**CS 300 Project One Revised Milestone One**

**Huan Ai**

**4/13/2025**

**// Include necessary libraries**

#include <iostream>

#include <fstream>

#include <sstream>

#include <vector>

#include <string>

**// Define a struct to hold course data**

struct Course {

std::string courseCode; // Course ID (e.g., CSCI100)

std::string title; // Course name

std::vector<std::string> prerequisites; // List of prerequisites (0-n)

};

**// Function prototypes**

std::vector<Course> loadCourseData(const std::string& filePath);

bool validatePrerequisites(const std::vector<Course>& courses);

void printCourse(const std::vector<Course>& courses, const std::string& courseCode);

Course\* findCourse(const std::vector<Course>& courses, const std::string& courseCode);

**// Main function**

**Main()**

Create a vector of Course structs named courses

Get CSV file path from user

If no input, use default path "courses.csv"

// Load and validate data

Set courses = loadCourseData(filePath)

If validatePrerequisites(courses) is False

Output "Error: Invalid prerequisites detected"

Exit program

// User interaction

Get user input for courseCode to search

Call printCourse(courses, courseCode)

End

**// Load course data**

loadCourseData(String filePath) Returns vector<Course>

Create empty vector<Course> named courseList

Open file at filePath

If file.fail()

Output "Error: Could not open " + filePath

Return empty courseList

While getline(file, currentLine)

Create stringstream named ss from currentLine

Create vector<string> named tokens

Create string named token

// Split line by commas

While getline(ss, token, ',')

// Remove leading/trailing whitespace

token = trim(token)

If token is not empty

tokens.push\_back(token)

// --- Validation ---

If tokens.size() < 2

Output "Error: Line missing required fields - " + currentLine

Continue to next line

// --- Course creation ---

Create Course named newCourse

newCourse.courseCode = tokens[0] // First token = course code

newCourse.title = tokens[1] // Second token = title

// Add prerequisites (remaining tokens)

For i from 2 to tokens.size()-1

newCourse.prerequisites.push\_back(tokens[i])

// Store in vector

courseList.push\_back(newCourse)

Return courseList

End

**// Validate all prerequisites exist as courses**

validatePrerequisites(vector<Course> courses) Returns Boolean

For each Course in courses

For each prereq in Course.prerequisites

Create found = false

// Check if prerequisite exists

For each potentialCourse in courses

If potentialCourse.courseCode equals prereq

Set found = true

Break loop

If not found

Output "Missing prerequisite: " + prereq + " for " + Course.courseCode

Return false

Return true

End

**// Print course details**

printCourse(vector<Course> courses, String courseCode)

Find course using findCourse(courses, courseCode)

If course is found

Output "Course: " + course.courseCode + " - " + course.title

If course.prerequisites is not empty

Output "Prerequisites:"

For each prereq in course.prerequisites

Output "- " + prereq

Else

Output "No prerequisites"

Else

Output "Course not found"

End

**// Helper function to find a course**

findCourse(vector<Course> courses, String targetCode) Returns Course\*

For each Course in courses

If Course.courseCode equals targetCode

Return memory address of Course // &Course

Return nullptr // Not found

End