Read this entire page. You are responsible knowing what it says.

Honor

- By submitting solutions for this test, you are agreeing that
    - You neither given nor received help directly or indirectly to or from anyone else;
    - You did not directly or indirectly use materials from non-allowed sources.

Important

- **You must use our files when coding.**
- The `WWHAD` strategy (what would a human do strategy) should serve you well.
- Question 1 is worth 10 points. The other questions are worth 15 points.
- During the test you may not access past code or algorithms (yours, ours, or anyone else's).
- During the test you may not access class notes, epistles, examples, artifacts, solutions on the web, or your own past assignments during the test.
- Class personnel cannot help you debug your answers.
- Comment out or delete all debugging `print()` statements before submitting.
- Whether code is testable is important.
- The only device you may access during the exam is your laptop. The only open windows allowed are PyCharm and a browser with tabs linked from the class website.
- During the test you can access the course module descriptions and the course Python information sheet.
- You are responsible for submitting for your work, so check before exiting the testing. Late submissions will not be graded, so do not submit once your testing time is up.
- Code should follow class programming practices; e.g., whitespace, identifier naming, etc.
- Because the solutions are all short, commenting is not necessary.
- You might add comments if you were unable to complete a problem and want to explain what you were attempting to do.

Problems

1. Implement program *promise.py*. The program prints whether you acted honorably on this test. If so, the output should be the string "*I was honorable.*". Thus, I expect all of your programs to produce the following output.

```
I was honorable.
```

2. Implement program *oops.py.* The program separately prompts for two integers *x* and *y*, and then processes them as follows. *Note, only print what is requested and in the indicated order.*

   a) Prints the product of *x* and *y*.

   b) Prints the integer quotient of *x* divided by *y*.

   c) Prints the integer remainder of *x* divided *y*.

   Some possible program runs are

```
Enter an integer: 9
Enter an integer: 2
18
4
1
```

```
Enter an integer: 7
Enter an integer: 11
77
0
7
```

3. Implement program *geom.py*. The program has a single prompt that gets two integers *x* and *n*. The program calculates and prints the value of the below expression (*it prints nothing else*).

$$1 + \frac{1 - x^n}{1 - x}$$

   Note: in the above expression, integer division is being performed.

   Reminder: the Python exponentiation operator is **.

   Some possible program runs are

```
Enter two integers: 2 10
1024
```

```
Enter two integers: 5 8
97657
```

4. Implement program *rss.py.* The program prompts for two words and processes them as follows. *Note, only print what is requested and in the indicated order.*

   a) Prints a version of the input with each space replaced with a hyphen (`-`).

   b) Prints a version of the input without any leading or trailing whitespace.

   c) Prints the first word of the response with the letters in upper case.

   d) Prints the second word of the response with the letters in lower case.

   Some possible program runs are

```
Enter text:   at   LAST
--at---LAST-
at   LAST
AT
last
```

User entered two spaces in front of `at`, three spaces after `at`, and one space after `LAST`.

```
Enter text:    Play  Time
---Play--Time
Play  Time
PLAY
time
```

User entered three spaces in front of `Play` and two spaces after `Play`.

5. Implement program *duck.py.* The program prompts for text and processes it as follows. *Note, only print what is requested and in the indicated order.*

   a) Prints the list of words.

   b) Prints an integer list of the word lengths.

   c) Prints the length of the longest word.

   d) Prints the longest word. Observation: the index of maximum length in the list of word lengths is also the index of the longest word in the list of words.

   Some possible program runs are

```
Enter text: mallard alabia eider
['mallard', 'alabia', 'eider']
[7, 6, 5]
7
mallard
```

```
Enter text: wood pochard harlequin muscovy
['wood', 'pochard', 'harlequin', 'muscovy']
[4, 7, 9, 7]
9
harlequin
```

6. Implement program *spach.py.* The program behaves as follows. *Note, only print what is requested and in the indicated order.*

   a) Prompts for an integer seed value. It uses that *integer* to set the Python random number generator.

   b) Prompts for text. It splits the text into a list of words and prints the list.

   c) Randomly selects and prints seven words from the list of words. The words are printed one-per line.

   Recommendation: A loop is not necessary because the number of outputs is known to be seven. However, I believe your code will be simpler to write if you do use a loop.

   Some possible program runs are

```
Enter seed (integer): 1112
Enter text: a b c
['a', 'b', 'c']
b
a
b
c
a
b
a
```

```
Enter seed (integer): 12
Enter text: day it that we have dream one I a so
['day', 'it', 'that', 'we', 'have', 'dream', 'one', 'I', 'a', 'so']
I
have
a
dream
that
one
day
```

7. Implement Program *manx.py*. The program prompts for the name of a dataset. *The dataset will not have a header row*. *Note, only print what is requested and in the indicated order*.

   The dataset will reside in web repository:

   ```
   REPOSITORY = 'http://www.cs.virginia.edu/~cs1112/datasets/testing/'
   ```

   a) The program prints the dataset.

   b) The program prints the maximum value in the dataset.

   Recommendation: after getting the dataset, create a list of maximum values whose first value is the maximum value in the first row, whose second value is the maximum value in the second row, and so on. The maximum value in the list of maximums is the answer.

   Two datasets *table1.csv* and *table2.csv* are available for testing your program.

   A program run examining *table1.csv* should produce the following.

   ```
   Enter file name: table1.csv
   [[8, 82, 4], [2, 13], [7, 36, 57], [5, 11], [9, 21, 86], [3, 26]]
   86
   ```

   A program run examining *table2.csv* should produce the following.

   ```
   Enter file name: table2.csv
   [['to', 'of', 'in'], ['for', 'on', 'at'], ['by', 'up', 'into'], ['over']]
   up
   ```