1.

Filename	Avg. Queries for hashQuestLog	Avg. Queries for treeQuestLog
quests4.csv	1.25	2.0
quests16.csv	1.13	4.0
quests250.csv	1.23	13.34
quests1000.csv	2.15	18.08
quests100000.csv	1.45	30.88

2. If you had to choose a simple function (i.e. from the list of functions used in big-O notation) to characterize the behaviour of the average number of items examined per query for the hashed quest log as the number of quests (n) in the log increases, what would it be?

The average number of items examined per query for the hashed quest log can be characterized by O(1).

3. If you had to choose a simple function (i.e. from the list of functions used in big-O notation) to characterize the behaviour of the the average number of items examined per query for the tree quest log as the number of quests (n) in the log increases, what would it be?

The average number of items examined per query for the tree quest log can be characterized by O(log n).

4. If your primary use of the quest log was to display all of the quests in the log in alphabetical order, would you prefer the hashed quest log or the tree quest log? Why?

The tree quest log would be preferred if the primary use of the quest log is to display all quests in alphabetical order. This is because a binary search tree naturally supports in-order traversal, which efficiently provides elements in sorted order with a time complexity of O(n). In contrast, a hash table does not inherently support ordering, which requires an additional sorting step with an O(n log n) addition.

5. If your primary use of the quest log was to periodically look up the details of specific quests in no particular order, would you prefer the hashed quest log or the tree quest log? Why?

We would prefer the hashed quest log because lookups on average take constant O(1) time, whereas lookups in a binary search tree typically take O(\log n) time, which is less efficient than the hash table for random-access queries.