**Sabastian Fasano**

**CS-300**

**February 9, 2025**

**Module 5 Milestone: Pseudocode for ABCU Project One**

**Load** text parsing libraries and headers

**Define** a struct to hold course data

**struct Course {**

**string courseID;**

**string courseName;**

**int preCount; string preList;**

**Course() {**

**courseID = courseName = "";**

**preCount = 0;**

**preList = "";**

**}**

**}**

**Class BinaryTree**

* **struct Node**
  + **Course**
  + **Right pointer**
  + **Left pointer**
* **root**
* **+printCourse()**
* **+BinaryTree()**

**Main()**

* **Create a new BinaryTree named courseTree of the struct-type Course**
* **Get the CSV file path from the user**
  + **If no data is provided, use the default location**
* **Call txtParser() and pass the CSV file path**
* **Call validateList() and pass courseTree**
* **Get the user’s search value and store it in userSearch**
* **Call printCourse() and pass userSearch**
* **End**

**txtParser (String)**

* **Open the file located at the path given in the string by invoking the text parsing libraries**
* **Loop through each row until the end of the file (EOF)**
  + **If both the first and second strings are present:**
    - **Add the first string to courseID in the struct**
    - **Add the second string to courseName in the struct**
    - **Loop until there is no value in a column (indicating no more prerequisites)**
      * **Increment a variable named preCount for each prerequisite found**
      * **Concatenate the prerequisite names into a local string called preNames**
    - **Add preCount to the struct**
    - **Add preNames to the struct**
* **Return tempList**

**searchList(String)**

* **Create a temporary Course object of type Node**
* **Set the tempCourse to the bucket at the hash location of the input string**
* **Loop through the list for each course**
  + **If the courseID matches the string:**
    - **Set tempCourse to the course**
* **Return tempCourse**

**printCourse(String)**

* **Create a tempCourse of type Node**
* **Set tempCourse equal to the root**
* **Loop until tempCourse is Null:**
  + **If the Node at tempCourse contains a courseID matching the input string:**
    - **Output the courseID from the Course struct in tempCourse to the console**
    - **Output the courseName from the Course struct in tempCourse to the console**
    - **Loop from 0 to preCount:**
      * **For each course in preList, call printCourse() passing the prerequisite**
  + **If the Node at tempCourse contains a courseID less than the string:**
    - **Set tempCourse equal to the left Node**
  + **If the Node at tempCourse contains a courseID greater than the string:**
    - **Set tempCourse equal to the right Node**

**validateList()**

* **Create a temporary Course object of type Node**
* **Create a variable valid and set it to True**
* **For each course:**
  + **If valid is False, break the loop**
  + **While tempCourse next is not Null:**
    - **Loop from 0 to preCount:**
      * **Set tempCourse equal to searchList(preList token)**
      * **If tempCourse.courseID is empty, set valid to False**
* **Return valid**

**End**