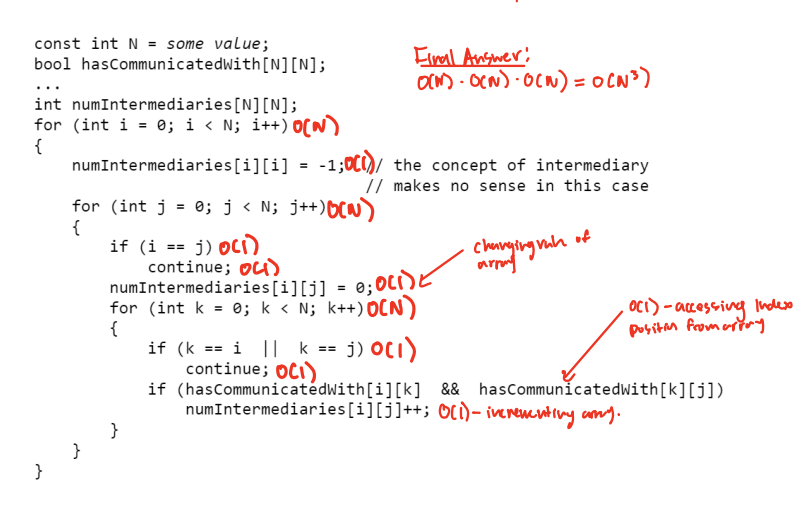
1e. During execution of test case 3, the push back operation and expansion of the capacity reallocates new memory for the resizing of the array. The pointers in the loop may not be referring to the right place because memory was deleted and reallocated to expand the vector.

3. The one-argument form of Sequence<Coord>::insert causes an error because, when the insert function is called with one parameter, the function has a greater than sign that compares value to a Node pointer to m\_value. Because Coord’s assignment operator was not defined, the assignment operator call of type Coord causes an error within the function.

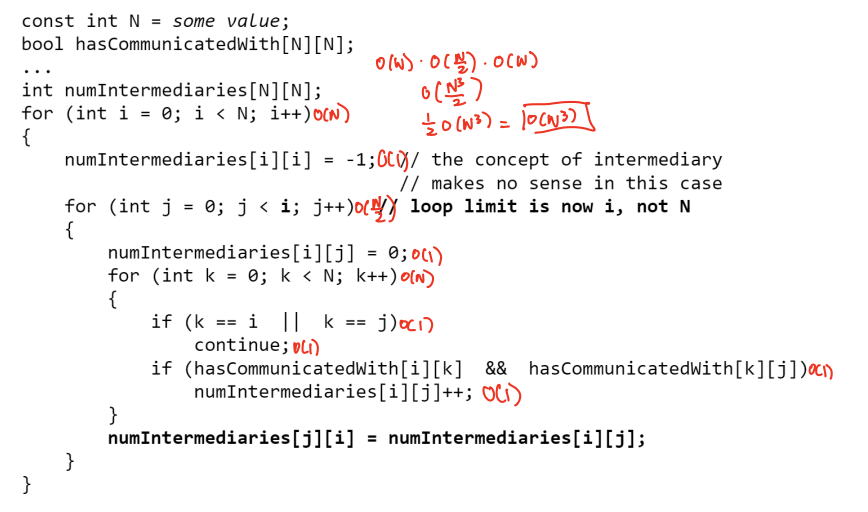
4 b. You can’t implement it as a one parameter listAll because you have to traverse a path and concatenate the string as you traverse the different object’s vectors. You need to pass in the path in order for the function to keep track of the path it just took and print the file path once the base case of an empty string gets encountered by the recursive call.

5 a. **The time complexity of this algorithm is O(N^3).** There are a total of three loops, all of which are order N because the loop starts from 0 and ends at N, while incrementing the iteration by each time. The lines of code within the loops are all O(1) because they are either comparing values, accessing an element in an array, or setting an element of an array to something.

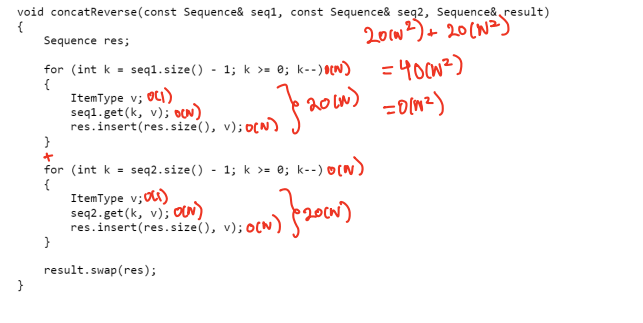


5 b. **The time complexity of this algorithm is O(N^3).** The most inner loop and most outer loop is O(N). The middle loop is O(N/2) because the iteration is from 0 to i, which is just half of the outer loop. Although this is technically faster because of the smaller constant of proportionality, in Time Complexity terms, we neglect the constant of proportionality and only look at the growth pattern.

Therefore, ignoring the constant of proportionality, the time complexity of the algorithm becomes O(N^3).



6 a. **The time complexity of concatReverse is O(N^2).** Since the get function is O(N) and the insert function is O(N), we get 2O(N) if we add the two functions. Yet, because it is in a loop,it is 2O(N^2). Because there are two separate loops, the loop for sequence 2, which is O(N^2), gets added to the loop for sequence 1, which is also O(N^2). If we add both orders, we get 2O(N^2). Ignoring the constant of proportionality and just paying attention to the growth pattern, the time complexity becomes O(N^2).



6 b. **The time complexity of this function is less, with a time complexity of O(N).** The call to insertBefore function produces a time complexity of O(1). Since the call is within a loop, the loop produces O(N). There are 2 loops, but not within each other, therefore, we add them to produce 2O(N), which is just O(N) in time complexity terms. The call to swap function produces O(1) time complexity. Finally, the sequence destructor gets called, which is O(N) because it contains a while loop. Combining all of the Big-O Notations, we get 3O(N). However, because we ignore the constant of proportionality this function just has a time complexity of O(N).

