**COSC 1336 Exam 1 Lab**

Python lab for Exam 1 (35 points). There are 2 parts.

**Part 1**. (20 points)

For this part, you will create a new Python program from scratch. Please follow these steps:

1. Start IDLE and save a new file locally as: DDHH\_X1\_Lastname.py. (Use the course designator for DDHH and your last name.)
2. Add a comment at the top with your name and course (COSC1336).
3. Write Python code that asks the user for an ACC campus abbreviation.
4. Accept any one of the following inputs: **NRG**, **RRC**, **RGC,** **CYP, HLC, Q** (to quit)
5. If the user enters ***something other than the above***, ***display an error message*** and ask again for the campus abbreviation. (Loop back to step 3.)
6. If provided a valid campus abbreviation, ask the user for:
   1. Number of buildings on that campus
   2. Number of classrooms per building
   3. Number of seats per classroom
7. Compute the maximum capacity for that campus. (How many students can attend?)
8. Format output to look like this: **ACC’s NRG campus can serve up to 1,344 students.**
9. Print the output to the screen. Use format() so numbers like 1344 appear as 1,344.
10. Provide a loop to enter campus data until the user enters: **Q** or **q**. Let the user enter **Q** or **q** for the campus abbreviation as a sentinel to quit the loop.
11. Test your program with the following test cases:
    1. campus: **NRG**, buildings: **4**, classrooms **14**, seats per classroom: **24**
    2. campus: **CYP**, buildings: **5**, classrooms **15**, seats per classroom: **25**
    3. campus: **ACC**, (Invalid campus provided, try again)
    4. campus: **RRC**, buildings: **6,** classrooms **16**, seats per classroom: **26**
    5. campus: **Q (or q)**, quit the loop

**Part 2** (15 points)

For this part, you will enhance your Python program from Part 1.

1. The program is a bit unfriendly. It immediately starts asking for a campus abbreviation and numbers. In this part, you make the program easier to use.
2. Add a brief introductory message at the top, telling the user what this program does. Keep it short, like: “This program computes the enrollment capacity for an ACC campus.” Always print this message, just once, when the program starts.
3. Next, give the user a menu of three options. The options are:  
   **d)isplay detailed instructions; c)alculate campus capacity; q)uit program**
4. Implement option **d**: If the user wants to see the detailed instructions, show them more detailed information about what this program does. For example: “Enter an ACC campus abbreviation; enter number of: buildings, classrooms and average seats per classroom, and this program will output the total seating capacity for that campus.”
5. Implement option **c**: do only the code from part 1 (do not display instructions).
6. Implement option **q**: quit the program (fall out of the loop as before).
7. Reorganize your code, so the program starts with the brief introduction message, then offers the three options. The program should loop until the user enters **Q** or **q** to quit.
8. After your program is working, retest the program **(use the data above)** and copy and paste the test output as comments at the end of your program. If you cannot get looping to work, you may run the program several times.

**Extra credit** (3 points) Keep a running total of the total maximum capacity calculated in option **c** (campus maximum capacity). When the program exits, output the value **formatted** like this:  
**The accumulated total maximum capacity for all campuses entered is: 5,715**

For this exam lab, you should submit one file: **DDHH\_X1\_Lastname.py**

Once you have uploaded this file to Blackboard, Exam 1 Lab, click **Submit.**