* 5 data structures that will be tested
  + Vector
  + List
  + Binary search tree (set, map)
  + Priority queue
  + Hash table (unordered\_set, unordered\_map)
* 6 operations using integers or STL strings on the five data structures
  + Sort (use the default operator< for the specific data type)
  + Remove duplicates from a sequence without otherwise changing the overall order (keeping the first occurrence of the element)
  + Determine the mode (most frequently occurring element) 🡪 if there is a tie, you may return any of the most frequently returning elements
  + Identify the closest pair of items within the dataset (integer data only) 🡪 use operator- to measure the distance; if there is a tie in distance, you may return any of the pairs of closest distance
  + Output the first/smallest f items (a portion of the complete sorted output)
  + Determine the longest matching substring between any two elements (STL string data only) 🡪 if the input contains the words ‘antelope’, ‘buffalo’, and ‘elephant’, the longest substring is ‘ant’ (found within both ‘antelope’ and ‘elephant’) - if there is a tie, you may return any of the longest maching substrings
* See also the provided sample output for each of these operations
* Rules for comparison
  + We will analyze the cost of a program/function that reads the input from an STL input stream object (cin or ifstream) and writes the answer to an output stream (cout or ofstream) 🡪 the function should read through the input only once and construct and use a single instance of the specified STL data structure to complete the output
  + The function may not use any other data structure to help with the computation (e.g. storing data in a C-style array)
* Make your predictions and determine which data structures can be used for which operations
* Focus on the first three operations from the table first (sort, remove\_duplicates, and mode) 🡪 once debugged, tested, and relative performance analyzed, you can implement the other operations
* Estimate memory usage locally using the commands below:

clang runstats.c -o runstats.out

./runstats.out ./perf.out vector sort string < medium\_string\_input.txt > my\_out.txt

* Estimate dominant coefficients as well