A virtual memory management system in an operating system at Amazon can use LeastRecently-Used (LRU) cache. When a requested memory page is not in the cache and the cache is full, the page that was least-recently-used should be removed from the cache to make room for the requested page. If the cache is not full, the requested page can simply be added to the cache and considered the most-recently-used page in the cache. A given page should occur at most once in the cache.

Given the maximum size of the cache and a list of page requests, write an algorithm to calculate the number of cache misses. A cache miss occurs when a page is requested and isn't found in the cache.

Input

The input to the function/method consists of three arguments:

num, an integer representing the number of pages;

pages, a list of integers representing the pages requested;

maxCacheSize, an integer representing the size of the cache.

Output

Return an integer representing the number of cache misses.

Note

The cache is initially empty and the list contains pages numbered in the range 1-50. A page at index "i" in the list is requested before a page at index "i+1".

Constraints

0 <= i < num

Example

Input:

num = 6

pages = [1,2,1,3,1,2]

maxCacheSize = 2

Output:

4

Explanation:

Cache state as requests come in ordered longest-time-in-cache to shortest-time-in-cache:

1*

1,2*

2,1

1,3*

3,1

1,2*

Asterisk (*) represents a cache miss. Hence, the total number of a cache miss is 4.

Signature

```
int lruCacheMisses(int num, List<Integer> pages, int maxCacheSize) {
}
```