Brian Engel

Milestone hash table

Class HashTable {

Structure Node to hold a course, a key, and a pointer to the next node

A vector of nodes to hold the hashtable

Node\* InsertCourse(Course course, Node\* previousNodePtr) Function to insert a course into the hashtable: at the start of the key if the list is empty or at the end of the key if it isn’t.

hash the course id to make a key

create a node and assign its key to the hashed key and course

if the vector of nodes at the key is empty

assign the node to the vector at the key

else

add the node to the end of the vector

previousNodePtr points to this node

return this nodes pointer

}

Struct Course {

String courseNumber

String name

Vector<string> prerequisites

}

HashTable readFile(string fileName) {

HashTable courseList

Ifstream file(fileName)

If file didn’t open

output “could not open file”

Return courseList

String lineInFile

PreviousNodePtr is set to null

While getline(file, lineInFile)

Course course

stringstream ss(lineInFile)

getline(ss, course.courseNumber, ‘,’)

if course.courseNumber is empty

output invalid coursenumber

Return courseList

getline(ss, course.name, ‘,’)

if course.name is empty

output invalid course name

return courseList

string prereq

while getline(ss, prereq, ‘,’)

if prereq in courseList.courseNumber

course.prerequisite.push\_back(prereq)

else

output invalid prerequisite

return courseList

InsertCourse(course, previousNodePtr)

File.close()

Return courseList

}

void printCourseInformation(HashTable courses, String courseNumber) {

For all keys in the HashTable

For all nodes in the key

if the course is the same as courseNumber

print out the course information

for each prerequisite of the course

print the prerequisite course information

}