**VECTOR DATA PSEUDOCODE**

'Design pseudocode to define how the program opens the file, reads the data from the file, parses each line, and checks for file format errors.'

* To read the file, open it.
* Verify that the file is present.
* Go over every line in the file once.
* Divide the line into fields using commas.
* Verify that the line has two or more parameters.
* Verify the courses' validity and whether there are any prerequisites.
  + Shut down the document.
* To show that the file format is valid, return true.
* Repeat for every course in the vector.
* Check if the courseNumber matches the given courseNumber
  + Course exists
  + Course does not exist

'Design pseudocode to show how to create course objects and store them in the appropriate data structure. '

* Open the file for reading
* Iterate through each line in the file
* Split the line by comma into fields
* Take a course extract. Number, name of the course, and requirements from the fields
* Create a new course object and store it in the vector
  + Close the file
* Create a new Course object
* Set the attributes of the Course object
* Return the created Course object

'Design pseudocode that will search the data structure for a specific course and print out course information and prerequisites. '

* Set up a variable to record whether the course is located.
* Iterate through each course in the vector
* Check if the courseNumber matches the given courseNumber
* Course is found
* Print the course information
* Check if prerequisites exist and print them if they do
  + If no prerequisites, print a message
  + If the course is not found, print a message

**HASH TABLE PSEUDOCODE**

* Start Program
* Open files to read data and analyze every line
* Check for courses titles and numbers
* IF the prerequisite is found
  + Add the course to an array
* IF the course parameters are less than two
  + The course should be skipped
  + An error message is displayed
  + The program ends
* ELSE
  + Add course name, number, and prerequisite to the hash table
* IF prerequisite already exists
  + Check if it comes before the course
  + If it does come before the course add it to the hash table
* IF prerequisite is not found
  + Skip course and display error message
* Create a function with parameters
  + Initialize variables for courses and read files
* While file is open
  + Print course information
  + Store the objective of the course in a vector data structure

**TREE DATA PSEUDOCODE**

**Design pseudocode to define how the program opens the file, reads the data from the file, parses each line, and checks for file format errors.**

* openFile
* coursesTree = new Tree<Course>()
* for each line in file
* tokens = split(line, ",")
* if tokens length is less than 2
  + - print error message
* continue
* if (tokens[2] != "" && !coursesTree.contains(tokens[2]))
  + - print error message
* continue
* course = new Course(tokens[0], tokens[1], tokens[2])
* coursesTree.add(course)

**Design pseudocode to show how to create course objects and store them in the appropriate data structure, and that will print out course information and prerequisites.**

* void Print Course Information
* Get course number
* Print Course Number
* Print Course Title
* Print Prerequisites
  + if (course.prerequisites != "")
  + Print Course Information
* void Print sample schedule
* for course in courses.traverse()
  + - print Course Number
    - print Course Title
    - print Prerequisites
    - int num PrerequisiteCourses
    - int count = 0
    - Get Course Number
* if (course.prerequisites != "")
* count++
* count += numPrerequisiteCourses(courses, course.prerequisites)
* return count

MENU PSEUDOCODE

* Int for switch statement
* Set Int to 0
* Create a Bid variable
* If input does not equal 4
  + First case prints data structure
  + Second case prints course list
  + Third case prints course
* End Program

LIST OF THE COURSES IN THE PROGRAM PSEUDOCODE

* Create a sorting string
* Create a char that sets the length more than 1
* Create a string to char array
  + Sort the array
* Create integer for numbers
* Create integer for alphabet
* WHILE the alphabet int is <97
  + SET int to more than 1
* FOR numbers
  + IF int is less than 97
  + SET number int to more than 1
* ELSE set the apha int to more than 1
  + End for loop
* Create a string for the classes
* Print the courses in alphanumeric order

5. The assignment recommends managing course data and reading files using a one-dimensional vector data structure. This structure has the advantage of using less memory and handling files and courses. The primary drawback, though, is that various data types cannot be utilized, and pieces cannot be removed.

A hash table in a project allows for organized and stored information, can be called with a key, and allows for creation, deletion, and synchronization of unique elements. However, the main disadvantage is that synchronization can affect the speed of the project.

Though it takes longer to edit than a hash table, a tree structure improves organization by storing data in left or right boxes, allowing for extension and search capabilities throughout the project.

6. Among the three data structures, hash tables are the one I would use. Considering that the hash table offers several advantages despite the potential for slower operation. For example, being able to arrange things better. We can already categorize courses and other basic features, but adding more functionality or new products to the project wouldn't be an issue if the organization choose to pursue that avenue. In light of that, it would be the rationale behind my consideration of hash tables.