BEGIN

// Define a structure to hold detailed course information

STRUCT CourseDetails

STRING courseID // The unique identifier of the course

STRING courseName // The name or title of the course

LIST<String> prerequisites // List of course prerequisites (other course IDs)

END STRUCT

// Function to load courses from a file into a hash map for quick lookup

FUNCTION loadCourseInformation(FILE\_PATH)

DECLARE HASH\_MAP<CourseDetails> courseCatalog

OPEN FILE\_PATH FOR READING

IF file cannot be opened THEN

PRINT "Error: Could not access the data file."

RETURN EMPTY HASH\_MAP

END IF

FOR each line in file DO

SPLIT line into parts using commas

// Ensure that the line contains at least the course ID and course name

IF length of parts < 2 THEN

PRINT "Invalid line detected, skipping: " + line

CONTINUE

END IF

DECLARE courseDetail = NEW CourseDetails

courseDetail.courseID = parts[0]

courseDetail.courseName = parts[1]

// Collect any additional prerequisites if they are provided

FOR each part from the third onward DO

APPEND part TO courseDetail.prerequisites

END FOR

// Store the course in the hash map with the course ID as the key

INSERT courseDetail.courseID => courseDetail INTO courseCatalog

END FOR

CLOSE file

RETURN courseCatalog

END FUNCTION

// Function to verify if all course prerequisites are valid and exist

FUNCTION checkCoursePrerequisites(HASH\_MAP<CourseDetails> courseCatalog)

DECLARE SET<String> availableCourseIDs

// Collect all course IDs present in the courseCatalog hash map

FOR each key, course IN courseCatalog DO

ADD course.courseID TO availableCourseIDs

END FOR

// Check for missing prerequisites in the catalog

FOR each key, course IN courseCatalog DO

FOR each prerequisite IN course.prerequisites DO

IF prerequisite NOT IN availableCourseIDs THEN

PRINT "Error: Prerequisite " + prerequisite + " for course " + course.courseID + " does not exist."

END IF

END FOR

END FOR

END FUNCTION

// Function to display details for a specific course, given a courseID

FUNCTION showCourseDetails(HASH\_MAP<CourseDetails> courseCatalog, STRING courseID)

IF courseID EXISTS IN courseCatalog THEN

DECLARE CourseDetails selectedCourse = courseCatalog[courseID]

PRINT "Course ID: " + selectedCourse.courseID

PRINT "Course Name: " + selectedCourse.courseName

PRINT "Prerequisites:"

IF selectedCourse.prerequisites IS EMPTY THEN

PRINT "No prerequisites."

ELSE

FOR each prereq IN selectedCourse.prerequisites DO

PRINT prereq

END FOR

END IF

ELSE

PRINT "Error: Course with ID " + courseID + " not found in the catalog."

END IF

END FUNCTION

// Main logic that runs the course management system

FUNCTION runCourseManagementSystem()

STRING courseFile = "course\_information.txt"

HASH\_MAP<CourseDetails> courseCatalog = loadCourseInformation(courseFile)

IF courseCatalog IS EMPTY THEN

RETURN

END IF

CALL checkCoursePrerequisites(courseCatalog)

WHILE TRUE DO

PRINT "Enter the course ID you want to search for (or type 'EXIT' to quit): "

STRING userChoice = GET USER INPUT

IF userChoice == "EXIT" THEN

BREAK

END IF

CALL showCourseDetails(courseCatalog, userChoice)

END WHILE

END FUNCTION

CALL main()

END

This version of the pseudocode uses a hash table to store and manage course data, which significantly improves performance when accessing specific courses. Reading the file and validating prerequisites both operate in O(n) time, since each course must be individually processed. The main advantage of using a hash table is its average O(1) lookup time, making it ideal for quickly retrieving course details by ID. In contrast, a vector would require O(n) time for each search, and a tree structure would offer O(log n) time but with added complexity and overhead. Overall, the hash table is the most efficient and practical choice for this program, especially when dealing with larger datasets and frequent searches.