# CS 300 Pseudocode Document

**//Vector - Milestone 1**

// Define a Course Object

class Course {

String courseNumber

String courseName

List prerequisites

// Constructor to create a course

Course(String courseNumber, String courseName, List prerequisites) {

this.courseNumber = courseNumber

this.courseName = courseName

this.prerequisites = prerequisites

}

}

// Open and Read File for Vector

void readFile(String fileName, Vector<Course> courses) {

file = open(fileName)

if file is null:

print("Error: File not found or could not be opened.")

return

for line in file:

data = split(line, ',')

if length(data) < 2:

print("Error: Invalid format in file.")

continue

courseNumber = data[0]

courseName = data[1]

prerequisites = []

if length(data) > 2:

prerequisites = data[2:]

course = Course(courseNumber, courseName, prerequisites)

courses.add(course) // Add to vector

close(file)

}

// Search and Print Course Information from Vector

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

for course in courses:

if course.courseNumber == courseNumber:

print("Course Number: " + course.courseNumber)

print("Course Name: " + course.courseName)

if length(course.prerequisites) == 0:

print("Prerequisites: None")

else:

print("Prerequisites: " + join(course.prerequisites, ", "))

return

print("Error: Course " + courseNumber + " not found")

}

// Print All Courses in Alphanumeric Order (Vector)

void printCourseList(Vector<Course> courses) {

sort(courses by courseNumber)

for course in courses:

print(course.courseNumber + ": " + course.courseName)

}

// Menu for User Interaction

void menu(Vector<Course> courses) {

while true:

print("1. Load courses from file")

print("2. Print all courses (alphanumeric order)")

print("3. Search for a course and display its details")

print("9. Exit")

choice = input("Enter your choice: ")

if choice == "1":

fileName = input("Enter the file name: ")

readFile(fileName, courses)

elif choice == "2":

printCourseList(courses)

elif choice == "3":

courseNumber = input("Enter course number to search: ")

searchCourse(courses, courseNumber)

elif choice == "9":

break

else:

print("Invalid choice, please try again.")

}

**//Hash Table - Milestone 2**

// Define a Course Object (Same as Vector)

class Course {

String courseNumber

String courseName

List prerequisites

Course(String courseNumber, String courseName, List prerequisites) {

this.courseNumber = courseNumber

this.courseName = courseName

this.prerequisites = prerequisites

}

}

// Open and Read File for Hash Table

void readFile(String fileName, HashTable<String, Course> courses) {

file = open(fileName)

if file is null:

print("Error: File not found or could not be opened.")

return

for line in file:

data = split(line, ',')

if length(data) < 2:

print("Error: Invalid format in file.")

continue

courseNumber = data[0]

courseName = data[1]

prerequisites = []

if length(data) > 2:

prerequisites = data[2:]

course = Course(courseNumber, courseName, prerequisites)

courses.put(courseNumber, course) // Add to hash table

close(file)

}

// Search and Print Course Information from Hash Table

void searchCourse(HashTable<String, Course> courses, String courseNumber) {

if not courses.containsKey(courseNumber):

print("Error: Course " + courseNumber + " not found")

return

course = courses.get(courseNumber)

print("Course Number: " + course.courseNumber)

print("Course Name: " + course.courseName)

if length(course.prerequisites) == 0:

print("Prerequisites: None")

else:

print("Prerequisites: " + join(course.prerequisites, ", "))

}

// Print All Courses in Alphanumeric Order (Hash Table)

void printCourseList(HashTable<String, Course> courses) {

courseList = courses.values()

sort(courseList by courseNumber)

for course in courseList:

print(course.courseNumber + ": " + course.courseName)

}

// Menu for User Interaction (Same as Vector)

void menu(HashTable<String, Course> courses) {

while true:

print("1. Load courses from file")

print("2. Print all courses (alphanumeric order)")

print("3. Search for a course and display its details")

print("9. Exit")

choice = input("Enter your choice: ")

if choice == "1":

fileName = input("Enter the file name: ")

readFile(fileName, courses)

elif choice == "2":

printCourseList(courses)

elif choice == "3":

courseNumber = input("Enter course number to search: ")

searchCourse(courses, courseNumber)

elif choice == "9":

break

else:

print("Invalid choice, please try again.")

}

**// Binary Search Tree – Milestone 3**

// Define a Course Object (Same as Vector)

class Course {

String courseNumber

String courseName

List prerequisites

Course(String courseNumber, String courseName, List prerequisites) {

this.courseNumber = courseNumber

this.courseName = courseName

this.prerequisites = prerequisites

}

}

// Binary Search Tree Node

class TreeNode {

Course data

TreeNode left

TreeNode right

TreeNode(Course data) {

this.data = data

this.left = null

this.right = null

}

}

// Insert Course into Binary Search Tree

void insertCourse(TreeNode root, Course course) {

if course.courseNumber < root.data.courseNumber:

if root.left is null:

root.left = TreeNode(course)

else:

insertCourse(root.left, course)

else:

if root.right is null:

root.right = TreeNode(course)

else:

insertCourse(root.right, course)

}

// Open and Read File for Binary Search Tree

void readFile(String fileName, TreeNode root) {

file = open(fileName)

if file is null:

print("Error: File not found or could not be opened.")

return

for line in file:

data = split(line, ',')

if length(data) < 2:

print("Error: Invalid format in file.")

continue

courseNumber = data[0]

courseName = data[1]

prerequisites = []

if length(data) > 2:

prerequisites = data[2:]

course = Course(courseNumber, courseName, prerequisites)

insertCourse(root, course) // Insert into binary search tree

close(file)

}

// Search and Print Course Information from Binary Search Tree

void searchCourse(TreeNode root, String courseNumber) {

if root is null:

print("Error: Course " + courseNumber + " not found")

return

if root.data.courseNumber == courseNumber:

print("Course Number: " + root.data.courseNumber)

print("Course Name: " + root.data.courseName)

if length(root.data.prerequisites) == 0:

print("Prerequisites: None")

else:

print("Prerequisites: " + join(root.data.prerequisites, ", "))

return

elif courseNumber < root.data.courseNumber:

searchCourse(root.left, courseNumber)

else:

searchCourse(root.right, courseNumber)

}

// Print All Courses in Alphanumeric Order (In-order Traversal)

void printCourseList(TreeNode root) {

if root is not null:

printCourseList(root.left)

print(root.data.courseNumber + ": " + root.data.courseName)

printCourseList(root.right)

}

// Menu for User Interaction (Same as Vector)

void menu(TreeNode root) {

while true:

print("1. Load courses from file")

print("2. Print all courses (alphanumeric order)")

print("3. Search for a course and display its details")

print("9. Exit")

choice = input("Enter your choice: ")

if choice == "1":

fileName = input("Enter the file name: ")

readFile(fileName, root)

elif choice == "2":

printCourseList(root)

elif choice == "3":

courseNumber = input("Enter course number to search: ")

searchCourse(root, courseNumber)

elif choice == "9":

break

else:

print("Invalid choice, please try again.")

}

## 

## Runtime Analysis

| **Data Structure** | **Operation** | **Time Complexity** | **Space Complexity** |
| --- | --- | --- | --- |
| **Vector** | Search course | O(n) | O(n) |
|  | Insert Course | O(1) | O(n) |
|  | Print Course List | O(n log n) | O(n) |
| **Hash Table** | Search course | O(1) (average case) | O(n) |
|  | Insert Course | O(1) (average case) | O(n) |
|  | Print Course List | O(n log n) | O(n) |
| **Binary Search Tree** | Search course | O(log n) (balanced) | O(n) |
|  | Insert Course | O(log n) (balanced) | O(n) |
|  | Print Course List | O(n) | O(n) |