Victor Jarvis

CS 300

Project One

**Vector Data Structure Pseudocode**

**Class Course**

int courseNum

string courseName

vector string prereqs

**Search**

Course search (vector Course, courses, string courseNum)

create empty course

for each course in courses

IF current courseNum equals courseNum

Return course

RETURN empty

**Read File**

vector string OpenReadFile (string file)

INITIALIZE vector string

INITIALIZE string to hold single line

INITIALIZE fstream to get file contents

open file using file name

IF unable to open

OUTPUT error message

pull line from instream until data is received

push line to the back

close the file

RETURN data

**Create Course Schedule**

vector course createSchedule (vector string materials)

INITIALIZE vector course\*

INITIALIZE string stream linestream

INITIALIZE string token to store words from each line

INITIALIZE counter

for each new course

set the counter to 0

create a new course

fill linestream with line materials

pull token from linestream until the end

IF the counter equals 0

course courseNum equals the token

increment counter

ELSE IF counter equals 1

courseNum equals token

increment counter

ELSE

IF token equals courseNum is already in courses

push the token to the back of course prereqs

ELSE

output error message

increment counter

IF the counter is greater than 2

OUTPUT error message

empty line stream

push course to the back of courses

RETURN courses

**Print**

void print (vector course courses, string courseNum)

create a course object

IF course is empty

OUTPUT error message

RETURN

OUTPUT course courseNum and courseName

for each prereq in prereqs

OUTPUT prereq

**Hash Table Pseudocode**

**Read File**

Open file with fstream

Call file open method

IF value is -1

Return error message

ELSE

Parse each line

IF less than two values per line

RETURN error

ELSE read

IF 3 or more values

IF in first value

Continue

ELSE

RETURN error

Close

**Creating Course Objects**

INITIALIZE Course vector nodes

CREATE hash table class

WHILE loop

WHILE not at the end of the file

For each line

For each first and second value

Create temp item

IF third value

Add to current

Call on insert method

**Search and Print**

Get user input

Assign user input to a key

IF the key is found

Print the information

For each prerequisite

Print the prerequisite

**Tree Data Structure Pseudocode**

**Read File**

Use fstream to open file

Call open file

IF return is “-1”

RETURN error message

ELSE WHILE not the end of the file

Parse each line

IF less than two values

RETURN error message

ELSE read

IF three or more params

IF third or greater param is in first param

Continue

ELSE

RETURN error message

Close file

**Creating Course Objects**

INITIALIZE Course Structure struct course

Loop through the file

WHILE not at end of the file

For each line

For first two values

Add ID and name

IF third value

Add prerequisites until new line

**Creating Tree, Adding Nodes**

Define BST class

Create a root, pointed to null

Create insert method

IF root equals null

Current course is root

ELSE IF course number is less than root

Add left

IF left is null

Add course number

ELSE

IF course num is less than leaf

Add left

IF course num is greater than leaf

Add right

ELSE IF course num is greater than root

Add right

IF right equals null

Add course num

ELSE

IF course num is less than leaf

Add left

ELSE IF course num is greater than leaf

Add right

**Print/Search**

Get user input

Create print

IF root not null

Traverse left

IF found

OUTPUT

Traverse right

IF found

OUTPUT

**Menu Pseudocode**

Set user input int for switch case to 0

Create new bid variable

WHILE input does not equal 4

OUTPUT Load Data Structure

OUTPUT Print Course List

OUTPUT Print Course

OUTPUT Exit

OUTPUT user input message (“select an option”)

Switch case (input int)

Case 1 load up bids

Break

Case 2 print course list

Break

Case 3 print course

Break

Case 4 end process

Break

**Alphanumeric Pseudocode**

Create string to sort

Create variable to add 1 to length/size

Create string to char array

Sort array

Set int for alpha

Set int for numeric

WHILE loop

IF alpha less than size

Increment alpha

IF I less than size

Increment numeric

RETURN sort string

Create class string

Print

**Evaluations**

**Vector**

Line Costs:

* All courses: 1
* Matches courseNum: 1
* Course Information: 2
* Prerequisites: 1
* Prerequisite Information: 2

Execute Times:

* All courses: n
* Matches courseNum: n
* Course Information: 1
* Prerequisites: n
* Prerequisite Information: n

Total cost: 6n + 1

Runtime: 1(n)

**Hash Table**

Line Costs:

* All courses: 2
* Matches courseNum: 1
* Course Information: 1
* Prerequisites: 2
* Prerequisite Information: 4

Execute Times:

* All courses: n
* Matches courseNum: n
* Course Information: 1
* Prerequisites: n
* Prerequisite Information: n

Total cost: 9n + 1

Runtime: O(n)

**Tree**

Line Costs:

* All courses: 1
* Matches courseNum: 1
* Course Information: 2
* Prerequisites: 1
* Prerequisite Information: 4

Execute Times:

* All courses: n
* Matches courseNum: n
* Course Information: 1
* Prerequisites: n
* Prerequisite Information: n

Total cost: 8n + 1

Runtime: O(n)

The main advantage of the vector structure may also be its main disadvantage in its own way. As it’s one dimensional, it handles our data in a much more efficient way with respect to runtime. However, with this advantage, and the potential for more memory, the big disadvantage is the inability to delete elements, or handle more than a single data type. Using a hash table allows for more organization and storage capabilities. The use of a key also allows for easier calling throughout the structure. Its main disadvantage is that it has a much longer runtime as it allows for processes such as creating, deleting, and calling within the structure. Using a tree is also a longer runtime structure, but it affords much better organization. Despite requiring a longer runtime, I would personally prefer to utilize the hash table, as it allows for more functionality now, and additional functionality in the future if more edits to the application are made.