Structure Course

courseNumber String

courseTitle String

prereq List of Strings

Fn loadCourses

Create empty hash table courseTable

Open the file

For each line in the file

Split the line into tokens by commas

If there are fewer than 2 tokens, skip the line

courseNumber = tokens 0

courseTitle = tokens 1

prereq = tokens from 2 onward

Create a new Course object and store it in courseTable

courseTable courseNumber = new Course(courseNumber, courseTitle, prereq)

For each course in courseTable

For each prereq in course prereq

If prereq not in courseTable

Print error Prerequisite not found for prereq

Return courseTable

Fn printCourseInfo courseTable courseNumber

If courseNumber not in courseTable

Print Course not found

Return

course = courseTable courseNumber

Print Course Number course courseNumber

Print Course Title course courseTitle

If course has prereq

Print Prerequisites

For each prereq in course prereq

If prereq in courseTable

Print prereq courseTable prereq courseTitle

Else

Print Missing info for prerequisite prereq

Else

Print No prerequisites

Fn printAllCourses courseTable

Get all course numbers from courseTable and sort them

For each courseNumber in sorted list

course = courseTable courseNumber

Print courseNumber + ": " + course courseTitle

Fn showMenu

courseTable = empty

dataLoaded = false

While true

Print "1. Load course data"

Print "2. Print all courses"

Print "3. Print course info"

Print "9. Exit"

Read userInput

If userInput is 1

Ask for file name

courseTable = loadCourses

If courseTable is not empty

dataLoaded = true

Print Course data loaded

Else

Print Failed to load data

If userInput is 2

If dataLoaded is false

Print Please load course data first

Else

Call printAllCourses courseTable

If userInput is 3

If dataLoaded is false

Print Please load course data first

Else

While true

Ask for course number or type 'back' to return

Read courseNumber

If courseNumber is 'back'

Break

Call printCourseInfo courseTable courseNumber

If userInput is 9

Print Goodbye

Break

The time complexity for loading courses into a hash table is O(n + p), where n is the number of courses and p is the total number of prerequisites across all courses. Reading and parsing the file takes O(n) time, assuming one line per course. Inserting each course into the hash table takes O(1) time on average, so inserting all courses takes O(n) overall. After all the courses are added, the program checks that each prerequisite is valid by looking it up in the hash table. Since lookups are also O(1) on average, this validation step takes O(p) time.

When searching for a specific course by course number, the hash table provides O(1) access time, making course lookups very fast. Printing the course title takes constant time, and printing its prerequisites involves going through the list and checking each one in the table. If a course has, for example, three prerequisites, the total time to print them would be O(3), or more generally O(number of prerequisites).

Overall, the hash table is very efficient for both loading and searching. It allows fast access to any course and makes it easy to check prerequisites quickly. In rare cases, performance could slow down due to hash collisions, but with a good hash function, this is unlikely to happen. That makes the hash table a strong choice for systems that need fast lookups, like this course advising program.