# Basic Struct within Vector

Course struct defines structure containing course information

**struct Course** {

string courseID - unique course identifier(example: CSCI101)

string courseTitle - title of course

vector<String> preRequisites - list of courseIDs that are prerequisites

Search courses by search term

**searchCourse(string searchTerm)**

- for (i = 0; i < courses.length; i++) - loop through courses

check if courseId or courseName contains searchTerm within the string

if found

output course info to console and return

otherwise, output "Course not found"

# Course Validation Logic

Checks basic validation to ensure the line is at least 2 arguments and throw if not

**validateLine(line)**

split the string by ","

check at least 2 items are in the array

if < 2

throw error

output message showing line number, content, and "not enough arguments"

Validate vector of preRequisites to ensure ids exist in course list

validatePrerequisites(vector prequisites)

for(i = 0; i < prerequisites.length; i++)

check prequisites exists in preReq hash table

if not, remove it and output error to console

# Quick Sort Logic

partitioning and sorting logic used in print methods

**int partition(vector<Course>& courses, int begin, int end)**

set low and high equal to begin and end

pick the middle element as pivot point

while not done

(We are comparing course.title)

keep incrementing low index while courses[low] < courses[pivot]

keep decrementing high index while courses[pivot] < courses[high]

If there are zero or one elements remaining,

all courses are partitioned. Return high

else swap the low and high courses

move low and high closer ++low, --high

return high;

**void quickSort(vector<Course>& courses, int begin, int end**)

int mid = 0

if (begin >= end) return

mid = partition(courses, begin, end)

quickSort(courses, begin, mid)

quickSort(courses, mid + 1, end)

# Print Logic

Iterate through vector, sort, and print

**printCourseList(Vector courses)**

if courses is null

output “No Courses”

return

create new vector sortedList from courses vector

quickSort(sortedList, 0, sortedList.size() - 1);

loop through courses(from sortedList begin to sortedList.size)

printCourse(courseID)

Print course info

**printCourse(string courseID)**

search(courseID)  
 cout << courseID << “ “ << courseTitle << “ “ << “Prerequisites: “

iterate through prerequisites vector and print each entry  
 if vector is empty, print “N/A”

Print Menu Text

**printMenu()**

output the following lines:

“Menu:”  
“1: Load Courses”  
“2: Display all courses”  
“3: Find Course”  
“9: Exit”  
“Enter option”

# Menu Logic

load courses from file

**vector<Course>loadCourses(string filePath)**

output "loading file" to console

initialize vector<Course> courses - This will hold the list of courses

load file

create coursesHashTable (size is rowcount)

for(i = 0; i < rowCount; i++)

validateLine() - check if this line is even valid first

if line valid

Course course

course.courseID = file[i][0] - id is the first argument in line

add courseID to courses hashtable

course.courseTitle = file[i][1] - title is second

add remaining arguments to course.preRequisites vector (we will

validate these once we have all courses)

courses.push\_back(course)

loop through all items in courses vector

validatePrerequisites(course)

Call menu functions, get user choice and dispatch actions

**loadMenu()**

int choice = 0

Vector courses

string csvPath = (some hardcoded path for file)

printMenu()

cin >> choice

switch

case 1:

loadCourses(csvPath)

case 2:

printCourseList(courses)

case 3:

cout << “Enter ID for course: ”

string courseID

cin >> courseID

printCourse(courseID)

case 9:

exit

default:

output “Please enter a valid choice”

recursively call printMenu()