**// Main Menu Function**

FUNCTION mainMenu()

DECLARE courseVector AS Vector<Course>

DECLARE courseHashTable AS HashTable

DECLARE courseTree AS BinarySearchTree

DECLARE dataLoaded AS BOOLEAN ← FALSE

WHILE TRUE

PRINT "-------------------------------------"

PRINT "1. Load course data"

PRINT "2. Display sorted course list (CS only)"

PRINT "3. Find and display course information"

PRINT "9. Exit"

PRINT "-------------------------------------"

INPUT userChoice

IF userChoice == 1 THEN

PRINT "Loading course data..."

courseVector ← loadCourses("courses.txt")

courseHashTable ← loadHashTable("courses.txt")

courseTree ← loadCourseTree("courses.txt")

SET dataLoaded ← TRUE

PRINT "Courses loaded successfully."

ELSE IF userChoice == 2 THEN

IF NOT dataLoaded THEN

PRINT "Error: Load data first."

ELSE

PRINT "Sorted Computer Science Courses (Vector):"

CALL displaySortedCourses(courseVector)

// Alternatively:

// CALL courseTree.inOrder() for BST version

END IF

ELSE IF userChoice == 3 THEN

IF NOT dataLoaded THEN

PRINT "Error: Load data first!"

ELSE

PRINT "Enter course number to search (e.g., CS-101):"

INPUT searchKey

CALL searchCourse(courseVector, searchKey)

// Or use hash table: CALL courseHashTable.search(searchKey)

// Or BST: CALL courseTree.search(searchKey)

END IF

ELSE IF userChoice == 9 THEN

PRINT "Exiting program."

BREAK

ELSE

PRINT "Invalid selection. Try again."

END IF

END WHILE

END FUNCTION

**// Display Courses**

FUNCTION displaySortedCourses(courseVector)

// Simple Bubble Sort by courseNumber

FOR i FROM 0 TO LENGTH(courseVector) - 2

FOR j FROM i + 1 TO LENGTH(courseVector) - 1

IF courseVector[i].courseNumber > courseVector[j].courseNumber THEN

SWAP courseVector[i] WITH courseVector[j]

END IF

END FOR

END FOR

FOR EACH course IN courseVector

IF course.courseNumber STARTS WITH "CS"

PRINT course.courseNumber + ": " + course.courseName

END IF

END FOR

END FUNCTION

**Analysis Chart**

|  |  |  |  |
| --- | --- | --- | --- |
| Operations | vector/List | Hash Table | Binary Search Tree |
| Insert | O(1) (end) / O(n) (beginning) | O(1) average, O(n) worst (chaining) | O(log n) average, O(n) worst (unbalanced) |
| Search | O(n) | O(1) average, O(n) worst | O(log n) average, O(n) worst |
| Remove | O(n) | O(1) average, O(n) worst | O(log n) average, O(n) worst |
| Print all | O(n) | O(n) | O(n) (in order) |
| Memory Use | Low | Moderate | High |

**KEY**

O(1) – Constant time: Operation takes the same amount of time regardless of data size.

O(n) – Linear time: Time grows in direct proportion to number of elements.

O(log n) – Logarithmic: Time grows slowly as data increases (e.g., binary search).

Insert O(1)/O(n) – Inserting at the end of a vector is fast (O(1)), but inserting at the beginning is slower due to shifting elements (O(n)).

**Advantages VS. Disadvantages**

Vectors and lists have some great advantages that have many practical uses. They are simple and easy to use, making it a good choice for low-end programs. They are great for programs that do not require a lot of space or memory. Another advantage of vectors/lists are their inherent ability for appending data and printing in order. Again, they are extremely practical and useful for programs that are simple and do not require a lot of complex algorithms, but not ideal for what this program needs. The downside to using vectors or lists is that their search time is extremely inefficient with large datasets. Binary search trees are important and efficient at keeping that data sorted and traversing through it all in an efficient manner. The biggest downfall of binary search trees is the memory overhead that it requires due to the recursive structure. Lastly, there is the hash table. I believe the hash table would be the best fit for a program like this because of its quick lookup time, efficient ability to insert and remove data, as well as its ability to handle large data sets. This program clearly will involve a lot of data that needs to be accessed and manipulated regularly. It does have one main downfall consisting of the data not being sorted, however, I still think this is the best option for the Computer Science Department at ABCU.