Final Project – MIS 304 (03630)
Due to Canvas: Thursday, May 10, 5:00pm

Registration Application

UT wants to build a new system for students to register for classes each term. To help them, write a program to store and manage course information and student enrollment, and allow students to add or drop classes. UT has provided two text files containing existing classes and student enrollment for you. The first few lines of student data found in students.txt looks like this:

```
as143:Arya:Stark:34128:31255:35123
bs503:Bran:Stark:31255:31453
cl402:Cersei:Lannister:31255:34128:31940
dt329:Daenerys:Targaryen:31255:31453
es402:Eddard:Stark:31940:31255:31453
```

The course data found in courses.txt looks like this:

```
31940;MIS 304;Conley;9;10
34128;MIS 325;Mote;8;10
31255;MIS 333K;Gray;13;14
31453;MIS 374;Tuttle;9;10
35123;MIS 375;Tanriverdi;3;10
```

You must write a program to process the data in students.txt and data in the courses.txt using functions. Each time a student adds or drops a course, you must update the student's list of courses and also update the number of seats taken in the course. The user may view the course schedule and also view the courses a particular student is enrolled in. When the user ends the program, you will create two new text files (courses-updated.txt, students-updated.txt) with all updated course and student information.

Main Menu to be displayed for the user:

```
1. Add course
2. Drop course
3. Print student's schedule
4. Print course schedule
5. Plot a bar graph of course number on X-axis, capacity and actual enrollment on
the Y-axis
6. Done
```

Note: When running the program, the user will enter the menu item number, NOT the name of the menu option (e.g. 1, not "Add course").

Program requirements:

1. Define a function to store student information.
2. Define a function to store course information.
3. Your program MUST read all of the information about the students from students.txt and store the information in the program. You must use a dictionary to keep track of all students.
4. Your program MUST read all of the information about courses from courses.txt and store the information in the program. You must use a dictionary to keep track of all courses.
5. Your program MUST define and call the following user-defined functions, in addition to the `main` function, that is different from Student and Course (e.g. LastName_FirstName_FP.py). You may create additional functions, but the following functions in the main program are required for full credit:
   - Process the students file
     - HINT: Open the input file, create the student instances and add them to the student dictionary
   - Process the courses file
     - HINT: Open the input file, create the course instances and add them to the course dictionary
   - Print menu
   - Get menu option from the user

    - Verify the option is a valid menu option
- Print course schedule (prints all courses and enrollments)
- Print student information, including a list of courses a particular student is enrolled in
- Get student ID from the user
    - Verify that the student exists in the student dictionary. If not, ask student for name and add id and student name to the dictionary.
- Get unique number from the user
    - Verify that the course exists in the course dictionary
- Write updated object information to an output file
    - Create the students-updated.txt or courses-updated.txt output files (NOTE: This function should only create one output file each time it is called)

Other program details:

- Be sure that your program works for input files that contain any number of students in students.txt and any number of courses in courses.txt. The data structure in each file will remain the same as the sample input files provided, but the values in these files may change (e.g. do not hard-code your program to only work with the unique numbers in courses.txt).
- Students should not be allowed to add a course they are already enrolled in.
- Students should not be allowed to drop a course they are not yet enrolled in.
- Students should only be allowed to add a course if there is at least 1 seat available in that course.
- When a student adds or drops a course, be sure to updated the enrollment of that course accordingly.
- Print the main menu each time you prompt the user to choose the next action.
- If error handling or data validation is not specified in the instructions or grade script, you do NOT have to include this logic in your program.
- Use constants to represent all literals, such as file names, sentinels, etc.
- All program logic, other than the call to the `main` function, must be included in user-defined functions. You may include additional logic, other than calls to functions, inside of `main`.

This assignment MUST be created individually by you. You must turn in your OWN Python files. You MAY NOT share files with other pairs or individuals.

Instructions

- Test your program to ensure that it works correctly using the sample script.
- Be sure to run your final program using the data included in the sample input file to ensure your program works properly.
- Be sure to format your output according to the instructions above and in the sample grade script – you will lose points if you do not.

Notes

- You may perform the tasks in any order as long as your output follows the order of the output in the above instructions.

To receive full credit:

- Submit your Python files (.py files) to Canvas. The function files should be named appropriately. The file with the main function should be named LastName_FirstName.py for full credit. **Create a zip file containing your .py files, and submit the .zip file to Canvas.**
- You must follow the appropriate Coding Standards listed in the Coding Standards document under Course Documents on Canvas.
    - o 20% of your grade will be based on how well you follow these standards and how well you comment your source code
- Submit your Python files to Canvas using the Assignment submission feature by the deadline listed above.