Cow Proximity

(2 points.) To start, run `python new-homework.py [language] cow-proximity`

Background. This is an irreverent example of a problem that's easy to solve inefficiently but hard to solve efficiently. Before writing any code, think about the data structures we've discussed in class, like queues, stacks, and hash tables, and consider which support the needed operations in constant time. You can combine several data structures, and you can use any data structures provided by your programming language, such as Python's `collections` module or C++'s STL. (Not all of these structures are available in base R, so you may have to go spelunking in CRAN. The `dequer` and `hash` packages may be useful.)

Task. Suppose that cows of the same breed get into an argument with each other if they are standing too close. Two cows of the same breed are "crowded" if their positions within a line of N cows differ by no more than K, where $1 \leq K < N$.

Given an array representing the breed IDs of a line of cows, compute the maximum breed ID of a pair of crowded cows. If there are no crowded cows, return -1.

Choose algorithms and data structures which do not require repeated scans through the list of cows: you should not, for example, loop through all cows, and for each cow loop through all others to detect if they are of the same breed. This problem can, in fact, be solved with a *single* pass through all cows.

Example. If we write a function *`crowded_cows`*`(`*`cow_list`*`, `*`K`*`)`, we have

```
crowded_cows([7, 3, 4, 2, 3, 4], 3) == 4
crowded_cows([7, 3, 4, 2, 3, 10, 4], 3) == 3
crowded_cows([7, 3, 1, 0, 4, 2, 16, 28, 3, 4], 3) == -1
```

Requirements.

☐ Write a function *`crowded_cows`*`(`*`cow_list`*`, `*`K`*`)` that returns the maximum breed ID of a pair of crowded cows in the supplied list.

☐ Write unit tests that comprehensively check the correctness of *`crowded_cows`*.

☐ Provide a command-line driver script which takes a list of cows on standard input and *`K`* as an argument, then prints the result of *`crowded_cows`*, so that you can run e.g. `Rscript crowded_cows.R 50 < cows.txt` and get the answer as output. See `https://36-750.github.io/tools/writing-commands/` for tips.

☐ Test your code on the provided file of 50,000 cows, `cow-proximity/Data/cows.txt`, using $K = 25000$. Ensure you get the correct answer, 503,739. The answer for `cows-2.txt`, also provided, is 364. Optimize your code and data structure choice so it runs for either example in under five seconds. (With the right data structures, it can run in under one second.)