Corey Gaspar

4/8/2025

Project One Pseudocode

**START**

**FUNCTION main()**

**READ** user input for file path

**STORE** user input as CSV file path

**IF** user did not input anything for file path

**LOAD** default file path

**LOOP** while user choice is **NOT** 9:

**DISPLAY** menu options

**PRINT** “1. Load CSV File”

**PRINT** “2. Print List Alphanumerically”

**PRINT** “3. Print Course Data”

**PRINT** “9. Exit”

**GET** menuChoice

**GET** dataChoice

**IF** menuChoice is not between 1 and 3 **OR** menuChoice is not 9

**OUTPUT** "Error: Invalid input. Please choose a valid option."

**CONTINUE** to next iteration of loop

**IF** menuChoice = 1

**IF** dataChoice = ‘BinarySearchTree’

**CALL** loadBids() to load CSV data into BinarySearchTree bst

**ELSE** **IF** dataChoice = 'Vector'

**CALL** loadBids() to load CSV data into vector courseList

**ELSE** **IF** dataChoice = 'HashTable'

**CALL** loadBids() **TO** load CSV data into HashTable courseTable (with ascending order)

**OUTPUT** "Number of records loaded: " + number of records

**ELSE** **IF** menuChoice = 2

**IF** dataChoice = 'BinarySearchTree'

**CALL** validateTree(bst)

**ELSE** **IF** dataChoice = 'Vector'

**CALL** validateList(courseList)

**ELSE** **IF** dataChoice = 'HashTable'

**CALL** validateTable(courseTable)

**ELSE IF** menuChoice = 3

**GET** userSearch

**IF** dataChoice = 'BinarySearchTree'

**CALL** printCourseTree(userSearch)

**ELSE IF** dataChoice = 'Vector'

**CALL** printCourseList(userSearch)

**ELSE IF** dataChoice = 'HashTable'

**CALL** printCourseTable(userSearch)

**ELSE IF** menuChoice = 9

**OUTPUT** "Good bye"

**EXIT** program

**END LOOP**

**END FUNCTION main()**

**struct Course {}**

**STRING** courseID

**STRING** courseName

**INT** preCount

**STRING** preList

**FUNCTION Course() {**

**SET** courseID to empty

**SET** courseName to empty

**SET** preCount to 0

**SET** preList to empty

**CLASS BinaryTree{}**

**STRUCT** Node

Course

root

**POINT** to left child node

**POINT** to right child node

**FUNCTION** binaryTree()

**FUNCTION** printTree()

**CLASS HashTable{}**

**STRUCT** Bucket:

Course

Key

**NEXT** pointer

hashTable

**FUNCTION** hash(Key)

**FUNCTION** printTable()

**sortList()**

**IF** lowIndex >= highIndex:

**RETURN** nothing

**CALL** partition function and get pivot’s final position

**SORT** the lowest part

**SORT** the highest part

**END**

**partition**()

**FIND** the pivot element

**WHILE** lowIndex <= highIndex:

**MOVE** left until a larger element than the pivot is found

**MOVE** right until a smaller element than the pivot is found

**IF** both the low and high index have not crossed

**SWAP** both elements

**RETURN** highIndex

**END**

**printList()**

**LOOP** through courseList

**PRINT** courseID and courseName

**LOOP** 0 to preCount

**FOR EACH** Course in preList

**PRINT** course

**END**

**printTree**()

**CREATE** new root Node

**SET** root to NULL

**CHECK IF** node is null:

**IF** node is null:

**CALL** Node’s left pointer to find the left most Node

**PRINT** courseID and courseName

**LOOP** 0 to preCount

**FOR EACH** Course in preList

**PRINT** to console: courseID

**CALL** Node’s right pointer to find the right most Node

**END**

**printTable()**

**CREATE** a new Node pointer and Set to the address of the nodes beginning

**LOOP** through the list; starting at the beginning

**PRINT** courseID in Course struct found within tempCourse to console

**PRINT** courseName in Course struct found within tempCourse to console

**LOOP** 0 to preCount

**FOR EACH** Course in preList

**CALL** printCourse() passing prelist

**END**