Joe Ducharme

Project One

**Resubmit pseudocode from previous pseudocode assignments and update as necessary**.

**Vector Data Structure**

Define a structure to hold the course data

Set a course id

set a course name

set the classes needed to take this class?

initialize the csv file for course

loop to read rows of the csv file

initialize a course using data from the current row

add this course to the end

private:

struct node

default contructor

node

next =nullptr

initialize a course

node Course aCourse

course = aCourse

next = nullptr

set head and tail

set size to 0

public

courseList

append,prepend, printlist, remove,search,int size.

Void course list

Make new node

If loop there is nothing at the head

Then the new node becomes the tail

Else the current tail will point to the new node, and it becomes a new node.

We will increase the size

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

for all courses

if the course is the same as courseNumber

print out the course information

for each prerequisite of the course

print the prerequisite course information

**Hashtable:**

Define a structure to hold the course data

Set course id

Set course title

Set prereq1

Set prereq2

Set a class for the hashtable

Private and public data

Default constructor :

Hashtable

Initaialize to hold courses

Constructor:

Hashtable (unsigned int size)

Invoke local tablesize

Resize the nodes size

Destructor:

Hashtable

Logic to free storage when a class is destroyed

Unsigned int hashtable (int key)

Implement logic to calculate a hash value

Void hashtalbe to insert a course

Void hashtable to print all courses

Void hashtable to remove a course

Course hashtable to search for the Course coursed

Void display (Course course)

Void loadCourse (string csvPath, hashtable\* hashTable)

Int main

This will produce the list of all course and print them out to the user.

**BST:**

//Struct course

//String course id, course title, courseprereq1, courseprereq2

//struct node

//default constructor

//initialize a course

// Binary Search Tree class definition

//default constructor

//default destructor

//traverse the tree in order

// traverse the tree in post-order

//traverse the tree in pre-order

//insert a course

//remove a course

//add node

//\*bst (Node\*node)\*//

//inorder

//postorder

//preorder

//remove node

//void display course

//for testing

//loading the csv files

//void loadcourse

//initialize the csv part

//make a loop to read the row of a csv file

//create a data structure and add to the collection

// Course course;

//course.courseId = file[i][0];

//course.courseTitle = file[i][1];

//course.coursePreRec1= file[i][2];

// course.coursePreRec2= file[i][3];

//print out id and title then check

//for loop to check prerec1 and print out

//for loop to check prerec2 and print out

//catch any error

**Create pseudocode for a menu**.

//int main

//have the csv path to the file

//print out a menu for the user.

// case 1 would load the courses within the csv file

//case 2 would print the courses using inorder

//case 3 would allow a search from the user to find a course, and print out

// case 9 will exit the program

**Design pseudocode that will print out the list of the courses in the Computer Science program in alphanumeric order.**

**For vector, we can use the quicksort method to sort the courses, then we can print the course out.**

**//void quick sort**

**// if begin is not equal to end return.**

**//mid would equal partition**

// recursively sort low partition (begin to mid)

// recursively sort high partition (mid+1 to end)

// print out sorted courses with id,title &prerec1 &prerec2

For hash tables there is no way to sort the courses alphanumeric. So we could only print these out how they are id within the csv.

For BST we can use “inorder” within the menu and this would print out the courses.

// /if node is not equal to null ptr

//InOrder not left

//output courseID, title, prereq1, prereq2

//InOder right

**Evaluate the run-time and memory of data structures that could be used to address the requirements**

Overall the only two we can use for ABCU are vector and BST. This means the biggest disadvantage for hash tables is it can’t be used for this type of program. Cost wise vector uses let lines of code and less functions to get it to run. But I feel like BST is the better option when adding n courses. I would say BST takes more time and lines to code, as we have to add all the functions for “inorder, pre-order, and post-order.

I think as a beginner I will code my ABCU program with vector. If I could figure out how to do BST I think that would be the better option.