CS 240 Coding Assignment 4
Double Indexed String Heap (DISH)

For this assignment, you will implement a container of strings, which you will "index" in two ways, namely alphabetically, and by string length. We will call this a Doubly Indexed String Heap (DISH). Begin by implementing a Dish C++ class that contains a vector or a dynamically allocated array (your choice) of strings, along with the following operations:

int Dish::insert(string s)	Inserts string s into the Dish. If you use an array, you may assume a maximum of 1024 strings, or implement your dynamic array to expand automatically, like vector does. The insert() function should just place the string in the next available location within the list (i.e. array or vector), and return the index corresponding to where the value was added. *Once strings are placed in their location in this list, they will not be moved or removed.* Returns the index of that string's location in the array on success, or -1 if the list has no more space to add s.
int Dish::find(string s)	Returns the index of string s in the Dish, which also corresponds to its insertion order. The first string inserted will be string 0 (and so this function would return 0), the 2nd will be string 1, etc. Returns the index on success, or -1 if the string is not found in the Dish. Works in O(N) time.
bool Dish::capitalize(int k)	Capitalizes the first letter of the k^{th} string added to the Dish Works in O(log N) time. Returns true if k < N, false if k >= N, where N is the number of strings in the Dish.
bool Dish::allcaps(int k)	Capitalizes all of the letters of the k^{th} string added to the Dish. Works in O(log N) time. Returns true if k < N, false if k >= N, where N is the number of strings in the Dish.
bool Dish::truncate(int k, int i)	Chops all but the first i characters of the k^{th} string added to the Dish. If the string contains fewer than i characters, the truncate() function leaves it alone. Works in O(log N) time. Returns true if k < N, false if k >= N, where N is the number of strings in the Dish.
string Dish::getshortest()	Returns the shortest string in the Dish, in O(1) time. Does not remove the string from the Dish.
string Dish::getfirst()	Returns the alphabetically first string in the Dish, in O(1) time. Does not remove the string from the Dish.

Note that without the final two functions (getshortest() and getfirst()) you could support capitalize(), allcaps(), and truncate() in O(1) time. You may begin implementing your data structure by supporting these operations first, in the straightforward way. Doing so will not set you back when you move on to also support getshortest() and getfirst().

To support the final two operations in the runtimes specified, you will need to maintain two different heaps, and these heaps will increase the runtime complexity of the capitalize(), allcaps(), and truncate() functions. You should *not* keep copies of the strings in each heap. Instead your two heaps should contain indices into the one array/vector that stores the strings. The two heaps should be updated appropriately, in the runtimes specified, using straightforward adaptations of the heap operations we have studied in class. Decide whether to use min-heaps, max-heaps, or one of each.

Take some time to design your data structure and convince yourself that you can do everything in the runtimes requested. We will discuss in class on Wednesday November 15th.