A mathematics professor in a Senior Secondary High School decided to evaluate students for the *Teachers Assessment* based on their problem-solving skills.

Given an array of integers *arr*, an integer, *sumVal*, the task is to pair the elements in *arr* into *interesting pairs*. Find the number of interesting pairs in the array. An unordered pair *(i, j)* is defined to be *interesting* if $|arr_i - arr_j| + |arr_i + arr_j| = sumVal$ (i.e., the sum of absolute difference and absolute sum at the values in respective indices is equal to *sumVal*). The goal is to find the number of interesting pairs in the array.

Example

- $arr = [1, 4, -1, 2]$
- $sumVal = 4$

Then, there are two interesting pairs, *(1, 4)* and *(3, 4)*. Because,

- $|arr_1 - arr_4| + |arr_1 + arr_4| = |1 - 2| + |1 + 2| = 4$.

pairs in the array.

Example

- *arr = [1, 4, -1, 2]*
- *sumVal = 4*

Then, there are two interesting pairs, *(1, 4)* and *(3, 4)*. Because,

- $|arr_1 - arr_4| + |arr_1 + arr_4| = |1 - 2| + |1 + 2| = 4$.
- $|arr_3 - arr_4| + |arr_3 + arr_4| = |-1 - 2| + |-1 + 2| = 4$.

Function Description

Complete the function *findInterestingPairs* in the editor below.

findInterestingPairs has the following parameters:

int arr[n]: an array of integers
int ***sumVal****: an integer*

Returns

int: an integer value denoting the number of interesting pairs

Constraints

int: an integer value denoting the number of interesting pairs

Constraints

- $1 \leq n \leq 10^5$
- $-10^6 \leq arr_i \leq 10^6$
- $1 \leq sumVal \leq 10^6$

▶ **Input Format For Custom Testing**

▼ **Sample Case 0**

Sample Input For Custom Testing

```
STDIN       FUNCTION
-----       --------
4       →   arr[] size n = 4
1       →   arr = [1, 3, 2, 0]
3
2
0
2       →   sumVal = 2
```

Sample Output

```
1
```

Explanation

There is only one interesting pair in the given array, i.e. (1, 4) since $|arr_1 - arr_4| + |arr_1 + arr_4| = |1 - 0| + |1 + 0| = 2$.

3. Question 3

Hackerbank allows all citizens of the city of Hackerland to maintain their finances.

In order to ensure security, given two integers n and k, a password is valid if:

- The length of the password is n.
- The password consists of lowercase English characters only.
- The password does not contain k consecutive equal characters.

Given the integers n and k, find the number of distinct valid passwords that can be generated. Since the answer can be large, compute it modulo $(10^9 + 7)$.

Example

Consider $n = 2, k = 2$.

The total number of passwords of length 2 is $26 * 26 = 676$. There are 26 cases where $k = 2$ consecutive characters are the same. Thus, the answer is $26 * 26 - 26 = 676 - 26 = 650$, and 650 modulo $(10^9 + 7) = 650$.

Example

Consider $n = 2, k = 2$.

The total number of passwords of length 2 is 26 * 26 = 676. There are 26 cases where $k = 2$ consecutive characters are the same. Thus, the answer is 26 * 26 - 26 = 676 - 26 = 650, and 650 modulo $(10^9 + 7) = 650$.

Function Description

Complete the function *countValidPasswords* in the editor below.

countValidPasswords has the following parameters:

int n: the length of the password

int k: the number of matching consecutive characters should be less than this number

Returns

int: the number of valid passwords, modulo $(10^9 + 7)$

Constraints

$2 \le n \le 10^5$

$2 \le k \le n$

▶ Input Format For Custom Testing

- $2 \leq n \leq 10^5$
- $2 \leq k \leq n$

▼ Input Format For Custom Testing

The first line contains an integer, n.
The second line contains an integer, k.

▼ Sample Case 0

Sample Input For Custom Testing

STDIN		FUNCTION
3	→	n = 3
3	→	k = 3

Sample Output

```
17550
```

Explanation

The number of passwords possible of length 3 are 26*26*26. Subtract the cases where all the characters are the same (26 such cases). The number of valid passwords is 26*26*26 - 26 = 17550 and 17550 % $(10^9 + 7)$ = 17550.

▶ Sample Case 1

1. Question 1

A popular social media platform provides a feature to connect people online. Connections are represented as an undirected graph where a user can see the profiles of those they are connected to.

There are *connection_nodes* users numbered 1 to *connection_nodes*, and *connection_edges* connections where the i^{th} pair connects nodes *connection_from[i]* and *connection_to[i]*. The *queries* array contains node numbers. Find the number of users whose profiles are visible to *query[i]*. Report an array of integers where the i^{th} value is the answer to the i^{th} query.

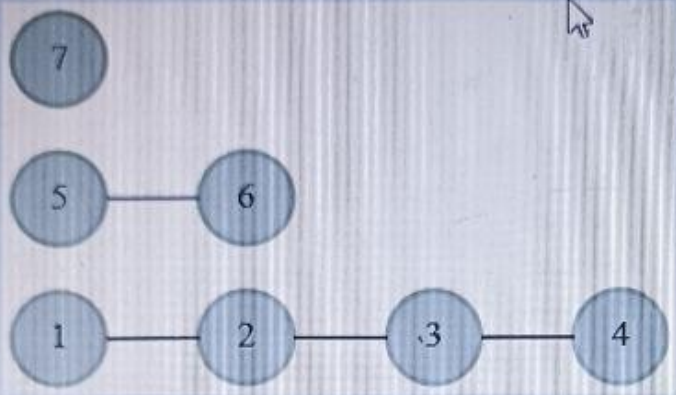
Example

connection_nodes = 7
connection_edges = 4
connection_from = [1, 2, 3, 5]
connection_to = [2, 3, 4, 6]
queries = [1, 3, 5, 7].

an array of integers where the i^{th} value is the answer to the i^{th} query.

Example

connection_nodes = 7
connection_edges = 4
connection_from = [1, 2, 3, 5]
connection_to = [2, 3, 4, 6]
queries = [1, 3, 5, 7].



Query	Visible Profiles	Number of Visible Profiles
1	[1, 2, 3, 4]	4
3	[1, 2, 3, 4]	4
5	[5, 6]	2
7	[7]	1

Return [4, 4, 2, 1].

Function Description

Complete the function *getVisibleProfilesCount* in the editor below.

getVisibleProfilesCount has the following parameter(s):

int connection_nodes: the number of users

int connection_from[connection_edges]: one user id of each connection

int connection_to[connection_edges]: the other user id of each connection

int queries[q]: the users to query

Returns

int[q]: the number of users whose profiles are visible to the queried users

Constraints

- $2 \leq connection\_nodes, connection\_edges \leq 10^5$
- $1 \leq connection\_from[i], connection\_to[i] \leq n$; $connection\_from[i] \neq connection\_to[i]$
- $1 \leq q \leq 10^5$
- $1 \leq queries[i] \leq connection\_nodes$

▶ Input Format for Custom Testing

▼ Sample Case 0

Sample Input 0

```
STDIN           FUNCTION
-----           --------
5               →   connection_nodes =
5
4               →   connection_from[]
size connection_edges = 4
2               →   connection_from =
[2, 2, 1, 1]
2
1
1
4               →   connection_to[]
size connection_edges = 4
1               →   connection_to = [1,
3, 3, 4]
3
3
4
3               →   queries[] size q = 3
4               →   queries[] = [4, 2,
5]
2
5
```

Sample Output 0

```
4
4
1
```

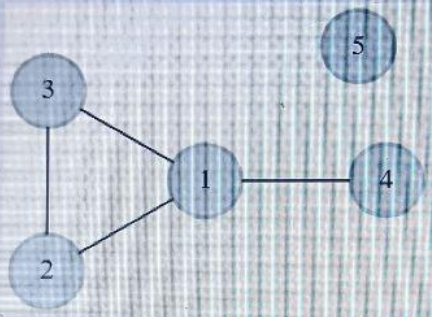
Explanation

```
1          →   connection_to = [1,
3, 3, 4]
3
3
4
3          →  queries[] size q = 3
4          →  queries[] = [4, 2,
5]
2
5
```

Sample Output 0

```
4
4
1
```

Explanation



Each of the users can see the profiles of 4 users except for 5 who can see only its own profile.

▶ **Sample Case 1**