# CS 300 Pseudocode Document

## Function Signatures

NOTE: REVISED THE LOAD COURSE DATA FUNCTION TO CHECK IF A PREREQUISITE ALREADY EXISTS ACCORDING TO INSTRUCTOR FEEDBACK. THIS FUNCTION CLOSELY MIRRORS WEEK 3’S FUNCTION ASIDE FROM THE COURSEMAP DATA OBJECT.

THE TABLE AT THE BOTTOM WAS REVISED AFTER CONSIDERING THAT MANY OF THE VALUES ARE RUN FOR [N-SIZED NUMBER OF COURSES] GIVEN [M-SIZED NUMBER OF PREREQUISITES].

Below are the function signatures that you can fill in to address each of the three program requirements using each of the data structures. The pseudocode for printing course information, if a vector is the data structure, is also given to you below (depicted in bold).

// ABC University Course Management System Pseudocode

// Structure to represent a course

struct Course {

string code;

string title;

vector<string> prerequisites;

};

// Function to load course data from a file into a hashtable

void loadCourseData(unordered\_map<string, Course>& courseMap, string filename) {

OPEN ifstream file(filename);

IF (file.is\_open()) {

DEFINEstring line;

WHILE (getline(file, line)) {

INITIALIZE string code, title, prerequisitesString;

// Parse the line to extract course code, title, and prerequisites

// ...

IF (/\* line is correctly formatted \*/) {

// Check if all prerequisites already exist in the courseMap

DEFINE bool prerequisitesExist = true;

FOR (const auto& prerequisite : prerequisites) {

IF (courseMap.find(prerequisite) == courseMap.end()) {

SET prerequisitesExist = false;

break;

}

}

IF (prerequisitesExist) {

// Create a new course object

INITIALIZE Course newCourse;

// Set the course code and title

// ...

// Parse the prerequisites string and add them to the course's prerequisites vector

// ...

// Add the course object to the courseMap with the course code as key

SET courseMap[code] = newCourse;

} ELSE {

// Display an error message indicating missing prerequisites

// ...

}

} ELSE {

// Display an error message indicating file format error

// ...

}

}

CALL file.close();

} ELSE {

// Display an error message indicating file open error

// ...

}

}

// Function to search for a course and print its information and prerequisites

void searchCourse(const unordered\_map<string, Course>& courseMap, string courseCode) {

if courseMap contains the 'courseCode':

course = courseMap[courseCode]

print the course code, title, and prerequisites

else:

print an error message indicating course not found

}

// Function to print a sample course listing

void printSampleListing(const unordered\_map<string, Course>& courseMap) {

for each course in courseMap:

print the course code and title

if the course has prerequisites:

for each prerequisite in the course's prerequisites:

print the prerequisite course code

}

// Function to count the number of prerequisites for a course

int numPrerequisiteCourses(const unordered\_map<string, Course>& courseMap, const Course& course) {

int totalPrerequisites = course.prerequisites.size()

for each prerequisite in course.prerequisites:

if courseMap contains the prerequisite:

prerequisiteCourse = courseMap[prerequisite]

totalPrerequisites += numPrerequisiteCourses(courseMap, prerequisiteCourse)

return totalPrerequisites

}

/ Function to print the course information and prerequisites

void printCourseInformation(const unordered\_map<string, Course>& courseMap, const string& courseNumber) {

if courseMap contains the 'courseNumber':

course = courseMap[courseNumber]

print the course code, title, and prerequisites

for each prerequisite in the course's prerequisites:

if courseMap contains the prerequisite:

prerequisiteCourse = courseMap[prerequisite]

print the prerequisite course information

else:

print an error message indicating course not found

}

// Main function

int main() {

declare an unordered\_map of Course objects: courseMap

declare a string variable: filename

declare a string variable: courseCode

prompt the user to enter the filename of the course data file

read the filename from the user

loadCourseData(courseMap, filename)

prompt the user to enter a course code to search

read the course code from the user

searchCourse(courseMap, courseCode)

printSampleListing(courses)

return 0;

}

## Example Runtime Analysis

When you are ready to begin analyzing the runtime for the data structures that you have created pseudocode for, use the chart below to support your work. This example is for printing course information when using the vector data structure. As a reminder, this is the same pairing that was bolded in the pseudocode from the first part of this document.

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for all courses** | 1 | n | n |
| **if the course is the same as courseNumber** | 1 | 1 | 1 |
| **print out the course information** | 1 | m | m |
| **for each prerequisite of the course** | 1 | m | m |
| **print the prerequisite course information** | 1 | m | O(m) |
| **Total Cost** | | | 3n+3m+2 |
| **Runtime** | | | O(n+m) |