**Sabastian Fasano**

**CS-300**

**February 2, 2025**

**Module 4 Milestone: Pseudocode**

Course Hash Table System

Load Required Libraries and Headers

* Include necessary text parsing libraries.
* Define a struct to store course data.

**Struct Definition: Course**

struct Course {

string courseID;

string courseName;

int preCount;

string preList;

Course() {

courseID = courseName = "";

preCount = 0;

preList = "";

}

};

Class Definition: HashTable

cpp

CopyEdit

class HashTable {

private:

struct Bucket {

Course course;

string key;

Bucket\* next;

};

vector<Bucket\*> hashTable;

public:

int hash(string key);

void printAll();

};

**Main Function**

cpp

CopyEdit

void main() {

// Create a list to store courses

List<Course> courseList;

// Get CSV file path from user

string filePath;

cout << "Enter CSV file path (or leave blank for default): ";

getline(cin, filePath);

if (filePath.empty()) {

filePath = "default.csv"; // Set default location

}

// Parse the CSV file

courseList = txtParser(filePath);

// Validate the parsed list

if (!validateList(courseList)) {

cout << "Error: Invalid course prerequisites detected." << endl;

return;

}

// Prompt user for a course search

string userSearch;

cout << "Enter course ID to search: ";

getline(cin, userSearch);

// Print course details

printCourse(userSearch);

}

**txtParser Function**

cpp

CopyEdit

List<Course> txtParser(string filePath) {

List<Course> tempList;

ifstream file(filePath);

if (!file.is\_open()) {

cout << "Error: Unable to open file." << endl;

return tempList;

}

string line;

while (getline(file, line)) {

vector<string> tokens = split(line, ','); // Assume a split function is available

if (tokens.size() >= 2) {

Course tempCourse;

tempCourse.courseID = tokens[0];

tempCourse.courseName = tokens[1];

string preNames;

int preCount = 0;

for (size\_t i = 2; i < tokens.size(); i++) {

preNames += tokens[i] + " ";

preCount++;

}

tempCourse.preCount = preCount;

tempCourse.preList = preNames;

tempList.push\_back(tempCourse);

}

}

file.close();

return tempList;

}

**searchList Function**

cpp

CopyEdit

Course searchList(string courseID) {

int index = hash(courseID);

Bucket\* tempBucket = hashTable[index];

while (tempBucket) {

if (tempBucket->course.courseID == courseID) {

return tempBucket->course;

}

tempBucket = tempBucket->next;

}

return Course(); // Return an empty course if not found

}

**printCourse Function**

cpp

CopyEdit

void printCourse(string courseID) {

Course course = searchList(courseID);

if (course.courseID.empty()) {

cout << "Course not found." << endl;

return;

}

cout << "Course ID: " << course.courseID << endl;

cout << "Course Name: " << course.courseName << endl;

if (course.preCount > 0) {

cout << "Prerequisites: " << course.preList << endl;

} else {

cout << "No prerequisites." << endl;

}

}

**validateList Function**

cpp

CopyEdit

bool validateList() {

for (auto& course : courseList) {

for (int i = 0; i < course.preCount; i++) {

Course prerequisite = searchList(course.preList[i]);

if (prerequisite.courseID.empty()) {

return false;

}

}

}

return true;

}

**Hash Function**

cpp

CopyEdit

int Hash(string key) {

// Custom hash function to handle courseID uniqueness

int hashValue = 0;

for (char ch : key) {

hashValue = (hashValue \* 31 + ch) % hashTable.size();

}

return hashValue;

}

**Key Updates and Improvements:**

* **Better structure and formatting**: Clearly separated sections with comments.
* **Explicit class and function definitions**: Converted the description into properly structured C++ code.
* **Error handling**: Added checks for file opening and course validation.
* **String splitting**: Assumed a split() function for CSV parsing.
* **Search function optimizations**: Improved loop handling within searchList