# Pseudocode for Vector

Open and Parse File

START  
Open file  
IF file not found  
 DISPLAY error and EXIT  
Initialize empty vector courseList  
FOR each line in file  
 Split line into tokens by comma  
 IF tokens < 2  
 DISPLAY "Invalid line" and CONTINUE  
 SET courseNumber = token[0]  
 SET courseTitle = token[1]  
 SET prerequisites = remaining tokens  
 CREATE Course object with courseNumber, courseTitle, prerequisites  
 APPEND Course to courseList  
END FOR  
FOR each Course in courseList  
 FOR each prereq in Course.prerequisites  
 IF prereq not in courseList  
 DISPLAY error  
END FOR

Print Course Info

PROMPT user for course number  
SET found = false  
FOR each Course in courseList  
 IF Course.number == input  
 PRINT Course details  
 SET found = true  
 BREAK  
IF found == false  
 PRINT "Course not found"

Print All Courses (Alphanumeric)

SORT courseList by Course.number  
FOR each Course in sorted list  
 PRINT Course.number + Course.title + Prerequisites

# Pseudocode for Hash Table

Open and Parse File

START  
Prompt for file path and open file  
IF file fails to open  
 DISPLAY error and EXIT  
Initialize empty hashTable  
FOR each line in file  
 Split line into tokens by comma  
 IF tokens < 2  
 DISPLAY "Invalid line" and CONTINUE  
 SET courseNumber = token[0]  
 SET courseTitle = token[1]  
 SET prerequisites = remaining tokens  
 CREATE Course object  
 INSERT into hashTable using courseNumber as key  
END FOR

Print Course Info

PROMPT for course number  
LOOKUP courseNumber in hashTable  
IF found  
 PRINT Course details  
ELSE  
 PRINT "Course not found"

Print All Courses

INITIALIZE courseList from hashTable values  
SORT courseList by Course.number  
FOR each Course in list  
 PRINT Course.number + Course.title + Prerequisites

# Pseudocode for Binary Search Tree (BST)

Open and Parse File

START  
Open file  
IF file fails  
 DISPLAY error and EXIT  
Initialize empty BST  
FOR each line in file  
 Split line by comma  
 IF tokens < 2  
 DISPLAY "Invalid line" and CONTINUE  
 SET courseNumber = token[0]  
 SET courseTitle = token[1]  
 SET prerequisites = remaining tokens  
 CREATE Course object  
 INSERT into BST using courseNumber  
END FOR

Print Course Info

PROMPT for course number  
CALL BST\_Search(root, courseNumber)  
IF found  
 PRINT Course details  
ELSE  
 PRINT "Course not found"

Print All Courses (In-Order Traversal)

FUNCTION PrintInOrder(node)  
 IF node != NULL  
 CALL PrintInOrder(node.left)  
 PRINT node.Course.number + title + prerequisites  
 CALL PrintInOrder(node.right)

# Runtime Analysis Chart

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Vector | Hash Table | Binary Search Tree |
| File read/parse | O(n) | O(n) | O(n) |
| Insert | O(1) | O(1) avg / O(n) worst | O(log n) avg / O(n) worst |
| Search by course number | O(n) | O(1) avg / O(n) worst | O(log n) avg / O(n) worst |
| Sort for alphanumeric output | O(n log n) | O(n log n) | O(n) via in-order traversal |
| Memory usage | Low | Medium | High |

# Evaluation

Vector  
Pros:  
- Simple and intuitive  
- Easy to iterate and sort  
Cons:  
- Search is linear (O(n))  
- Duplicate checking is manual

Hash Table  
Pros:  
- Fast search and insert (O(1) average)  
- Direct lookup by course number  
Cons:  
- Requires good hash function  
- Cannot retrieve sorted list without extra step  
- Possible collisions and chaining overhead

Binary Search Tree  
Pros:  
- Automatically sorted data via in-order traversal  
- Efficient search and insert (O(log n) average)  
Cons:  
- Needs balanced tree to ensure performance  
- More complex to implement  
- Memory overhead due to node pointers

# Recommendation

I recommend using the Binary Search Tree (BST) for this program. It offers the best balance between efficient searching, automatic sorted order, and scalability. Although it uses more memory and requires balancing logic for optimal performance, it simplifies the task of displaying all courses alphanumerically—an essential feature for ABCU's advisors.