This challenge is part of a tutorial track by MyCodeSchool

You're given the pointer to the head node of a linked list and a specific position. Counting backwards from the tail node of the linked list, get the value of the node at the given position. A position of 0 corresponds to the tail, 1 corresponds to the node before the tail and so on.

Input Format

You have to complete the `int getNode(SinglyLinkedListNode* head, int positionFromTail)` method which takes two arguments - the head of the linked list and the position of the node from the tail. positionFromTail will be at least 0 and less than the number of nodes in the list. You should NOT read any input from stdin/console.

The first line will contain an integer t, the number of test cases.
Each test case has the following format:
The first line contains an integer n, the number of elements in the linked list.
The next n lines contains, an element each denoting the element of the linked list.
The last line contains an integer $positionFromTail$ denoting the position from the tail, whose value needs to be found out and returned.

Constraints

- $1 \leq t \leq 10$
- $1 \leq n \leq 1000$
- $1 \leq list_i \leq 1000$, where $list_i$ is the i^{th} element of the linked list.
- $0 \leq positionFromTail < n$

Output Format

Find the node at the given position counting backwards from the tail. Then `return` the `data` contained in this node. Do NOT print anything to stdout/console.

The code in the editor handles output.
For each test case, print the value of the node, each in a new line.

Sample Input

```
2
1
1
0
3
3
2
1
2
```

Sample Output

```
1
3
```

Explanation

In first case, there is one element in linked list with value 1. Hence, last element is 1.

In second case, there are 3 elements with values 3, 2 and 1 (3 -> 2 -> 1). Hence, element with position of 2 from tail is 3.