For Vector:

CLASS Course

courseNumber: STRING

courseTitle: STRING

prerequisites: VECTOR<STRING>

FUNCTION loadCoursesVector(filename: STRING) RETURNS VECTOR<Course>

vector = new Vector<Course>()

file = openFile(filename)

WHILE NOT endOfFile(file)

line = readLine(file)

course = parseLineToCourse(line)

vector.pushBack(course)

END WHILE

file.close()

vector.sortByCourseNumber()

RETURN vector

END FUNCTION

FUNCTION printCourseVector(vector: VECTOR<Course>, courseNumber: STRING)

FOR course IN vector

IF course.courseNumber == courseNumber

PRINT course.courseTitle

PRINT course.prerequisites

END IF

END FOR

END FUNCTION

FUNCTION printAllCoursesVector(vector: VECTOR<Course>)

FOR course IN vector

PRINT course.courseNumber, course.courseTitle

END FOR

END FUNCTION

For Hash Table:

CLASS Course

courseNumber: STRING

courseTitle: STRING

prerequisites: LIST<STRING>

FUNCTION loadCoursesHashTable(filename: STRING) RETURNS HASHTABLE<STRING, Course>

hashtable = new Hashtable<STRING, Course>()

file = openFile(filename)

WHILE NOT endOfFile(file)

line = readLine(file)

course = parseLineToCourse(line)

hashtable.insert(course.courseNumber, course)

END WHILE

file.close()

RETURN hashtable

END FUNCTION

FUNCTION printCourseHashTable(hashtable: HASHTABLE<STRING, Course>, courseNumber: STRING)

IF hashtable.contains(courseNumber)

course = hashtable.get(courseNumber)

PRINT course.courseTitle

PRINT course.prerequisites

ELSE

PRINT "Course not found."

END IF

END FUNCTION

FUNCTION printAllCoursesHashTable(hashtable: HASHTABLE<STRING, Course>)

keys = hashtable.keys()

sortedKeys = sort(keys)

FOR key IN sortedKeys

course = hashtable.get(key)

PRINT course.courseNumber, course.courseTitle

END FOR

END FUNCTION

For Tree:

CLASS Course

courseNumber: STRING

courseTitle: STRING

prerequisites: LIST<Course>

FUNCTION loadCoursesTree(filename: STRING) RETURNS TREE<Course>

tree = new BinaryTree<Course>()

file = openFile(filename)

WHILE NOT endOfFile(file)

line = readLine(file)

course = parseLineToCourse(line, tree)

tree.insert(course)

END WHILE

file.close()

RETURN tree

END FUNCTION

FUNCTION printCourseTree(tree: TREE<Course>, courseNumber: STRING)

course = tree.search(courseNumber)

IF course != NULL

PRINT course.courseTitle

FOR prerequisite IN course.prerequisites

PRINT prerequisite.courseNumber, prerequisite.courseTitle

END FOR

ELSE

PRINT "Course not found."

END IF

END FUNCTION

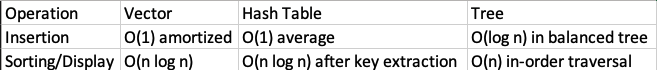
FUNCTION printAllCoursesTree(node: NODE<Course>)

IF node != NULL

printAllCoursesTree(node.left)

PRINT node.value.courseNumber, node.value.courseTitle

printAll



**Data Structure Analysis**

* **Vector**:
  + Pros: Simple implementation, efficient access by index.
  + Cons: Requires sorting after insertions for alphanumeric order.
* **Hash Table**:
  + Pros: Fast access, especially for looking up specific courses.
  + Cons: Does not maintain order; requires sorting of keys for ordered display.
* **Tree**:
  + Pros: Maintains order naturally, efficient insertion and search.
  + Cons: Can become unbalanced without self-balancing logic.