Exercise 5

Again, the focus is on unit testing, thus: test your solutions thoroughly.

Part 1

Determine whether an integer is a palindrome. Do this without allocating extra (non-constant) memory, i.e. complete the following signature function:

```
bool isPalindrome(int x) {
  // TODO
}
```

Part 2

Given an array of citations (each citation is a non-negative integer) of a researcher, write a function to compute the researcher's h-index.

According to the definition of h-index on Wikipedia: "A scientist has index h if h of his/her N papers have at least h citations each, and the other N − h papers have no more than h citations each."

For example, given `citations = [3, 0, 6, 1, 5]`, which means the researcher has 5 papers in total and each of them had received 3, 0, 6, 1, 5 citations respectively. Since the researcher has 3 papers with at least 3 citations each and the remaining two with no more than 3 citations each, his h-index is 3.

Note: If there are several possible values for h, the maximum one is taken as the h-index.

Basically, complete the following signature:

```
int hIndex(vector<int>& citations) {
  // TODO
}
```

Part 3

What if the citations array from Part 2 is sorted in ascending order? Could you optimize your algorithm?

Part 4

Given an array nums, there is a sliding window of size k which is moving from the very left of the array to the very right. You can only see the k numbers in the window. Each time the sliding window moves right by one position.

For example,
Given nums = [1,3,-1,-3,5,3,6,7], and k = 3.

Window position	Max
[1 3 -1] -3 5 3 6 7	3
1 [3 -1 -3] 5 3 6 7	3
1 3 [-1 -3 5] 3 6 7	5
1 3 -1 [-3 5 3] 6 7	5
1 3 -1 -3 [5 3 6] 7	6
1 3 -1 -3 5 [3 6 7]	7

Therefore, return the max sliding window as [3,3,5,5,6,7].

Note:
You may assume k is always valid, ie: 1 ≤ k ≤ input array's size for non-empty array.
Basically, complete the following function signature:

```
vector<int> maxSlidingWindow(vector<int>& nums, int k) {
   // TODO
}
```

Follow up:
Could you solve it in linear time?