**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 block at the top with your **name,** **DDHH** (course designator) and **course** (“COSC1336”).
3. Display a brief introductory message at the top, telling the user what this program does. Use this short message: "This program computes the enrollment capacity for an ACC campus." **Print this message, just once, when the program starts.**
4. Write Python code that asks the user for an ACC campus abbreviation.
5. Accept any one of the following inputs: **CYP, NRG**, **RRC, Q/q** (to quit)
6. If the user enters ***something other than the above***, ***display an error message: "***Invalid campus, try again." and ask again for the campus abbreviation. (Loop back to step 4. Do not repeat the short introductory message.)
7. If the user 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
8. Compute the maximum capacity for that campus. (How many students can attend?)
9. Format output to look like this: **ACC’s NRG campus can serve up to 1,344 students.**
10. Print the output to the screen. Use format() so numbers like 1344 appear as 1,344.
11. 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.
12. 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**, (Display error message, such as: “Invalid campus, 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 does not provide much help for first-time users. It asks for user input without much instruction. In this part, make the program easier to use.
2. Provide the user a main menu with three options. The options are:  
   **c)alculate campus capacity; h)elpful instructions; q)uit program**
3. Implement option **h**: If the user wants to see the helpful instructions, show them more detailed information about what this program does. For example: “To compute campus capacity, this program requires an ACC campus abbreviation, number of buildings, classrooms and seats per classroom. This program will output the total seating capacity for that campus.”
4. Implement option **c**: use the code from part 1 (do not display instructions).
5. Implement option **q**: quit the program (fall out of the loop as before).
6. If the user inputs an invalid option, display: ***"***Invalid option, try again."
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, test the program. See the sample testing output below. **(Testing uses the same data from step 1 and also tests the new help option.)** 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 sum 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.**

**Hints:**

1. Do part 1 first, then “nest it” into additional code from part 2.
2. The format string for integers with a comma is: ',d'.
3. The program has two nested while loops for the outer (part 2) and inner (part 1) menus.
4. The program, with extra credit, can be done in about 30-40 python statements.

**Sample testing output:**

**This program computes the enrollment capacity for an ACC campus.**

**c)alculate campus capacity; h)elpful instructions; q)uit program: h**

**To compute campus capacity, this program requires an ACC campus**

**abbreviation, number of buildings, classrooms and seats per classroom.**

**This program will output the total seating capacity for that campus.”**

**c)alculate campus capacity; h)elpful instructions; q)uit program: c**

**Enter your choice: CYP, NRG, RRC, Q/q: NRG**

**Enter number of buildings on that campus: 4**

**Enter number of classrooms in each building: 14**

**Enter number of seats in each classroom: 24**

**ACC’s NRG campus can serve up to 1,344 students.**

**Enter your choice: CYP, NRG, RRC, Q/q: CYP**

**Enter number of buildings on that campus: 5**

**Enter number of classrooms in each building: 15**

**Enter number of seats in each classroom: 25**

**ACC’s CYP campus can serve up to 1,875 students.**

**Enter your choice: CYP, NRG, RRC, Q/q: ACC**

**Invalid campus, try again.**

**Enter your choice: CYP, NRG, RRC, Q/q: RRC**

**Enter number of buildings on that campus: 6**

**Enter number of classrooms in each building: 16**

**Enter number of seats in each classroom: 26**

**ACC’s RRC campus can serve up to 2,496 students.**

**Enter your choice: CYP, NRG, RRC, Q/q: q**

**The sum total maximum capacity for all campuses entered is: 5,715**

**c)alculate campus capacity; h)elpful instructions; q)uit program: q**

**Good-bye.**