**COSC 1336 Exam 2 Lab**

Python lab for Exam 2 (60 points). Open book, open note. There are three parts. Read the instructions carefully. Where an IPO is provided, follow the IPO design as specified.

**Part 1**. (30 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 as **DDHH\_X2p1\_Lastname.py**, where **DDHH** is replaced with the course designator and **Lastname** is replaced with your last name.
2. Add a comment at the top with your name, course (COSC1336) and assignment (Exam 2).
3. Write a Python program that does this:
   1. Ask the user for a file name (excluding the .txt). Append the file extension: **.txt** to the name, and create the new file for output.
   2. If the file does not create correctly, catch the exception and print out an error message: **<file.txt> could not be created,** *where* **<file.txt>** *is the exact name of the file being opened. Hint: use try: except: See Program 6-30 in the textbook pg 335 (4ed); pg 283 (3ed); Program 7-30 on pg 287 (2ed).*
   3. If the file opens OK, ask the user for a number of how many random numbers to generate. Call the variable: **randCount**. If user enters 0 for the **randCount** then quit the program.
   4. Use a loop to generate **randCount** random numbers between 1 and 1000, inclusive. To generate random numbers, import random, use random.randint().*See textbook pg 241(4ed); pg 197(3ed); pg 205(2ed).*Write each number to the file. For example, if the user wants 3 random numbers, your file will contain something like:

**451 215 306**

* 1. Close the file.
  2. Test the program on both valid file name: **random.txt** and invalid file name: **random::txt. (It is illegal to have :: in a file name.)**
  3. Paste the output from your test runs at the end of your program in comments. The valid name output should look similar to this:

**What file should hold the random numbers (exclude .txt): random**

**How many random numbers do you want in file random.txt? 5**

**File random.txt written successfully.**

1. Save the file: **DDHH\_X2p1\_Lastname.py** as a backup for what was done so far.

**Part 2**.(30 points)

In this part, you will enhance the code written in Part 1. Add the ability to print the data file generated in Part 1. If your Part 1 does not make a file, you can use the data file: X2\_data.txt.

1. Save another copy of **DDHH\_X2p1\_Lastname.py** to **DDHH\_X2p2\_Lastname.py**
2. Move a portion of the code into a function called **createData**(). Here is the IPO for createData(): *(Think of “****inputs****” as “parameters; “****output****” as return value.)*

**Inputs**: file\_name (name of file to write random numbers to, excluding .txt)  
 randCount (number of random numbers to generate)

**Processing:** open file, write randCount random numbers (1 to 1000) to file, close file.   
 Catch exceptions. In case of failure, return False.

**Output:** True if file opened and written successfully, False otherwise.  
 (file\_name.txt is created on the storage device if successful)

1. In main, ask the user for the filename and number of random numbers, then call the function createData() with the filename and number of random numbers as parameters.
2. ***Verify that your program works as before. It is just reorganized.***
3. Create another function: **showData** which prints a file. Here is the IPO for **showData**():

**Input**: filename (name of file to print to the screen)

**Processing:** open filename, print every line to the screen, close filename. Catch exceptions in case of failure. If failure, return False.

**Output:** True if file opened and printed successfully, False otherwise.

1. Modify main().First. inform the user what the program does. Create a menu-driven loop. In a loop, ask the user what they want to do. The options are:

**create** a file (to contain random numbers); **show** a file; **quit** the program.

Suggested prompt:  
  **Enter option: c)reate file with random numbers; s)how file; q)uit**   
  
If the user wants to create a file, ask for the file name and the number of random numbers and call createData(). Else, if they want to show the file, ask for the file name and call showData(). Else, if they want to quit the program, drop out of the menu-driven loop and exit the program. In your loop, display the prompt each time until the user quits.

1. Test your program. It should: create a new file; print the file; and quit. After your program is working, copy and paste sample output as comments at the end of your program.
2. Save **DDHH\_X2p2\_Lastname.py** for later. If out of time, submit to blackboard **Exam 2 Lab**.

**Part 3**. (5 points extra credit)

In Part 1, you created a data file. In Part 2, you could also print it to the screen. In Part 3, you add a processing step. Make a backup copy of what you produced so far.

1. Make a copy of **DDHH\_X2p2\_Lastname.py.** Name it: **DDHH\_X2p3\_Lastname.py**
2. Add a new function: evenodd() Here is the IPO for evenodd():

**Input**: number  
**Processing:** determine whether number is even or odd.  
**Output:** True if number is even, False otherwise

1. Modify showData to call evenodd() for each number in the input file. When printing out the data, print whether it is even or odd: Example:  
   **451 is odd  
   215 is odd  
   306 is even**

For this exam lab, submit **one** file. Submit the file with the most function, the farthest you got: **DDHH\_X2p1\_Lastname.py OR DDHH\_X2p2\_Lastname.py, OR DDHH\_X2p3\_Lastname.py**

Once you have uploaded your file, click **Submit**

*Note: If you did not study Chapter 6 on Files and Exceptions, for partial credit, try doing this lab without using files. You can generate random numbers and output even or odd.*