All codes must be commented and justified
You get points if your solution is correct
In order to have all the points, you must respect the complexity specifications

1) A word is a *power* if it can be generated by repeating at least twice the same sub-word. Example: "`aaaa`" (word "`a`" repeated 4 times) and "`abcabcabc`" (word "`abc`" repeated 3 times) are powers. But "`abcd`" and "`abac`" are not.

   Write a program **`bool`** `isPow(`**`const`** `string& s)` that asserts whether the input word s is a power. *Hint: keep track of the occurences of the first letter.*
   Complexity: O(n*log(n)). **/5**

   If you are not able to solve the above exercise, then you may solve one of the following easier variants (but doing so, you do not get all the points):

   a. Write a program **`bool`** `isPow(`**`const`** `string& s,` **`int`** `k)` that asserts whether the input word s can be generated by repeating *exactly* k times the same sub-word.
   Complexity: O(n) **/2**

   b. Write a program `isPow2(`**`const`** `string& s)` that asserts whether the input s can be generated by repeating at least twice the same sub-word w *and* there is no repeated letter in w.
   Example: "abcabc" can be generated by repeating twice the word abc, and all the letter in abc are different. However, "abaaba" can be generated by repeating twice the word "aba", but here letter a is repeated twice.
   Complexity: O(n) **/3**

2) Implement a data structure supporting the following operations:
   a. **`bool`** `empty()`: returns 1 if there is no element stored in the structure
   b. **`void`** `add(`**`int`** `v)`: adds a new element equal to v in the structure
   c. **`int`** `next()`: returns and removes the k^th oldest element in the structure, if it exists (otherwise, returns and removes the most recent element added to the structure). Here, k is a fixed parameter which is set only once at the initialization.

   All the operations should run in O(1) time. **/5**