# CS 300 Pseudocode Document

## Function Signatures

Below are the function signatures that you can fill in to address each of the three program requirements using each of the data structures. The pseudocode for printing course information, if a vector is the data structure, is also given to you below (depicted in bold).

**// Vector pseudocode**

Void openFile {

Open file;

set prerequisites to variables to

}

int Node {

int courseNumber;

int numPrerequisites;

string prerequisites;

}

Void printSchedule {

}

int prerequisites; {

**Call string prerequisites**

**Loop that check if prerequisites have been met**

**Return true or false**

}

void sampleSchedule {

**call outcome from prerequisites function.**

}

**// Hashtable pseudocode**

int numPrerequisiteCourses(Hashtable<Course> courses) {

if loop numPrerequisite = key

completed = true

if loop numPrerequisite != key

completed = false

if true

external file schedule -> node

}

void printSampleSchedule(Hashtable<Course> courses) {

output table that now include external file

}

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

call numPrerequisite

output table that has completed external file

}

**// Tree pseudocode**

int numPrerequisiteCourses(Tree<Course> courses) {

}

void printSampleSchedule(Tree<Course> courses) {

}

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

}

**Runtime Analysis Vector**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for all courses** | 1 | 1 | 1 |
| **if the course is the same as courseNumber** | 1 | 5 | 5 |
| **print out the course information** | 1 | 1 | 1 |
| **for each prerequisite of the course** | 1 | 6 | 6 |
| **print the prerequisite course information** | 1 | 6 | 6 |
| **Total Cost** | | | 4n + 1 |
| **Runtime** | | | O(n) |

**Runtime Analysis Hash Table**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for all courses** | 1 | 3 | 3 |
| **if the course is the same as courseNumber** | 1 | 3 | 3 |
| **print out the course information** | 1 | 4 | 4 |
| **for each prerequisite of the course** | 1 | 4 | 4 |
| **print the prerequisite course information** | 1 | 10 | 10 |
| **Total Cost** | | | 4n + 1 |
| **Runtime** | | | O(n) |

**Runtime Analysis Tree**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for all courses** | 1 | 3 | 3 |
| **if the course is the same as courseNumber** | 1 | 3 | 3 |
| **print out the course information** | 1 | 1 | 1 |
| **for each prerequisite of the course** | 1 | 4 | 4 |
| **print the prerequisite course information** | 1 | 4 | 4 |
| **Total Cost** | | | 4n + 1 |
| **Runtime** | | | O(n) |

**Advantages and Disadvantages**

With the information from the Runtime Analysis My recommendations would be the tree data structure. The tree data structure from the pseudocode above has less lines that need to be parsed and ran through making it more efficient and less time is spent going reparsing lines of code when the functions are called. With the hash table although simpler in how it can be laid out is more costly due foe the need of the multiple loops that add more time since those lines need to be evaluated every time. With the vector data structure its an even mix of both with more lines and functions but less repeating loops that need to be evaluated leading to less time spent and a timelier runtime than Hash table but less than the tree data structure.