**Project One Milestone Two: Hash Table Data Structure**

Justin Starr

Department of STEM

CS 300 – DSA: Analysis and Design

Professor Ricardo Scarello

June 2, 2024

**Project One Milestone Two: Hash Table Data Structure**

// File handling

**void parseFile(string csvPath)** {

OPEN csvFile

IF csvFile found && csvFile size is not zero {

READ from input next line of csvFile and parse to file

IF line has less than two parameters {

DISPLAY message, the course cannot be added

} END IF

ELSE {

IF line is greater than or equal to two parameters and course pre-requisite

is already in file {

CREATE new course object

LOOP FOR parameters in the line {

ASSIGN Course attributes for each parameter in the line

} END FOR LOOP

ADD the new line to the Course data structure

} END IF

} END ELSE

ELSE csvFile size is zero {

DISPLAY error message file could not be opened

} END ELSE

CLOSE csvfile

} END of parseFile function

// Load a CSV file containing courses into a container

// csvPath is the path to the CSV file to load

// function returns a container holding all of the courses read.

**void loadCourses(string csvPath, HashTable\* hashtable)** {

INITIALIZE the CSV Parser using the given path

TRY {

FOR all rows in the csvFile {

IF number of parameters in row is greater than two {

CREATE a new course data structure.

ASSIGN the new course attributes from csv file

INSERT the new course to the Courses data structure (hashTable)

} END IF

ELSE number of parameters in row is less than two {

throw error message "course does not have enough parameters to

be added"

} END ELSE

} END FOR LOOP

} END TRY

CATCH (csv::Error& e){

// An error is caught when there are not at least two parameters on a line of

// the csvFile

// An error is caught when any prerequisite at the end of a line does not have

// another line in the csvFile that starts with that course number

DISPLAY any errors encountered reading the csv file

}END CATCH

} END of loadCoarses function

// function for searching for a course by course number

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

CREATE the key for the given course

ASSIGN node by trying to retrieve the node using the key

// if a searched course is not found, return it

IF no entry exists for the key {

RETURN the course

} END IF

IF the entry is found for the key {

DISPLAY course information

FOR each prerequisite of the course {

CREATE tempKey for the prerequisite

ASSIGN tempNode by trying to retrieve the node using the tempKey

IF the tempKey is found for the tempKey {

DISPLAY course information

} END IF

} END FOR LOOP

} END IF

} END function