**Runtime Complexity Analysis**

In my analysis of vectors, hash tables, and binary search trees, I learned that a vectors or dynamic arrays allow their contents to be accessed via an index number. This allows for you to easily access any element by specifying the index number of the element. Vectors have a linear average runtime complexity for searching, inserting, and deleting elements. Hash tables are unsorted and therefore the ability to sort courses alphabetically is not something that can be easily implemented in this data structure. You cannot access hash table elements by using an index. A hash table does allow for quicker searches, insertions, and deletions, with an average runtime complexity of O(1). Binary search trees have a runtime complexity for accessing elements, searching elements, inserting elements, and deleting elements of O(log(N)). Binary search trees do not allow access to elements via an index.  
 While each data structure has its own advantages and disadvantages, I feel that a vector would be the best fit for the course application. One reason I can defend choosing a vector despite a higher runtime complexity as the input size grows, this part of the analysis is focused only on worst case scenario. Since the number of courses is limited and most courses do not have a long list of prerequisites, there are other factors to consider when selecting a data structure. While hash tables and binary search trees offer some benefits in inserting, deleting, and searching for elements, using a vector will result in less lines of code that are easier to understand and maintain. There will not be a noticeable runtime difference between the three structures in this scenario because there is not enough data to take advantage of the benefits of a more complex data structure. A hash table is an unordered data structure. This makes a hash table a poor choice for the course data application as one of the functional requirements is the ability to print the courses in order. While it is possible to print the contents of a hash table in order, this would be additional code needed outside of the hash table to organize the data prior to printing it to the screen. This additional work would negate any benefit gained by using a hash table. The code to load data from file and error check the file has the same complexity since the code essentially runs the same checks on the file no matter which structure is chosen.