Project one

By: Alex Roberts

**VECTOR:**

DECLARE courseList.

ifstream courseList;

OPEN FILE.

courseList.open;

While courseList = true

PRINT list

StoreFile

DECLARE numPrerequisiteCourses

totalPrerequisites = prerequisites of course c

for each prerequisite p in totalPrerequisites

add prerequisites of p to totalPrerequisites

print number of totalPrerequisites

PrintFile

INITIALIZE printSampleSchedule

INITIALIZE printCourseInformation

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

**HASH TABLE:**

**Reading File:**

Use fstream to be able to open file

Make call to open file

While it is not the end of file

Read each line

IF There are less than two values in a line, return ERROR

ELSE read parameters

IF there is a third or more parameter

IF third or more parameter is in first parameter elsewhere continue

ELSE return Error

Close file

**Create Course Objects HashTable:**

Initialize Course Vector nodes

Create HashTable Class

Create Insert method to insert items to HashTable

Loop through list

While not end of list

For each item in list

For first and second value

Create key to hold values

If a 3rd value exists

Add to current value

Call insert method for each value

**Search and Print from HashTable:**

Ask for Input

Assign input to key

If key is found

print out the course information

for each prerequisite of the course

print the prerequisite course information

**TREE:**

CREATE tree class

DECLARE courseList.

ifstream courseList;

OPEN FILE.

courseList.open;

While courseList = true

PRINT list

DEFINE struct

Struct Course

DECLARE courseID;

DECLARE title;

DECLARE preReq1;

DECLARE preReq2;

DECLARE preReqCount;

DECLARE Course()

INITIALIZE preReqCount = 0;

DEFINE internal structure of Node

DECLARE Course;

DECLARE Node;

INITIALIZE node;

DECLARE method Node();

INITIALIZE < to null pointer;

INITIALIZE > to null pointer;

PRINT:

Cout << courseList << endl;

**MENU:**

LOAD courseList into data structure

PRINT courseList

CourseID

CourseTitle

Loop for alphabetical order

PRINT course

CourseID

CourseTitle

Loop for alphabetical order

INCLUDE preReq

EXIT course

CREATE orderedList function

SORT courseTitle alphabetically

LOOP lowest to highest

PRINT orderedList

Vector: The total cost to run is 6n and the runtime is 1(N)

Hash Table: The total cost to run is 9n and the runtime is 0(N)

Tree: The total cost to run is 8n and the runtime is 0(N)

Based on the results, vector had the lowest cost and the runtime was the fastest. Vector would be the best suit for the job due to this. Hash table had the biggest cost and hash table and tree had the same runtime. Tree would be the better choice than the hash table even though they had the same runtime the tree’s cost was sightly lower. Vector is the victor here as the cost and runtime were the best.