Struct Course

courseNumber

courseTitle

list of prerequisites

Fn insertCourse

If tree is empty

Add course as root

Else

Compare course number with current node

If course number is smaller, go left

If course number is bigger, go right

Insert the course

Fn findCourse

Start at root

While not at the end of the tree

If course number matches, return the course

If course number is smaller, go left

If course number is bigger, go right

If not found, return nothing

Fn loadFromFile

Open the file

If the file can't open

Show an error message

Return

Make a list to hold course information

Make a list to hold all course numbers

For each line in the file

Split the line into pieces

If there are less than 2 pieces

Show an error message

Continue to next line

Save the course number, title, and any prerequisites

Add the course number to the list of course numbers

Add this course information to the course list

For each course in the course list

For each prerequisite in that course

If the prerequisite is not in the list of course numbers

Show an error for missing prerequisite

Stop

Create an empty tree to store courses

For each course in the course list

Create a new course object (with number, title, and prerequisites)

Add it into the tree using the insertCourse function

Return the tree of courses

Fn searchCourse

course = findCourse

If course is not found

Show "Course not found"

Stop

Show course number and title

If course has prerequisites

Show "Prerequisites"

For each prereq in the list

Find the course in the tree

If found, show course number and title

If not found, show an error that the course is missing

Else

Show "No prerequisites"

Fn printCoursesInOrder

If node is empty, return

Go left

Show course number and title

Go right

Fn showMenu

Set root to empty

Set dataLoaded to false

While true

Show "1. Load course data"

Show "2. Print all courses"

Show "3. Print course info"

Show "9. Exit"

Read userInput

If userInput is 1

Ask for file name

root = loadFromFile

If root is not empty

Set dataLoaded to true

Show "Course data loaded"

Else

Show "Failed to load data"

Else if userInput is 2

If data is not loaded

Show "Please load course data first"

Else

Call printCoursesInOrder

Else if userInput is 3

If data is not loaded

Show "Please load course data first"

Else

While true

Ask for course number or 'back' to return

If input is 'back'

Break

Call searchCourse

Else if userInput is 9

Show "Goodbye"

Stop loop

Else

Show "Invalid option"

My recommendation for the best data structure is the binary search tree, or BST. It offers a strong balance between efficient searching and maintaining a sorted order. This makes it ideal for the course advising system, where users must search for specific courses quickly and view an alphanumerical list of all courses. BST allows both operations to be performed efficiently.

While hash tables are faster for direct lookups with average-case constant time complexity, they do not maintain any order. This means additional work is required if a sorted list of courses is needed, a core requirement for the advising program. Although rare, hash tables can also suffer from collisions, negatively impacting performance in worst-case scenarios.

Vectors are simple to implement and great for basic storage and sequential access. However, they are the least efficient for search operations, as they require linear time to locate a course. Sorting the vector to list courses alphabetically adds extra complexity, especially as the dataset grows.

On the other hand, the BST maintains sorted order naturally through in-order traversal and offers average-case logarithmic performance for insertion and search operations. This makes it the most effective and scalable structure to support course management and user queries in this advising system. Therefore, I strongly recommend using a binary search tree as the primary data structure for this application.

A screenshot of a computer

AI-generated content may be incorrect.  
n= courses

p = num of prereqs

m= prereqs for specific course