# CS 300 Pseudocode Document

# 1.

**Pseudocode for Opening and Validating the file:**

Open the file named "CourseInformation.txt".

For every line in the file:

Split the line by commas

Make sure there are at least two pieces of information

Check if the second piece matches an existing course

If any issues arise:

Print that there is a problem with the file format

**Pseudocode for Creating the Courses and Storing in Vector:**

Start with an empty vector called "courses".

For each line in the file:

Divide the line into sections by commas

Create a new course:

Use the first section as the course number

Use the second section as the course title

Keep the rest as prerequisites

Add the course into the vector

**Pseudocode for Creating the Courses and Storing in Hashtable:**

Start with an empty hashtable called "courseTable".

For each line in the file:

Divide the line into sections by commas

Create a new course:

Use the first section as the Course number

Use the second section as the Course title

Keep the rest as prerequisites

Put the course into the hash table, using the course number as a key

**Pseudocode for Creating the Courses and Storing in Tree:**

Start with an empty tree called "CourseTree".

For each line in the file:

Divide the line into sections by commas

Create a new course:

Use the first section as the course number

Use the second section as the course title

Keep the rest as prerequisites

Insert the course into the tree

**Pseudocode for Displaying the Course Information and Requirements:**

For each course in the data structure:

Get the course details using its number

Show the course number and title

If the course has prerequisites:

Display its list of prerequisites

**Pseudocode for Loading Data Structure:**

Load Data Structure:

Open the file named "CourseInformation.txt"

For every line in the file:

Split the line by commas

Create a new course:

Use the first section as the course number

Use the second section as the course title

Keep the rest as prerequisites

Add the course to the appropriate data structure

**Pseudocode for Print the course information and prerequisites:**

Print alpha-numeric course list:

Sort the course information by alpha-numeric course number from lowest to highest

Print the sorted list to display

**2.**

**Menu Pseudocode:**

Menu:

Print "Menu:"

Print "1. Load Data Structure"

Print "2. Print Course List"

Print "3. Print Course"

Print "4. Exit"

Read user choice

Perform actions based on user choice  
  
**3.**   
  
***Vector Pseudocode:***

**Pseudocode for Printing Courses in Alpha-numeric Order:**

Print Courses in Alpha-numeric Order:

Sort the courses vector by alpha-numeric course number from lowest to highest

For each course in the sorted vector:

Get the course details using its number

Show the course number and title

***Hash table Pseudocode:***

**Pseudocode for Printing Courses in Alpha-numeric Order:**

Print Courses in Alpha-numeric Order:

Extract all courses from the hashtable into a vector

Sort the vector by alpha-numeric course number from lowest to highest

For each course in the sorted vector:

Get the course details using its number

Show the course number and title

***Tree Pseudocode:***

**Pseudocode for Printing Courses in Alpha-numeric Order:**

Print Courses in Alpha-numeric Order:

Perform an inorder traversal of the tree "courseTree"

For each course visited:

Get the course details using its number

Show the course number and title

**4.**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **Vector** | | | |
| **Opening and reading the file** | 1 | O(n) | O(n) |
| **Parsing every line and creating the course objects** | 1 | O(n) | O(n) |
| **Hash Table** | | | |
| **Opening and reading the file** | 1 | O(n) | O(n) |
| **Parsing every line and creating the course objects** | O(1) | O(n) | O(n) |
| **Balanced Binary Search Tree** | | | |
| **Opening and reading the file** | 1 | O(n) | O(n) |
| **Parsing every line and creating the course objects** | O(log n) | O(n) | O(n log n) |

**5.**

***Vectors:***

**Pros:**

* Vectors are great when you need to go through data one piece at a time, like reading a book from start to finish.
* They can store a lot of data without any trouble.

**Cons:**

* Adding or removing data from the middle of a vector can be slow because it must shuffle everything around.
* Finding specific data in a vector might take longer compared to other methods like using a list of words or a tree.

***Hash Tables:***

**Pros:**

* Hash tables are super quick when you're looking for something specific.
* They're perfect for quickly finding things you need in a big pile of stuff.

**Cons:**

* They can be a bit complicated to set up, and they need a lot of space to work properly.

***Trees:***

**Pros:**

* Trees are good at putting data in order and adding or removing stuff.
* They're handy for making sure things are neatly organized.

**Cons:**

* Trees can use up a lot of space because of the way they're built.

**6.** My recommendation is to use Hash Table for our code. It has significantly faster look up times for (O(1)), the hash table makes sure that we can access course information whenever needed, as fast as possible, while providing a seamless experience for users. Its ability to store data with unique identifiers makes it the perfect choice to organize our course details efficiently. By choosing hash table, we're prioritizing speed and efficiency, which makes sure our program runs smoothly and effectively.