Edwin Jones

CS300 3-3: Project One Milestone One

//define structures

struct CourseInfo {

string code

string name

vector<string> prerequisites

}

struct Node {

CourseInfo course

Node \*left

Node \*right

Node(){

left = nullptr

right = nullptr

}

Node(CourseInfo aCourse) inherit Node() {

course = aCourse

}

}

// define class with methods to create and add nodes to binary search tree

class BST {

**public:**

BST() { // default constructor

root = nullptr

}

~BST() { //destructor

while root != nullptr

create pointer current type Node = root

while not a leaf node

if current->left != nullptr

set current = current-> left

continue

if current ->right != nullptr

set current = current->right

delete current

}

void ValidateCourses() {

pass root through validateCourses()

}

void InOrder() {

pass root through inOrder

}

void PostOrder() {

pass root through postOrder()

}

void Insert(CourseInfo course) {

if root = nullptr

set root = pass course through a new Node

else

pass root and course through addNode()

}

void Remove(string code) {

set root = root and course passed through removeNode()

}

CourseInfo Search(string code) {

create pointer current type Node = root

while current != nullptr

if current code = code

return current code

else if current code > code

set current = current->left

else

current = current->right

create empty course

return course

}

**private:**

Node\* root;

void validateCourses(Node \*node){ // Verify that each unique course prerequisite // is listed as a course

if node != nullptr

pass node->left through validateCourses()

for each prereq string in node->course.code

set result = result of passing prereq through Search

if result is not empty

continue

else

throw an error // prereq listed but no associated course

pass node->right through validateCourses()

}

//addNode compares string values (e.g. A < B, B < C, "1" (49 in ascii) < "2" (50 in ascii)).

//assumes all letters in each course code is either uppercase or lowercase

void addNode(Node\* node, CourseInfo course) {

create pointer current type Node = node

while current != nullptr

if current code > course's code

if current->left = nullptr

set current->left = course passed through a new Node()

return

else

current = current->left

else

if current->right = nullptr

set current->right = course passed through a new Node()

return

else

set current = current->right

}

void inOrder(Node \*node) {

if node != nullptr

pass node->left through inOrder()

print node's course information

pass node->right through inOrder()

}

void postOrder(Node \*node) {

if node != nullptr

pass node->left through postOrder()

pass node->right through postOrder()

print node's course information

}

void preOrder(Node \*node) {

if node != nullptr

print node's course information

pass node->left through preOrder()

pass node->right through preOrder()

}

Node \*removeNode(Node \*node, string code) {

if node = nullptr

return node

if code < node's code

node->left = node->left and code passed through removeNode()

return node

else if code > node's code

node->right =node->right and code passed through removeNode()

if node->left = nullptr

create pointer temp type Node = node->right

delete node

return temp

else if node->right = nullptr

create pointer node temp type Node = node->left

delete node

return temp

create pointer tempNode type Node = node->right

while node->left != nullptr

set tempNode = tempNode->left

set node->left = tempNode->course

set node->right=pass node->right and tempNode's code through removeNode()

return node

}

}

void parseCSV(string csvFile, BST \*bst) { //format CSV file for BST

ifstream file(csvFile)

string line

if file is not open

throw an error

while there is a row, assign the content of the row to line

if the length of the line is 0

continue

if the end of the line doesn’t have a comma

add a comma to the end

if the number of commas found is < 2

throw an error

create an empty CourseInfo course object

set course.code = string before first comma

delete index 0 to index of comma

set couse.name = string before first comma

delete index 0 to index of comma

while there are more than 1 chars in line

if there is no comma in line

add line to end of course.prerequisites vector

break

add string before next comma to the end of the course.prerequisites vector

set line = line with index 0 index of comma deleted

use Insert(course) to add to bst

close file

}

int main {

//how to load csv file into binary search tree

assign variable with CSV file

create pointer bst type BST = new BST object

create empty course of type CourseInfo

//how to print all in order courses (left->root->right)

bst->InOrder

//how to print all post order (left->right->root)

bst->PostOrder

//how to print all pre order (root->left->right)

bst->PreOrder

//how to check if prerequisites are also listed as courses

bst->ValidateCourses()

}