**Practicing 2D lists by Designing a Hiring Algorithm**

Exercise originally by Evan Peck (Bucknell) for CSCI 203

Modified by Amy Csizmar Dalal (Carleton) for CS 111

Imagine you are working for *Moogle,* for a well-known tech company that receives tens of thousands of job applications from graduating seniors every year.

Since the company receives too many job applications for HR to individually assess in a reasonable amount of time, you are asked to create a program that algorithmically looks at applications and selects the ones most worth looking at (and passing onto HR).

It’s difficult to create these first-pass cuts, so *Moogle* designs their application forms to get some numerical data about their applicants’ education. Job applicants must enter the grades they received in 6 core CS courses as well as their overall GPA. This is stored in a list which you can access. For example, a student who received the following scores in classes:

* **Intro to CS (0 index):** 100
* **Data Structures (1 index):** 95
* **Software Design (2 index):** 80
* **Algorithms (3 index):** 89
* **Computer Organization and Architecture (4 index):** 91
* **Operating Systems (5 index):** 75
* **Overall *College* GPA (6 index):** 83

Would result in the following list: [100, 95, 80, 89, 91, 75, 83]

You can assume the 0th index is always intro to CS, 1st is always Data Structures, etc. Because you are processing many applicants, you receive a *list of lists*. For example, this would the information for 3 applicants:

­[ [100, 95, 80, 89, 91, 75, 83], [75, 80, 85, 90, 85, 88, 90], [85, 70, 99, 100, 81, 82, 91] ]

Your job is to

1. Determine how you are going to select the top applicants to pass onto HR
2. Given a list of applicant data (a list of lists), write a function that returns a new list of worthwhile candidates.

I’ve provided you with hiring.py that includes a sample list of 10 applicants. See if you can get that working 1st. After that, try using the data from allApps.py (changing exampleList to lottaApps in your code), which contains a list of approximately 5000 applicants!

As you develop your function, examine the results, and refine your algorithm, jot down notes along the way. What were your initial assumptions? How did these assumptions evolve over the course of development?

Compare your generated top candidates with your neighbor’s top candidates. Did you choose similar candidates? Where did your discrepancies lie? How did your assumptions impact your results?

At the end of class, I will give you some specific scenarios to consider, but in the meantime you might want to think about “corner cases” that your algorithm might miss.