Matthew Haggard

CS 300

Project 1 Submission

# Pseudocode

// Vector pseudocode

int numPrerequisiteCourses(Vector<Course> courses, Course c) {

totalPrerequisites = prerequisites of course c

for each prerequisite p in totalPrerequisites

add prerequisites of p to totalPrerequisites

print number of totalPrerequisites

}

void printSampleSchedule(Vector<Course> courses) {

declare schedule variable

for each course

if course does not have preReq or if preReq has been taken

add course to schedule

}

**// Data structure to hold course info**

**Struct Course{**

**String courseId**

**String courseTitle**

**String coursePreReq**

**}**

**// Vector load courses pseudocode**

**Vector<Course> loadCourses(string filePath){**

**Define vector data structure to hold courses**

**Initialize the parser using the file path**

**Try{**

**For each row in the file**

**Create data structure and add to the courses vector**

**Push the course to the end of the “list”**

**} catch(error)**

**Return course;**

**// Print course info**

**void displayCourse(Course course) {**

**Print course.courseId : course.courseTitle : course.coursePreReq**

**return;**

**}**

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 pseudocode

int numPrerequisiteCourses(Hashtable<Course> courses) {

declare variable numPreReqs of int type and initialize to 0

for each prerequisite

increment numPreReqs by 1

return numPreReqs

}

void printSampleSchedule(Hashtable<Course> courses) {

declare schedule variable

for each course

if course does not have preReq or if preReq has been taken

add course to schedule

}

void printCourseInformation(Hashtable<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

}

**// Data structure to hold course info**

**Struct Course{**

**String courseId**

**String courseTitle**

**String coursePreReq**

**}**

**// Hashtable load courses pseudocode**

**Void loadCourses(string filePath, Hashtable\* hashTable){**

**Initialize the parser using filepath**

**loop to read rows of a CSV file**

**Create a data structure and add to the collection of bids**

**push this bid to the end**

**}**

**// Print course info**

**void displayCourse(Course course) {**

**Print course.courseId : course.courseTitle : course.coursePreReq**

**return;**

**}**

// Tree pseudocode

int numPrerequisiteCourses(Tree<Course> courses) {

new int numPreReqs equals 0

for all preReqs

numPreReqs += 1

return numPreReqs

}

void printSampleSchedule(Tree<Course> courses) {

declare schedule variable

for each course

if course does not have preReq or if preReq taken

add course to schedule

}

void printCourseInformation(Tree<Course> courses, String courseNumber)

{

For all courses

IF course equals courseNumber

Cout course information

For each preReq

Cout preReq information

}

**// Data structure to hold course info**

**Struct Course{**

**String courseId**

**String courseTitle**

**String coursePreReq**

**}**

**// Tree load courses pseudocode**

**Void loadCourses(string filePath, BinarySearchTree\* bst){**

**Initialize the parser using filepath**

**loop to read rows of a CSV file**

**Create a data structure and add to the collection of bids**

**push this bid to the end**

**}**

**// Print course info**

**void displayCourse(Course course) {**

**Print course.courseId : course.courseTitle : course.coursePreReq**

**return;**

**}**

**// Main menu**

**Int main(){**

**Declare choice variable of int type**

**While choice != 1, 2, 3{**

**Print menu << endl**

**Print Load courses << endl**

**Print Display All Courses << endl**

**Print Course Info << endl**

**Switch statement for all avail choices**

**Case 1:**

**Call loadCourses()**

**Break**

**Case 2:**

**For loop for displayCourse[i]**

**Break**

**Case 3:**

**displayCourse()**

**Break**

**}**

Some advantages of using vectors in this project are the ease of use and ease of access to the elements. A big disadvantage of vectors is the insertion and deletion of the elements in the middle of the vector. If an element is deleted somewhere in the middle of the vector, every item after that element must be modified.

Hash tables have the advantage of search, insert, and delete being fast and they are able to store large amounts of data. Some issues can arise with using hash tables, however. Creating duplicate keys and causing collisions can be implemented easily and ensuring distinct keys can be somewhat expensive and difficult to implement.

Trees are great for organization and printing sorted results. The problem with trees is that they can take a few extra steps to modify, especially when trying to remove entries with children.

For this project, I would likely use a vector due to the data size and the ease of use when writing the code. Any speed deficiencies that would be caused by adding and deleting items in the vector would be virtually negligible with such a small set of data (7 entries according to the course information sheet).