# Struct Course

//outlines the content of each course that will be stored in the vector of courses

String courseNumber

String courseTitle

Vector <Prerecs> coursePrerecs // declares the vector of prerecs

# Struct Prerecs

//this will give us a structure to load prerequisites into

String preCourse

# displayCourse(Course course)

//prints out course information that is passed to the function

cout << course.courseNumber << “ | “<< course.courseTitle << “ | “<< endl

if (course.prerec.size() == 0)

Cout << “no prerequisites… “ << endl

Else

Cout << “Course Prerequisites: “<< endl

For (int i = 0, I < course. coursePrerecs.size(), ++I)

Cout << course.coursePrerecs.at(i) << endl

Return

# printCourses(vector <Course> courseList)

/\* This should print out a list of all the courses until the end of the linked list.

Int I = 0

While (I < courseList.size -1) //

Cout<< courseList.at(i).courseNumber<< “, ”<< courseList.at(i).courseTitle << endl

I++

//end iterator 1

Return

# loadCourses (string somePath)

// this loads and parses the *text file*. I find this interesting because we haven’t actually, or I haven’t actually parsed a text file to extract its data. Going to give it my best shot.

vector<Course> courseList // this creates a vector named courseList

cout << “Loading TXT file “ << somePath

//this initializes input file stream and opens up the text file

ifstream inputFile(“somePath\some\_file.txt”)

// if the input file isn’t actually open

if(!inputfile.is\_open())

cerr << “Error opening file.” << endl

return

//if the input file is open try the following stuff:

try

string line //This is to capture each line

// while loop to capture each line using getline

while(getline(some\_file.txt, line) // iterator 1 while there is a line in some file, store as line

Course course // create a course structure

int iterator = 0 // used to keep track of where we are in the line

stringstream ss(line) // get stringstream going to read the line

string info // this is to capture each course

while( getLine(ss, info, ‘,’) // Iterator 2 while loop to capture each bit within the text file using a comma as a delimiter.

if(iterator == 0)

// this is the course number

course.courseNumber = info

if(iterator == 1)

// this is the full title

course.CourseTitle = info

if(iterator >=2)

// this is where it starts to add prerecs to the vector of prerecs

course.course.Prerecs.push\_back(info)

Iterator++ // iterate the iterator

// end iterator 2; if there is no more delimited pieces of info, Iterate to the next line, if there is one

//insert the course into the linked list

courseList.push\_back(course)

// end iterator 1; if there is no more lines in the text file

file.close()

// close the text file once we’re done adding courses

return courseListing

# QualityControl(vector<Course> courseList)

/\* The way this is supposed to work: compare two course listings simultaneously using two for loops and a while loop: courseListing.at(i), courseListing.at(j). First check is if courseListing.at(i) doesn’t have any prereqs, iterate to the next courseListing.at(i). If it does have any prereqs then look at them and compare them to the list of courses to make sure they exist in the courseList. \*/

Int I

For (I = 0, I < courseListing.size(), i++) // iterator 1

// iterator to get us through the courses

If (courseListing.course.at(i).coursePrerecs.size() >= 1)

//If the course actually has prerequisites

Int j // iterator to get us through prerecs

// compare the prerequisite course numbers to the course numbers contained in the course listing

For (j = 0, j < courseListing.course.at(i).coursePrerecs.size(), j++) // iterator 2

Int k = 0

//while the courseNumber that’s being compared doesn’t equal the other courses courseNumber in the courseList…

While(courseListing.at(i).coursePrerecs.at(j).courseNumber.compare(courselisting.at(k).courseNumber()) != 0) // iterator 3

If (k < courseListing.size())

// as long as our iterator is smaller than the courseListing size keep looking for the course in the list

k++

else

// if k equals courseListing size and still no match has been found… Error out and return to the menu..

cout << “Course Prerequisite: “<<courseListing.course.at(i).coursePrerecs.at(j)<< “ doesn’t exist in the listing of courses” << endl;

return

// end iterator 3 meaning, the courseNumbers being compared match

// iterate iterator 2 until the end of prerecs list

// iterate iterator 1 until the end of courseListing

# courseQuickSort(vector<Course> courseList, int begin, int end)

// this is set up nearly identical to the quick sorting coding assignment that we did in module 2 using recursive calls to sort things out.

int p = 0

// if there isn’t anything in the courseList

if (begin >= end)

return

p = partitionCourses(courseList, begin, end)

quicksort(courseList, begin, p)

quicksort(courseList, p+1, end)

# partitionCourses(vector <Course> courseList, int begin, int end )

/\* this is set up nearly identical to the quick sorting coding assignment that we did in module 2 using recursive calls to sort things out\*/

int low = begin

int hi = end

//mid point string that is the courseNumber based on a computation using courseList size

string pivot = courseList[low+(hi-low)/2].courseNumber

while(low < hi)

//compare the alphanumeric values at the courseList at low’s Course number to that of Pivot. If its greater than pivot iterate to the next courseNumber.

while (courseList[low].courseNumber.compare(pivot) >0)

++low

// compare the alphanumeric values at the courseList at hi’s courseNumber to that of Pivot. If its less than pivot, iterate to the next courseNumber

while (courseList[hi].courseNumber.compare(pivot) < 0)

--hi

//Swap low and hi courseList

If ( low < hi )

Swap(courseList.at(low), courseList.at(hi))

++low

--hi

Return hi

# searchCourse(vector <Course> courseList, string criteria)

/\*this search should iterate through the course list looking for the search criteria given. If its found and has prerecs, iterate through the prerecs printing them as it goes. If it has no prerecs, state its got no prerecs, if the search criteria doesn’t yield any results, state nothing found. It’ll look something like this:

Course found

MAT123

Some mathclass

Course Prerequisites:

1+1 = yes

OR

Course has no prerequisites

OR

Course not found\*/

int i

For(i=0; i<courseList.size(); ++i) // iterator 1

// iterator to get us through the list of courses

If (courseList.at(i).courseNumber() == criteria)

cout << “Course Found!” << endl

cout <<courseList.at(i).courseNumber() <<endl

cout<< courseList.at(i).courseTitle << endl

if(courseList.at(i).coursePrerecs.size() > 0)

// if the course has prerecs

int j

cout << “Course Prerequisites:” << endl

for (j = 0, j < coursePrerecs.size(), j++) // iterator 2

cout << coursePrerecs.at(i) << endl;

//end iterator 2 once j hits coursePrerecs size

Else

// if there are no prerequisites

“Course has no prerequisites.”

Else

// if nothing matches the criteria

cout << “Course not found” << endl;

# main()

vector<Course> courseList // this creates a vector named courseList

String criteria, case, somePath

Int choice

clock\_t ticks

clock\_t tTicks

### //menu

while ( choice != 9 )// Menu iterator

cout << "Menu:" << endl

cout << " 1. Load Courses" << endl

cout << " 2. Get Course Information" << endl

cout << " 9. Exit" << endl

cout << "Enter choice: "

cin >> choice;

### case 1 Load courses

cout << “What file are you looking for?”<< endl

cin << somePath

ticks = clock()

tTicks = clock()

courseList = loadCourses(somePath)

cout << courseList.size() << “ Courses loaded.” << endl

cout << “time to load courses:” << ticks << “Clock ticks” << endl

cout << “time it took to load courses in seconds: << ticks\*1.0 / CLOCKS\_PER\_SEC << endl

//Check the list using QualityControl

ticks = clock()

QualityControl(courseList)

ticks = clock() – ticks

cout << “time to run QC:” << ticks << “Clock ticks” << endl

cout << “time it took to run QC in seconds: << ticks\*1.0 / CLOCKS\_PER\_SEC << endl

//sort our courseList using 0 and the size of the course list minus 1

ticks = clock()

courseQuickSort(courseList, 0, courseList.size()-1)

ticks = clock() – ticks

cout << “time to sort:” << ticks << “Clock ticks” << endl

cout << “time it took to sort in seconds: << ticks\*1.0 / CLOCKS\_PER\_SEC << endl

tTicks = clock() – tTicks

cout << “time to Load courses, QC them, and sort:” << tTicks << “Clock ticks” << endl

cout << “time it took to sort in seconds: << tTicks\*1.0 / CLOCKS\_PER\_SEC << endl

cout << “Courses Sorted”<< endl

break

### case 2 Get Course Information

cout << “Do you want to see all courses(a) or do you want to see just one course(b)?” << endl

getLine(cin, case)

#### if (case == “a”) // display all courses

printCourses(courseList)

#### if ( case == “b”) // search for a single course

cout << “Which course are you looking for? << endl;

getLine(cin, criteria))

cout << “searching for “ << criteria << endl;

searchCourses(courseList, criteria)

break

//end menu iterator

cout << “Goodbye!” << endl;