Evaluation and Recommendation:

The primary benefit of using linked lists is low implementation complexity. While the constant speed of inserting to linked lists and moderately low speed of searches make this a good choice for simple programs, there are other choices which can increase the speed of searches at the cost of implementation complexity. These include hash tables and binary search trees. A binary search tree is more complex to implement than a linked list, but each insertion to a BST includes sorting the tree and placing the node in its proper location. As a result, outputting a sorted binary search tree is much simpler than sorting and outputting other data structures. This is a primary benefit of using a BST for the course program. A hash table, however, has other advantages over linked lists that a tree cannot provide. A clear example of this is this speed of insertions and searches, which are O(1) on average. This is due to the structure of a hash table, which uses keys-value pairs to store data in separated locations in memory. Given a key, locating a value in a hash table is a very simple procedure. Still, the complexity of implementing a hash table is higher than of other data structures, and is not entirely necessary where speed is not a high priority. Given these options, I would recommend a binary search tree for the structure to be used in the course program as it allows for fast inserting and searching (the two primary functions of the program) with a relatively simple implementation.