Dylan Lovin

SNHU CS-300

2/2/2025

### **Module 4 Milestone: Pseudocode for ABCU Project One**

#### **Load Required Libraries and Headers**

* Import necessary libraries for text parsing.

#### **Define Course Structure**

* Create a Course struct to store course details:
  + courseID (string)
  + courseName (string)
  + preCount (integer) – stores the number of prerequisites
  + preList (string) – stores prerequisite course IDs
* **Constructor:**
  + Initialize courseID and courseName as empty strings.
  + Set preCount to 0 and preList to an empty string.

### **Class: HashTable**

* **Struct: Bucket**
  + Contains a Course instance.
  + Holds a key.
  + Has a pointer to the next node (for handling collisions).
* **Methods:**
  + hash() – Computes the hash key for a given course.
  + printAll() – Displays all courses in the hash table.
  + List<> hashTable – Stores hashed course data.

### **Main Function**

* Create a courseList (a list of Course objects).
* Prompt the user for the CSV file path.
* If no input is provided, use the default path.
* Call txtParser() with the CSV file path to load course data.
* Call validateList() to verify course integrity.
* Prompt the user for a course search query and store it in userSearch.
* Call printCourse(userSearch) to display the course details.

### **txtParser(String filePath)**

* Open the file located at filePath using text parsing functions.
* Read through the file, row by row, until the end is reached.
* If the first and second values exist in a row:
  + Call hash() on the first value.
  + Store the first value (courseID) in a temporary structure at the computed hash position.
  + Store the second value (courseName) in the same structure.
* Process prerequisite courses:
  + Continue reading columns until an empty value is encountered.
  + Increment preCount for each prerequisite found.
  + Append each prerequisite to a local string preNames.
* Store the total prerequisite count and names in the structure.
* Return the populated temporary list.

### **searchList(String courseID)**

* Create tempCourse, a variable of type Bucket.
* Set tempCourse to the bucket at the hashed location of courseID.
* Iterate through the linked list at that location:
  + If courseID matches a stored course, assign it to tempCourse and return it.

### **printCourse(String courseID)**

* Create tempCourse of type Bucket.
* Assign tempCourse to the result of hash(courseID).
* Traverse any chained buckets at tempCourse.
* Display courseID and courseName from tempCourse.
* If the course has prerequisites:
  + Loop through preList and recursively call printCourse() for each prerequisite.

### **validateList()**

* Create tempCourse of type Bucket.
* Set a valid flag to True.
* Iterate through all courses:
  + If valid becomes False, exit the loop.
  + While tempCourse->next exists:
    - Loop through each prerequisite in preList.
    - Use searchList(preList token) to find the prerequisite.
    - If no matching course is found, set valid to False.
* Return whether the course list is valid.

### **int Hash(String key)**

* **Considerations for Hashing Function:**
  + The first four letters (e.g., "CSCI") frequently repeat for courses in the same major.
  + The last three digits may not be unique across different departments (e.g., CSCI100 vs. MATH100).
* Compute a hash value for key based on these considerations and return it.