DECLARE map courses

FUNCTION loadData()

Open file

FOR each line in the file DO

Parse course code, course title, and prerequisites

Add parsed data

END FOR

Close file

END FUNCTION

FUNCTION printCourses()

Extract list of course codes

Sort course codes

FOR each courseCode in sorted list DO

Print courseCode and title

END FOR

END FUNCTION

FUNCTION printCourseDetails(courseCode)

IF courseCode exists in courses map THEN

Display course title

display prerequisites

ELSE

Print Course not found error

END IF

END FUNCTION

FUNCTION main()

DECLARE boolean dataLoad= false

DECLARE integer option

DO

PRINT "1. Load data"

PRINT "2. Print sorted list of courses"

PRINT "3. Print course details"

PRINT "9. Exit"

PRINT "Enter your option: "

READ option

SWITCH option

CASE 1:

CALL loadData

SET dataLoaded = true

CASE 2:

IF dataLoaded THEN

CALL printCourses

ELSE

PRINT "Please load data first."

END IF

CASE 3:

IF dataLoaded THEN

DECLARE string courseCode

PRINT "Enter the course code: "

READ courseCode

CALL printCourseDetails(courseCode)

ELSE

PRINT "Please load data first."

END IF

CASE 9:

PRINT "Exiting program."

DEFAULT:

PRINT "Invalid option. Please try again."

END SWITCH

WHILE option != 9

RETURN 0

END FUNCTION

------------------------------------------------------------------------------------------------------------------

Part 3

Vector

Function list\_courses(vector){\

Sort Vector

FOR courses IN Vector

Display Courses

END FUNCTION

Function print\_course\_details(vector, course\_name)

FOR course IN vector

IF course.name == course\_name

Display course.title

Display course pre requirements

RETURN

Course not found

END FUNCTION

HASH TABLE

Function list\_courses(hashtable)

Emprty list course\_list

FOR course in hash\_table

Append course.name to course\_list

SORT course\_list alphabetically

FOR course\_name IN course\_list

Display course\_name

Function \_print\_course\_details(hash-table, course\_name)

IF course\_name IN hase\_table

Course =- hash\_table[course\_name]

Dispaly course title

Display course pre req

ELSE

Course not found

END FUNCTION

TREE

Function list\_course\_tree(tree)

FUNCTION in\_order(node)

IF node is not null

in\_order(node.left)

Display node.course name

in\_order(node.right)

call in\_order(tree.root)

END FUNCTION

Function print course\_details(tree,course\_name)

Node = tree.root

WHILE node is not null

IF course\_name = node.course\_name

DISplay node.course.titel

Display node.course pre req

RETURN

ELSE IF course \_name<node.coursename

Node = node.left

ELSE

Node=node.right

Dispaly course not found

END FUNCTION

4

Vector

Open and read file

Cost per line =1

Number of time executed O(n)

Big O value O(n)

Parsing Each line and creating obj

Open and read file

Cost per line =1

Number of time executed =O(n)

Big O value =O(n)

----------------------------------------------------------------------------------------------------------

HASH TABLE

Open and read file

Cost per line =1

Number of time executed O(n)

Big O value O(n)

Parsing Each line and creating obj

Open and read file

Cost per line = 0(1)

Number of time executed =O(n)

Big O value =O(n)

----------------------------------------------------------------------------------------------------------

TREE

Open and read file

Cost per line =1

Number of time executed O(n)

Big O value O(n)

Parsing Each line and creating obj

Open and read file

Cost per line = 0(LOG n)

Number of time executed =O(n)

Big O value =O(n log n)

Looking at Vectors the one important thing they have going for them is that they perform better in sequential access of data which goes hand in hand with that they have extremely large memory storage. Some of the Disadvantages of Vectors is that they putting in information or deleting some of the information can be expensive and taxing when doing in the middle of O(n) With that one of the Other disadvantages is that it tends to be slower than using hash tables and if you are also using trees. When looking at the advantages fir hash table you get things like a really fat look up time and it the ability to have really easy access to finding data. But with these fast and nice pros the disadvantage is that it requires a big amount of memory storage because of how complex the its implementation is. For Tree one big advantage is its efficiency for insertion and deletion with memory and its good at sorting. With this also come with a high memory usage. After looking through some options I will most likely use a hash table for the project unless I find something different and better or something work better for me even though it might be harder to code.