**// Design pseudocode to define how the program opens the file, reads the data from the file, parses each line, and checks for file format errors.**

void loadCourseDataFromFile(String csvPath, Vector<Course> courses) {

print(“Loading CSV file: “ + csvPath)

// Initalize the CsV parser using the given path

csvParser file = csvParser(csvPath) // O(n)

vector<String> header = file.getHead()

for each column in header {

print(column + “ | “)

}

print(“”)

try {

// Loop through each row in the CSV file

for (unsigned int i = 0; I < file.rowCount(); i++) { // O(n)

// Validate that each row has at least two parameters

if (file[i].size() < 2) { // O(m \* n)

print(“Error: Invalid format.”)

continue

}

String courseNumber = file[i][0]

String courseName = file[i][1]

Course course = new Course(courseNumber, courseName)

// Check for prerequisites and ensure they exist as valid courses

for (unsigned int j = 2; j < file[i].size(); j++) {

String prerequisite = file[i][j]

if (!isCourseInFile(prerequisite, file)) {

print(“Error: prerequisite “ + prerequisite + “does not exist”

} else {

course.addPrerequisite(prerequisite)

} }

courses.add(course) }

} catch (csv::Errro &e) {

print(e.what())

} }

**// Design pseudocode to show how to create course objects and store them in the appropriate data structure.**

struct Course { // O(n)

String CourseNumber

String courseName

Vector<String> prerequisites // O(n \* m)

// Constructor

Course(String number, String name) {

courseNumber = number

courseName = name

prerequisites = new Vector<string> ()

}

// add prerequisite to the course

void addPrerequisite(String prerequisite) {

prerequisites.add(prerequisite)

} }

**// Quick Sort**

void quickSortCourses(Vector<Course>& courses, int low int high) {

if (low < high) {

int pivotIndex = partition(courses, low high);

quickSortCourses(courses, low, pivotIndex – 1); // sort left

quickSortCourses(courses, pivotIndex + 1, high); // sort right

} }

int partition(Vector<Course>& courses, int low int high) {

String pivot = courses[high].courseNumber;

int i = low – 1;

for (int j = low; j < high; j++) {

if(courses[j].courseNumber < pivot) {

i++

swap(courses[i],courses[j]);

} }

swap(courses[i + 1], courses[high]);

return i + 1;

}

**// Design pseudocode that will print out course information and prerequisites**

void printCourseInfo (Vector<Course> courses, String courseNumber) {

int index = binarySearchCourses(courses, courseNumber):

if (index != -1) {

Course\* course = &courses[index];

} else {

print(“Course not found”)’

} }

int binarySearchCourses(Vector<Course>& courses, String courseNumber) {

int low = 0;

int high = courses.size() -1;

while (low <= high) {

int mid = low + (high – low) / 2;

if (courses[mid].courseNumber == courseNunber) {

return mid;

} else if (courses[mid].courseNumber < courseNumber) }{

low = mid + 1;

} else {

high = mid – 1;

} }

return -1;

}

void printAllCourses(Vector<Course> courses) {

for each Course course in courses {

print(“Course Number: “ + course.courseNumber)

print(“Course Name: “ course.courseName)

if (course.prerequisites.size() > 0) {

print(“Prerequisites: “)

for each prerequisite in courseprerequisites {

print(“ – “ + prerequisite)

}

} else {

print(“No Prerequisites”)

} } }

// Check if course exists

bool isCourseInFile(String courseNumber, csvParser file) {

for (unsigned int i = 0; I < file.rowCount(); i++)

if (file[i][0] == courseNumber) {

return true

} }

return false

}

**// Create pseudocode for a menu.**

void displayMenu() {

println(“Main menu:”)

println (“Option 1: Load the file data into the data structure”)

println (“Option 2: Print an alphanumerically ordered list of all the courses in the Computer Science department.”)

println (“Option 3: Print the course title and the prerequisites for any individual course.”)

println (“Option 9: Exit the program.”)

}

void main() {

Vector<Course> courses = new Vector<Course>()

int choice = 0

while choice != 9 {

displayMenu()

print(“Enter your choice: “)

choice = get user input

switch choice {

case 1:

print(“Etner file path: “)

String filePath = get user input

loadCourseDataFromFile(filePath, courses)

break

case 2:

if courses.isEmpty() {

print(“No course data available”)

} else {

printSortedCourses(courses)

} break

case 3:

if courses.isEmpty() {

print(“No course data available.”)

} else {

print(“Enter course number: “)

String courseNumber = get user input

printCourseInfo(courses, courseNumber)

} break

case 9:

print(“Existing program.”)

break

default:

print(“Invalid choice. Please select a valid option.”)

} } }

**// Print alphabetically & sort**

void printSortedCourses(Vector<Course>& courses) {

quickSortCourses(courses, 0, courses.size() – 1); **// Sorting**

for each course in courses {

print(‘Course Number: “ + course.courseNumber + " | Course Title: " + course.courseName);

} }