# CS 300 Pseudocode Document

**This pseudocode outlines how to load data from a text file containing course information, store it in a vector of course objects, validate prerequisites, and allow searching and printing of course details for ABC University (ABCU) advisors.**

//Data Structures

Define struct Course:

String courseNumber

String courseName

List<String> prerequisites

Initialize Vector<Course> courses

//Load Parse File

// loadCourses() Function

Function loadCourses(filePath):

Open file at filePath

If file cannot be opened:

Print error message

Exit function

For each line in the file:

Split line by commas into tokens

If number of tokens < 2:

Print error: "Invalid line, not enough parameters"

Continue to next line

Set courseNumber = tokens[0]

Set courseName = tokens[1]

Initialize empty list prerequisites

For each token from index 2 to end:

Add token to prerequisites

Create Course object with courseNumber, courseName, prerequisites

Add Course to courses vector

Close file

// Validate prerequisites

For each course in courses:

For each prereq in course.prerequisites:

If no course in courses has courseNumber equal to prereq:

Print error: "Invalid prerequisite " + prereq + " for course " + course.courseNumber

Return courses

// printAllCoursesSorted() Function

Function printAllCoursesSorted(Vector<Course> courses):

Copy courses to new Vector sortedCourses

Sort sortedCourses by courseNumber (alphanumeric)

For each course in sortedCourses:

Print course.courseNumber + ": " + course.courseName

//Search and Print Course Information

// searchCourse() Function

Function searchCourse(Vector<Course> courses, String searchNumber):

For each course in courses:

If course.courseNumber == searchNumber:

Print "Course Number: " + course.courseNumber

Print "Course Name: " + course.courseName

If course.prerequisites is empty:

Print "Prerequisites: None"

Else:

Print "Prerequisites:"

For each prereq in course.prerequisites:

Print prereq

Return

Print "Course not found"

//Main Program Flow

Start Program

Prompt user for file path

Call loadCourses(filePath), store result in courses vector

While True:

Print "Menu:"

Print "1. Load Course Data"

Print "2. Print All Courses (Sorted)"

Print "3. Search for a Course"

Print "9. Exit"

Get user choice

If choice == 1:

Prompt user for file path

Call loadCourses(filePath)

Else if choice == 2:

Call printAllCoursesSorted(courses)

Else if choice == 3:

Prompt user for course number

Call searchCourse(courses, enteredCourseNumber)

Else if choice == 9:

Print "Goodbye"

Exit loop

Else:

Print "Invalid choice, try again"

End Program

## Runtime Analysis

* n shows up in multiple parts of your pseudocode (load, validate, print).
* n·k comes from validating each course's prerequisites.
* n·log(n) comes from sorting the course list alphanumerically.
* In Big O, we drop constants and less significant terms, so O(n·k + n·log(n)) is the most accurate and simplified form.

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **Load Course Data** | 1 | n | n |
| |  | | --- | | **Validate Prerequisites** |  |  | | --- | |  | | 1 | n · k | n ·k |
| **Search for a course** | 1 | n + k | n + k |
| **Sort Courses by number** | log(n) | n | n log n |
| **Print all courses** | 1 | n | n |
| **Total Cost** | | | 2n + n·k + k + n·log(n) |
| **Runtime** | | | O(n·k + n·log(n)) |

**Advantages and Disadvantages of Each Data Structure**

**Vector**

**Advantages:**

* Simple to implement and use.
* Efficient memory usage with contiguous storage.
* Fast access by index.

**Disadvantages:**

* Searching for a course is O(n) in the worst case.
* Sorting is required before displaying it in alphanumeric order.
* Insertion and deletion can be inefficient, especially in the middle of the structure.

**Hash Table**

**Advantages:**

* Very fast lookup, insertion, and deletion (O(1) on average).
* Efficient for searching specific course numbers.

**Disadvantages:**

* Unordered structure; must extract and sort keys separately to print sorted list.
* Requires more memory due to hashing overhead.
* Collision handling adds implementation complexity.

**Binary Search Tree (BST)**

Advantages:

* Automatically maintains sorted order through in-order traversal.
* Efficient search, insertion, and deletion in average cases (O(log n)).

Disadvantages:

* Can degrade to O(n) in the worst case (unbalanced tree).
* Requires additional memory for node pointers.
* More complex to implement compared to vectors.

**References**

GeeksforGeeks. (2023, October 16). *Applications, advantages, and disadvantages of hash data structure*. <https://www.geeksforgeeks.org/dsa/applications-advantages-and-disadvantages-of-hash-data-structure/>

GeeksforGeeks. (2023, October 16). *Introduction to tree data structure*. <https://www.geeksforgeeks.org/dsa/introduction-to-tree-data-structure/>