# Coding Challenge 3: C++ Lookup Cache

You are working with a program written in modern C++ that takes network packets as input data. Each network packet is considered to be part of a network connection, defined as having two pairs of [IP address, port number]. The order of those IP and port pairs does not matter.

Write a ‘cache’-style container class that holds a defined, limited number of the most frequently seen network connections.

Write and return a header file and an implementation file, they should be as presentable as you would to deliver to a client. Include at least one unit-test function.

During the follow-up interview we will discuss both your implementation and various possible extensions to the requirements or uses of the class.  
  
Functionality

- ‘insert’ function that takes as input a pair of [address, port] combinations and checks if such combination exists and if not, adds it to the cache. Insert function should return a pair of values: whether the connection was newly added or already existed in the cache and how many times it was looked-up so far. For newly added connections the number of lookups should be 1, for already existing connections the (number of previous lookups + 1) i.e. adding something that already exists also counts as a lookup. For now, let's say that the insert function will be inserting a maximum of size\_of\_cache new items i.e. the cache will not be given more connections than its fixed size can hold.  
  
- ‘lookup’ function that returns true/false and the number of successful lookups made for that connection including this one. The cache should be oriented towards lookup performance rather than insert performance - lookup has to be faster, we do not mind so much how long it takes to actually insert a connection if necessary.   
  
- ‘dump’ function that dumps the cache content to stdout (for simplicity). The output must be ordered by the frequency of lookups.  
  
Definitions  
  
- A port is a 16-bit unsigned integer  
  
- IP address should be presented in its human readable form as a four 1-byte numbers (e.g. “192.168.0.1”). It can be stored in any form you wish, but any comparisons (e.g. IP1 < IP2) must use a custom function overloading the operator. Note that most STL containers use comparison operators.  
  
- “modern” C++ should utilise at least some of the core features and conventions added in C++11/14/17 along with the STL.   
  
Output Example

add\_cache (IP1, port1, IP2, port2) -> “new”, 1  
lookup\_cache(IP3, port3, IP1, port1) -> F, 0  
add\_cache (IP1, port1, IP3, port3) -> “new”, 1  
lookup\_cache(IP2, port2, IP1, port1) -> T, 2  
add\_cache (IP2, port2, IP1, port1) -> “existing”, 3  
add\_cache (IP1, port1, IP2, port2) -> “existing”, 4  
lookup\_cache(IP1, port1, IP2, port2) -> T, 5  
lookup\_cache(IP1, port1, IP3, port3) -> T, 2  
lookup\_cache(IP1, port1, IP4, port4) -> F, 0  
  
dump() ->  
IP1,port1 <-> IP2,port2:  5  
IP1,port1 <-> IP3,port3:  2