# CS 300 Pseudocode Document

## Pseudocode is used to open files, read data from the file, parse, and check for file format errors.

The application will first try to open a CSV file and will parse it line by line if the file exists and is readable. It will validate that each line is greater or equal to two items on a successful line reading. Otherwise, it will throw an error and quit.  
For every 3rd, 4th, …Nth index element in each parsed row (end of line), it will validate each prerequisite that, if such, exists in Vector<Node> courses as a node object. If an element was found in courses, create a new Course object, populate it with the parsed line, and call the hash function by passing the courseNumber parameter to compute the hash key value. Next, instantiate a course node and store the course in the courses Vector at index[key] as the first linked list element if the bucket is empty or as the 2nd or next node if the head node already exists to prevent collisions. Otherwise, quit.

Main():

filePath = "path/to/course\_data.csv"

Try to open file:

Parser Line = csv::Parser(filePath);

For each Line in the filePath, Try reading a line:

If line length ≥ 2:

Construct a New Course object:

courseNumber = lineData[0]

courseName = lineData[1]

For every Item in Line ≥ lineData[2]

Validate prerequisites:

If lineData[2 to N] exist:  
Get the hashedKey value from the Hash(courseNumber)

Check Vector courses[key] if course lineData[2 to end] exists as a head in the linked list or as any chained next nodes.

If node course is found in the hash table:

Populate the Course object.  
 Construct and store new Node(Course, hashedKey)

Add new Node to Vector<Course> courses[key] as head or as next node if the head is not empty.

Catch Can’tReadLineError: print("Error: Line not read at…")

Catch Can’tOpenFileError: print("Error: File not found at…")

## Pseudocode to show how to create course objects and store them in the appropriate data structure.

* The data will be stored in a HashTable of Course object nodes predefined to 53 buckets:

Vector<Node> courses(53);

* Each bucket will have an index as defined by the Modulo hash function:

function Hash(courseNumber, tableSize=53):

return hashValue % tableSize

* Each course will be stored as a node in a linked list and reside in its bucket depending on its hashed key value:

class Node:

Course course

Unsigned int key

Node \*next

Node constructor(courseData, hashedKey):

course = courseData

key = hashedKey

Node = nullptr

* Each course object will be structured and initialized as follows:

class Course object:

string courseNumber

string courceName

list<string> prerequisites

Course constructor(cNum, cNam, cpReq[]):

this courseNumber = cNum;

this courceName = cNam;

this prerequisites = cpReq[];

## Pseudocode that will print out course information and prerequisites.

The function searchCourse(HashTable<Course> courses, String courseNumber) will first use the Hash method to retrieve the hashed key value from Hash(course). Next, it will check the head of the linked list at the bucket key in HashTable [key] and will iterate through all the next nodes if head node.course.courseNumber is not equal to courseNumber. As courseNumber is found, all Course object attributes, including traversing through the prerequisites list, will be printed.

void searchCourse(HashTable<Course> courses, String courseNumber):

key = Hash(courseNumber)

currentNode = courses[key]

If currentNode is not null:

While currentNode.next is not null:

print("Course Number" + currentNode.course.courseNumber)

print("Course Name: " + currentNode.course.courceName)

While currentNode.course.prerequisites is not empty:

print("Prerequisite: " + courseNumber)

currentNode = currentNode.next

Return