**CS/IT 200**

# Lab 11: Transcript Manager

**Due Date**: Tuesday, November 26, 11:59pm

**Submission**

- Submit all code and other materials in a single zip file to the Lab 11 assignment folder on Kodiak.

**Goal:** Select and combine several data structures to construct a system that allows a user to view and edit a student's transcript.

## Introduction

Your goal for this assignment is to create a system that allows a user to edit and look up information about a student's transcript. Unlike prior labs, you are solely responsible for determining what ADTs and data structures to use in solving this problem.

A transcript is made up of courses. At minimum, courses have the following attributes (with example values in parentheses): Course code (CS 200), course name (Data Structures), credits (4), grade (A-), semester taken (Fall 2019), and instructor (Dr. Moulema). For the sake of this assignment, you may assume that course codes are unique.

## Problem

The transcript manager must have the following functions, which users can interact with through a simple command-line menu interface:

- Lookup course – Given a unique attribute of a course, show all attributes of that course
- *Add course – Add a course to the transcript. If the course is already present, reject the new addition (make no changes).
- *Update course – Given a unique attribute of a course, edit a different attribute of that course. The user must be given the opportunity to select which attribute should be modified. (Should be able to choose between multiple attributes, but it is not necessary to be able to modify all attributes.)
- *Delete course – Given a unique attribute of a course, delete that course from the transcript
- Find – Find all courses with a given attribute
  - Must be implemented for at least 3 attributes, all of which must be non-unique.
  - Example: Find all courses with a particular instructor, or a particular credit value.
- Show all – Show all attributes of all courses, in a table
- Calculate GPA – Shows the GPA based on the courses in the transcript
  - For guidance on calculating GPA, see this website.

- Calculate Field GPA – Given a department acronym (e.g. CS or MATH or IT), calculate the GPA using only courses that include that acronym
- Undo – Undoes the last action. Only applies to actions marked with an asterisk above. Must be able to undo multiple actions in a row. If the last action was not undoable (like "Show all") then you should undo the last action that can be undone.
- Redo – Redo the last action that was undone.
    - Note: You can only redo an action when the last action was an undo or a redo, and when there are actions that can be redone. If the last action was not an undo or redo, this should do nothing.
- Save transcript to file – Should prompt the user for a filename and save this data in a text file.
- Load transcript from file – Loads a saved transcript file into the system. Before loading a transcript from a file, a user is working with an empty transcript.

**What You Can Use**

You can use any of the data structures shown in class or in the textbook. (This code is available on Kodiak and Piazza.) Additionally, you can use the following built-in Python data structures: Lists [], tuples (), dictionaries {}, and sets. Use of any other Python module will be penalized heavily.

**Write-up**

You must include a thorough write-up along with your code. This write-up must address the following questions:

- What abstract data type(s) and data structure(s) do you use in your code?
- How do you use these data structure(s)? Be specific about their task(s) in the overall system. That is, if you use a priority queue for the undo/redo actions, say that that is their purpose. (Hint: Do not use priority queues for undo/redo.)
- Why did you choose those data structure(s) for those task(s)? Why were they the best choices? What are the running times for the required functionalities given that you selected these data structure(s)?
- What alternatives did you consider?

Even if your code does not work for some of the functions above, you should do your best to include answers to these questions for those functions, so that we can at least see your thought process and design concepts.

**Grading**

Your code will be graded on a 50-point scale, largely based on the twelve functions listed above. Data structure choice is not part of the code grade, unless the choice impacts functionality (either by not working or by being noticeably inefficient). Commenting will make up approximately 15% of this grade.

Your write-up will also be graded on a 50-point scale. The majority of this grade is based on your answers to the questions above, though spelling and grammar may affect the grade partially. The following is a rough rubric for the write-up:

- 50: Answers all questions above about all aspects of the submission. Reasonable data structure(s) were selected for all aspects of the assignment. Statements about data structures advantages/disadvantages are entirely correct.
- 45: Answers contain 1-2 errors OR some answers lack sufficient detail.
- 43: Answers contain 1-2 errors AND some answers lack sufficient detail.
- 40: An unreasonable data structure was used for part of the assignment, but the logic of answers is otherwise excellent.
- 35: An unreasonable data structure was used for part of assignment, and several answers contain flaws.
- 30: Most data structures selected were not reasonable, or answers were largely incomplete.
- 25: All data structures selected were unreasonable choices, or the answers failed to address most questions.
- 0: No submission, or a submission that is unreadable and/or irrelevant.

Your final grade for this lab = Code Grade + Write-up Grade

**What to Submit:**

- Your code (including **<u>all necessary modules</u>**)
- Your write-up (in Word, plaintext, or PDF format)