This Document explains why we chose to make each test case what it is

1) The sample
   It's the provided sample.
2) Perfect Test
   The sample uses the perfect command, but it has no effect on the answer. This test case ensures that it works properly.
3) Imbalance Query
   All of the queries in this test should require recursion down multiple paths to ensure the solution properly adds ranges that don't hit the node exactly.
4) Oops all swaps
   Simply using the provided update function and calling it twice will not work. This is because the update calls will add two separate versions to the persistent segment tree. This is an error as only one new version should be added and ensures that the solution does this correctly.
5) Replace then perfect
   Every desk will be replaced with an inferior employee evaluation score. Then, perfect will be called on each desk. This is to ensure that the solution properly updates the perfect tracker when replacing.
6) Swap then perfect
   All desks will be swapped with at least one other desk. Then, perfect will be called on each desk. This is to ensure that the solution properly updates both perfect trackers when swapping.
7) Already Perfect
   The perfect command can be called on a desk that already has the best employee at it. This ensures that this is handled correctly.
8) A load of nothing
   This test case never has a query command. It is mostly to mess with the coder.
9) Long Long
   The maximum value of each node is 10^9, and with a maximum of 10^5 employees this will overflow an int. This makes sure the solution uses long longs when appropriate to avoid this.
10) Enough Versions
   The provided persistent segment tree implementation has a maximum of 100 versions. This ensures that the solution either uses a vector or has room for 10^5 versions.

Samples 11-20 are larger test cases for time complexity