impossible if no seed number can produce the given sequence.
- unpredictable if there exists a seed number that produces the given sequence but the first T elements are not completely determined by the first L elements.
- the T:th element of the sequence in base 10, otherwise.

Sample Input

```
3
10
5 5 9 6 7 0
7
16
4 11 7 8 4
12
2
5 0 1 1 1 0
10
```

Sample Output

```
8
unpredictable
impossible
```