There needs to be a runtime for each of the three data structures: vector, hash table, and tree.

**Vector:**

1. Open the file and read the data:

OPEN file “course\_data.txt”

FOR each line in fine

Parse line into course details.

IF line has formatting errors

PRINT “Errors in line”

ELSE

CREATE Course object with parsed details

ADD Course object to Vector

CLOSE FILE

1. Create course objects:

Function createCourseObject(line)

SPLIT line into fields (ID, Title, Prerequisites)

Create new course objects with ID, Title, Prerequisites

RETURN course object

1. Print course information:

FUNCTION printCourseInfo(courseID)

Search Vector for course object with courseID

IF course object found

PRINT title and prerequisites

ELSE

PRINT “Course not found”

1. Menu pseudocode

DISPLAY “1: Load File”

DISPLAY “2: Print Course List”

DISPLAY “3: Print Course Information”

DISPLAY "9: Exit”

GET userInput

SWITCH userInput

CASE 1:

CALL loadFileToVector()

CASE 2:

CALL printSortedCourses()

CASE 3:

GET courseID

CALL printCourseInfo(courseID)

CASE 9:

EXIT

1. Print alphanumeric course list:

FUNCTION printSortedCourses()

SORT Vector by courseID

FOR each course object in Vector

PRINT course object details

**Runtime analysis.**

Loading the data O(n) where n is the number of courses. Each course is processed once and added to the vector.

Printing sorted courses. O(n log n) for sorting and. O(n) for print.

Print course info. O(n) in the worse case if linear searched was used.

**Hash Table:**

Step 1-4 are the same as vectors

1. Print alphanumeric course list:

FUNCTION printSortedCourses()

EXTRACT all keys from Hash Table

SORT keys alphanumerically

FOR each key in sorted list

RETRIEVE course object from Hash Table using key

PRINT course object details

**Runtime analysis.**

Loading data. O(n) where n is the number of courses. Insertion into has table is average O(1) per course.

Print sorted courses. O(n) for extracting Keys, O(n log n) for sorting, and O(n) for printing.

Print course info. O(1) average time for retrieving a course.

**Binary Search TREE(BST):**

Step 1-4 are the same as vectors

1. Print alphanumeric course list:

3

FUNCTION printSortedCourses()

INORDER travers BST

FOR each course object in traversal

PRINT course object details

**Runtime analysis.**

Lading data. O(n log n\_ for insertion assuming a balanced tree.

Print sorted courses. O(n) for in-order traversal and O(n) for printing.

Print course info. O(log n) average time for searching.

I would recommend a hash table for this application. It provides the fastest retrieval of course information but it does require additional sorting logic to print the list. If we want to keep all the information always sorted, a binary search tree would be the ideal choice.