## Menu Pseudocode:

BEGIN PROGRAM

DECLARE userChoice AS INTEGER

DECLARE dataLoaded AS BOOLEAN = false

WHILE userChoice != 9 DO

PRINT "1. Load Data"

PRINT "2. Print Course List"

PRINT "3. Search for Course"

PRINT "9. Exit"

READ userChoice

IF userChoice == 1 THEN

CALL loadCourses()

SET dataLoaded = true

ELSE IF userChoice == 2 THEN

IF dataLoaded == false THEN

PRINT "Error: Load data first."

ELSE

CALL printCourseList()

END IF

ELSE IF userChoice == 3 THEN

IF dataLoaded == false THEN

PRINT "Error: Load data first."

ELSE

PRINT "Enter course number:"

READ searchKey

CALL searchCourse(searchKey)

END IF

ELSE IF userChoice == 9 THEN

PRINT "Exiting program..."

ELSE

PRINT "Invalid option."

END IF

END WHILE

END PROGRAM

COURSE STRUCTURE DEFINITION (Shared by All Structures)

CLASS Course

DECLARE courseNumber AS STRING

DECLARE courseTitle AS STRING

DECLARE prerequisites AS LIST OF STRING

END CLASS

## VECTOR VERSION (Milestone 1)

**1. Load and Validate**

FUNCTION loadCourses()

DECLARE filePath AS STRING = "courses.txt"

DECLARE lines AS LIST OF STRING

DECLARE courses AS LIST OF Course

OPEN filePath FOR reading

IF file not found THEN

PRINT "Error: File not found."

EXIT FUNCTION

END IF

READ all lines into lines

FOR EACH line IN lines DO

SPLIT line by "," INTO tokens

IF LENGTH(tokens) < 2 THEN

PRINT "Invalid line format."

EXIT FUNCTION

END IF

CREATE newCourse AS Course

SET newCourse.courseNumber = tokens[0]

SET newCourse.courseTitle = tokens[1]

FOR i FROM 2 TO LENGTH(tokens) - 1 DO

ADD tokens[i] TO newCourse.prerequisites

END FOR

ADD newCourse TO courses

END FOR

END FUNCTION

**2. Print Sorted List**

FUNCTION printCourseList()

SORT courses BY course.courseNumber ALPHABETICALLY

FOR EACH course IN courses DO

PRINT course.courseNumber + ", " + course.courseTitle

END FOR

END FUNCTION

**3. Search Course**

FUNCTION searchCourse(searchKey)

FOR EACH course IN courses DO

IF course.courseNumber == searchKey THEN

PRINT course.courseNumber + ", " + course.courseTitle

FOR EACH prereq IN course.prerequisites DO

FOR EACH c IN courses DO

IF c.courseNumber == prereq THEN

PRINT " Prerequisite: " + c.courseNumber + ", " + c.courseTitle

END IF

END FOR

END FOR

RETURN

END IF

END FOR

PRINT "Course not found."

END FUNCTION

## HASH TABLE VERSION (Milestone 2)

**1. Load Courses into Hash Table**

DECLARE courseTable AS HashTable<String, Course>

FUNCTION loadCourses()

OPEN file FOR reading

READ lines into allLines

FOR EACH line IN allLines DO

SPLIT line by "," INTO tokens

CREATE newCourse AS Course

SET courseNumber = tokens[0]

SET courseTitle = tokens[1]

FOR i FROM 2 TO LENGTH(tokens)-1 DO

ADD tokens[i] TO newCourse.prerequisites

END FOR

INSERT courseNumber -> newCourse INTO courseTable

END FOR

END FUNCTION

**2. Print Sorted List**

FUNCTION printCourseList()

DECLARE keys AS LIST = GET ALL KEYS FROM courseTable

SORT keys

FOR EACH key IN keys DO

DECLARE course AS Course = courseTable[key]

PRINT course.courseNumber + ", " + course.courseTitle

END FOR

END FUNCTION

**3. Search Course**

FUNCTION searchCourse(searchKey)

IF searchKey NOT IN courseTable THEN

PRINT "Course not found."

RETURN

END IF

DECLARE course AS Course = courseTable[searchKey]

PRINT course.courseNumber + ", " + course.courseTitle

FOR EACH prereq IN course.prerequisites DO

PRINT " Prerequisite: " + courseTable[prereq].courseNumber + ", " + courseTable[prereq].courseTitle

END FOR

END FUNCTION

## BINARY SEARCH TREE VERSION (Milestone 3)

**1. Load Courses into BST**

DECLARE courseTree AS BST<String, Course>

FUNCTION loadCourses()

OPEN file FOR reading

READ all lines into allLines

FOR EACH line IN allLines DO

SPLIT line into tokens

CREATE newCourse AS Course

SET courseNumber = tokens[0]

SET courseTitle = tokens[1]

FOR i FROM 2 TO LENGTH(tokens)-1 DO

ADD tokens[i] TO newCourse.prerequisites

END FOR

INSERT courseNumber -> newCourse INTO courseTree

END FOR

END FUNCTION

**2. Print Sorted List (in-order traversal)**

FUNCTION printCourseList()

INORDER\_TRAVERSAL(courseTree.root)

END FUNCTION

FUNCTION INORDER\_TRAVERSAL(node)

IF node IS NOT NULL THEN

INORDER\_TRAVERSAL(node.left)

PRINT node.course.courseNumber + ", " + node.course.courseTitle

INORDER\_TRAVERSAL(node.right)

END IF

END FUNCTION

**3. Search Course**

FUNCTION searchCourse(searchKey)

course = courseTree.get(searchKey)

IF course IS NULL THEN

PRINT "Course not found."

RETURN

END IF

PRINT course.courseNumber + ", " + course.courseTitle

FOR EACH prereq IN course.prerequisites DO

prereqCourse = courseTree.get(prereq)

PRINT " Prerequisite: " + prereqCourse.courseNumber + ", " + prereqCourse.courseTitle

END FOR

END FUNCTION

## RUNTIME ANALYSIS

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **Vector** | **Hash Table** | **Binary Search Tree** |
| Load Data | O(n) | O(n) | O(n log n) |
| Print List | O(n log n) | O(n log n) | O(n) |
| Search Course | O(n) | O(1) | O(log n) |
| Search Prereqs | O(n·p) | O(p) | O(p log n) |
| **Overall Worst-Case** | **O(n²)** | **O(n log n+p)** | **O(n log n + p log n)** |

## STRUCTURE ANALYSIS AND RECOMMENDATION

|  |  |  |  |
| --- | --- | --- | --- |
| **Structure** | **Pros** | **Cons** | **Overall Complexity** |
| Vector | Easy to implement, simple linear layout in memory | Slow searching and prerequisite lookups, resorting needed | O(n²) worst-case |
| Hash Table | Very fast lookups and insertions, efficient prerequisite search | Not naturally sorted (requires extra sort for listing) | O(n log n + p) |
| Binary Search Tree | Maintains sorted order naturally, efficient searches | More complex to implement and maintain (balancing may be required) | O(n log n + p log n) |

Recommended Structure: **Hash Table**

Justification:

* Fast lookup for course info and prerequisites.
* Even with sorting needed for listing, overall performance is better than Vector.
* Easier implementation than balanced BST for this use case