1. **Common Course Structure**Struct Course:  
    courseNumber: String  
    title: String  
    prerequisites: List of Strings (0 to n)
2. **Vector Version**

**Load and Validate File**

Function LoadCourseData(filename):

Open file

If file fails to open:

Print error and exit

For each line in file:

Split line by commas → tokens

If tokens.length < 2:

Print warning and skip line

courseNumber = tokens[0]

title = tokens[1]

prerequisites = []

For i = 2 to tokens.length - 1:

Add tokens[i] to prerequisites

Create Course object with above

Add Course to vector 'allCourses'

Close file

**Validate Prerequisites**  
Function ValidateCourses(allCourses):

validCourseNumbers = set of all courseNumbers in vector

For each course in allCourses:

For each prereq in course.prerequisites:

If prereq not in validCourseNumbers:

Print warning with course number and missing prereq

**Find and Print One Course**

Function PrintCourseDetails(courseNumber):

For each course in allCourses:

If course.courseNumber equals courseNumber:

Print course number and title

If no prerequisites:

Print "No prerequisites"

Else:

Print "Prerequisites:" + list

Return

Print "Course not found"

**Print All Courses Sorted**

**// This function prints all courses in a sorted order by course number.**

Function PrintAllCourses():

Sort allCourses by courseNumber (ascending)

For each course in sorted allCourses:

Print courseNumber + ": " + title

1. **Hash Table Version**

**Load and Validate File**

Function LoadCourseData(filename):

Open file

If file fails to open:

Print error and exit

For each line in file:

Split line by commas → tokens

If tokens.length < 2:

Print warning and skip line

courseNumber = tokens[0]

title = tokens[1]

prerequisites = []

For i = 2 to tokens.length - 1:

Add tokens[i] to prerequisites

Create Course object

Insert Course into hash table with courseNumber as key

Close file

**Validate Prerequisites**

Function ValidateCourses(courseTable):

validCourseNumbers = list of all keys in hash table

For each course in hash table:

For each prereq in course.prerequisites:

If prereq not in validCourseNumbers:

Print warning with course number and missing prereq

**Find and Print One Course**

Function PrintCourseDetails(courseNumber):

Look up courseNumber in hash table

If found:

Print course number and title

If no prerequisites:

Print "No prerequisites"

Else:

Print "Prerequisites:" + list

Else:

Print "Course not found"

**Print All Courses Sorted**

**// This function prints all courses in a sorted order by course number.**

Function PrintAllCourses():

Create empty list

For each course in hash table:

Add course to list

Sort list by courseNumber

For each course in sorted list:

Print courseNumber + ": " + title

1. **Binary Search Tree Version**

**Load and Validate File**

Function LoadCourseData(filename):

Open file

If file fails to open:

Print error and exit

For each line in file:

Split line by commas → tokens

If tokens.length < 2:

Print warning and skip line

courseNumber = tokens[0]

title = tokens[1]

prerequisites = []

For i = 2 to tokens.length - 1:

Add tokens[i] to prerequisites

Create Course object

Insert Course into BST based on courseNumber

Close file

**Validate Prerequisites**

Function ValidateCourses(BST):

validCourseNumbers = InOrderTraverse(BST) to collect courseNumbers

For each course in BST (traverse):

For each prereq in course.prerequisites:

If prereq not in validCourseNumbers:

Print warning with course number and missing prereq

**Find and Print One Course**

Function PrintCourseDetails(courseNumber):

Traverse BST to find courseNumber

If found:

Print course number and title

If no prerequisites:

Print "No prerequisites"

Else:

Print "Prerequisites:" + list

Else:

Print "Course not found"

**Print All Courses Sorted**

**// This function prints all courses in a sorted order by course number.**

Function PrintAllCourses():

InOrderTraverse(BST)

For each node visited:

Print courseNumber + ": " + title

1. **Menu (Common to All)**

While True:

Print menu:

1 - Load Data

2 - Print All Courses (sorted)

3 - Print One Course Details

9 - Exit

Get user choice

If choice == 1:

LoadCourseData(filename)

Else if choice == 2:

Sort and print all courses by courseNumber

Else if choice == 3:

Ask user for courseNumber

PrintCourseDetails(courseNumber)

Else if choice == 9:

Break

Else:

Print invalid option

1. **Big O Runtime Analysis and Recommendation**

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **Vector** | **Hash Table** | **BST** |
| Load File | O(n) | O(n) | O(n log n) |
| Validate Prereqs | O(n²) | O(n) | O(n log n) |
| Search Course | O(n) | O(1) | O(log n) |
| Print All Sorted | O(n log n) | O(n log n) | O(n) |

1. **Advantage and Disadvantage**

**Vector:**Simple to use, easy syntax  
Slow for searching large lists (O(n)); sorting needed every time for ordered output → O(n log n).

**Hash Table:**  
Fast search (O(1)); good for direct key lookups.  
Not naturally sorted; must extract and sort every time to print courses ordered → adds O(n log n). Collisions may slightly affect lookup speed.

**BST:**  
Efficient searching (O(log n)) for balanced trees.  
Naturally sorted; easy in-order traversal prints ordered list in O(n).  
Slightly more complex to implement and maintain balanced tree if needed.

1. **Recommendation**

After comparing runtime and sorting impact, **I recommend Binary Search Tree** is for ABCU’s advising tool:

* Searching for courses: fast average O(log n)
* Printing all courses sorted: simple in-order traversal, O(n)
* Good balance between speed and ordered data without extra sorting step.

Hash Table is fast for lookups but needs extra steps to sort, which affects efficiency for frequent ordered listing.