Implement the class ProductOfNumbers that supports two methods:

1. add(int num)

* Adds the number num to the back of the current list of numbers.

2. getProduct(int k)

* Returns the product of the last k numbers in the current list.
* You can assume that always the current list has **at least** k numbers.

At any time, the product of any contiguous sequence of numbers will fit into a single 32-bit integer without overflowing.

**Example:**

**Input**

["ProductOfNumbers","add","add","add","add","add","getProduct","getProduct","getProduct","add","getProduct"]

[[],[3],[0],[2],[5],[4],[2],[3],[4],[8],[2]]

**Output**

[null,null,null,null,null,null,20,40,0,null,32]

**Explanation**

ProductOfNumbers productOfNumbers = new ProductOfNumbers();

productOfNumbers.add(3); // [3]

productOfNumbers.add(0); // [3,0]

productOfNumbers.add(2); // [3,0,2]

productOfNumbers.add(5); // [3,0,2,5]

productOfNumbers.add(4); // [3,0,2,5,4]

productOfNumbers.getProduct(2); // return 20. The product of the last 2 numbers is 5 \* 4 = 20

productOfNumbers.getProduct(3); // return 40. The product of the last 3 numbers is 2 \* 5 \* 4 = 40

productOfNumbers.getProduct(4); // return 0. The product of the last 4 numbers is 0 \* 2 \* 5 \* 4 = 0

productOfNumbers.add(8); // [3,0,2,5,4,8]

productOfNumbers.getProduct(2); // return 32. The product of the last 2 numbers is 4 \* 8 = 32

**Constraints:**

* There will be at most 40000 operations considering both add and getProduct.
* 0 <= num <= 100
* 1 <= k <= 40000