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"

The overall time complexity for loading courses into a binary search tree (BST) is **O(n log n)** on average, where *n* is the number of courses. Each course is inserted into the tree in **O(log n)** time, assuming the tree stays balanced. Reading and parsing the file takes **O(n)**, and checking prerequisites against a set adds **O(p)**, where *p* is the total number of prerequisites across all courses.

Searching for a course in the tree is much faster than in a vector, taking **O(log n)** time instead of **O(n)**. Once the course is found, printing it takes **O(1)** time. If the course has prerequisites, each one must be searched in the tree, which takes **O(log n)** per prerequisite. So, if a course has several prerequisites, printing them all takes more time depending on how many there are.

Printing all courses in order is done using an in-order traversal, which takes **O(n)** time.

In summary, the binary search tree is efficient for both loading and searching. It also keeps the courses sorted automatically, which makes it a strong choice for this course advising system.