# Basic Structs within Tree

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

Course(string courseID, string courseTitle) default constructor

courseID = courseID;

courseTitle = courseTitle

Course(string courseID, string courseTitle, vector<string> preRequisites) overload

courseID = courseID;

courseTitle = courseTitle

preRequisites = preRequisites

Tree Node Struct

struct Node

Course course

Node left

Node right

Node() – default constructor

set left and right to null

Node(course)

set left and right to null

set course to course param

BinaryTree Class

Node root

Constructor  
 set root to null

# Tree Methods

Add course to Tree

**AddCourse**(node, course)

if node is larger

add to left

if no left node

this node becomes left

else

recurse down the left node

else

if no right node

this node becomes right

else

recurse down the left node

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"

Search Tree using courseId as search term

**Search**(courseId)

create current variable = root

while current != null pointer

if match found,

return current bid

if course is smaller than current node

then traverse left

else

larger so traverse right

# Print Logic

Recursively print courses in tree

**PrintCourseList**(root)  
 printCourses(root)

**printCourseList**(Node\* node)  
 if node != nullptr  
 printCourses(node->left)  
 Print course ID, title, and prerequisites  
 printCourses(node->right)

Print Single course Info

**printCourse(courseID)**

foundCouse = 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”

# Course Validation Logic

Iterate through course Prerequisites and validate courseIds, deleting invalid entries

**validateCoursePrerequisites**(Course course)  
 iterate through course prequisites vector  
 call Search(courseID) for each item  
 if not found  
 output error  
 remove prerequisite from vector  
 otherwise keep going until end

Iterate through entire Tree and validate all prerequisites for each course in tree

**ValidatePrerequisites**(root)  
 validatePrerequisites(root)

**validatePrerequisites**(Node\* node)  
 if node != nullptr  
 validatePrerequisites (node->left)  
 validateCoursePrerequisites(node)  
 validatePrerequisites (node->right)

# Menu Logic

Load courses from CSV

**loadCourses**(string filePath, tree)

output "loading file" to console

initialize Binary Tree courses

load file

create Binary Tree Courses

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

validateLine() - Check if line is valid before writing to Binary Tree

// If basic validation passes, write to Binary Tree

if valid

Course course

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

add courseID to courses tree

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

add remaining arguments to course.preRequisites vector (we will

validate these once we have all courses)

AddCourse(course)

otherwise print error

Call menu functions, get user choice and dispatch actions

**loadMenu()**

int choice = 0

string csvPath = (some hardcoded path for file)

courseTree = new Tree()

printMenu()

cin >> choice

switch

case 1:

caseTable = new courseTable()

loadCourses(csvPath, courseTable)

case 2:

printCourseList(courseTree)

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()