CSE101 – Spring 2021
Programming Assignment #7

Due May 27, 2021 by 11:59pm, KST. The assignment is worth 13 points (and 2 additional points of extra credit).

Instructions

For each of the following problems, create an error-free Python program.

- Each program should be submitted in a separate Python file that follows a particular naming convention: Submit the answer for problem 1 as "Assign7Answer1.py" and for problem 2 as "Assign7Answer2.py" and so on.
- These programs should execute properly in VS Code using the setup we created in lab.
- At the top of every file add your name and Stony Brook email address in a comment.
- Include all the provided test cases for all problems that do not take user input.
- Download these files – unzip the files and add them to your project directory. These provide starter code and supplemental files for use with the problems below.

Regarding working in pairs:

- You are welcome to work with a partner on the homework assignment, but you **MUST write both your names and email addresses in each file** in a comment. Only one person needs to submit the homework on Blackboard.
- You are only allowed to work together with one other person – larger group submissions or collaborations (beyond high-level discussions of problems, as stated on the syllabus) are not allowed.

Problem 1: (3 points)

For this part of the assignment you will implement a single-player variant of the children's game *Going to Boston*. The computer plays three rounds of the game using three dice:

1. On the first throw, the highest number is put to one side. If two or more of the dice show the highest number, only one is put to the side.
2. The remaining two dice are thrown and once again the highest-numbered die is put aside.
3. The final die is then rolled and the total of all three dice is the score.

Complete the `boston` function, which takes a single argument representing the number of sides on the dice. You may assume that the input argument is valid and is at least 4. The function returns the score after rolling the dice during the 3 rounds.

Start from the provided Assign7Answer1.py, which also includes test cases to test your program.

Problem 2: (4 points)

Write a Python program using classes that will keep track of people attending an international conference. For this problem, start from the provided Assign7Answer2.py file and add the attendees.txt file to your project to test it.

Write an Attendee class that will be used to keep track of an attendee's information as follows:

- Name
- Company
- Country
- Email address

The Attendee class should provide a constructor method that takes as input the above arguments and saves the information for an Attendee. It should be possible to create an object of an Attendee class as follows:

```
Attendee1 = Attendee("John", "ABC Pharma", "Barbados", "john@abcpharma.com")
```

The Attendee class should have a method `getInfo()` that returns the attendee's information as a dictionary. For example:

```
>>>Attendee1.getInfo()
{"Name":"John", "Company":"ABC Pharma", "Country":"Barbados",
"Email":"john@abcpharma.com"}
```

Additionally, you need to write a Conference class. This class will have methods to support several features, listed below. Read all of the directions first, but I would suggest doing these tasks one at a time and testing it before implementing the next task:

1. The conference class constructor takes as arguments the conference name, location, and dates. All of these should be strings, as shown below:

```
>>>conference1 = Conference("National conference on computing", "Seoul", "22-
25 December, 2020")
Conference "National conference on computing" is created.
```

2. Add an attendee to the conference. For Example:

```
>>> conference1.addAttendee(Attendee1)
Attendee John is added to the list of conference participants.
```

3. Remove an existing attendee from the conference. For example:

```
>>> conference1.removeAttendee(Attendee1)
Attendee John is removed from the list of conference participants.
```

4. Display all of the attendees (displaying the name, company, country, and email address for each attendee). For example:

```
>>> conference1.displayAttendees()
-----------------------------------------------------------------------
Name        Company          Country                Email
-----------------------------------------------------------------------
John        ABC Pharma       Barbados          john@abcpharma.com
Lisa        HealthTech Inc.  Malaysia          lisa@healthtech.inc
```

5. Check if a specific person is attending the conference (matching on their name) and return True if that person is attending.

```
>>> conference1.checkAttendee("John")
John is attending the conference.
>>> conference1.checkAttendee("Seung")
Seung is not attending the conference.
```

6. Display all of the attendees from a specified country.

```
>>> conference1.displayAttendeesFromCountry("Barbados")
-----------------------------------------------------------------------
Name        Company          Country                Email
-----------------------------------------------------------------------
John        ABC Pharma       Barbados          john@abcpharma.com
```

Problem 3: Wolfie Numerals (6 points)

For this problem you will be writing some functions to convert between two number representations: the familiar Arabic numerals we use every day and a new scheme we call *Wolfie Numerals*.

Use the provided Assign7Answer3.py initial code to get started.

Part 1. Arabic Numerals to Wolfie Numerals Converter (3 points)

You've heard of Roman numerals, but have you heard of Wolfie numerals? Probably not because some CS people at SBU have invented them and we are using it for this assignment. Wolfie numerals are similar to Roman numerals in that numbers are formed by combining symbols and adding the values. In Wolfie numerals, each numeral has a fixed value representing a power of 8 (rather than a power of 10 or half of a power of 10, as in Roman numerals):

Symbol:	I	E	S
Value:	1	8	64

In addition, there are also symbols representing the halved powers of 8. A complete list of Wolfie numerals is given below:

Symbol:	I	F	E	T	S
Value:	1	4	8	32	64

Symbols are placed from left to right in order of value, starting with the largest. For example, `EFII` is 14 : (8 × 1) + (4 × 1) + (1 × 2) = 8 + 4 + 2 = 14.

Note: You may want to review (or learn) the rules for the Roman numerals to better understand this assignment. In a few specific cases, to avoid having three or more characters being repeated in succession (such as `III` or `EEE`), subtractive notation is used, meaning that a smaller number is listed before the bigger value, as in this table:

Number:	3 (4-1)	7 (8-1)	24 (32-8)	56 (64-8)
Notation:	IF	IE	ET	ES

For example, with these subtractive notations, instead of writing 7 as 4 + 1 + 1 + 1 (`FIII`), you must write it as 8 − 1 (`IE`), to shorten the numeral sequence.

Thus, there must **never be more than two** instances of a single symbol next to each other in a Wolfie numeral. Also, only the numerals that are powers of 8 can be repeated. Moreover, it is disallowed for a symbol to be used in an additive manner *after* it has been used in a subtractive manner. To see why this is true, consider Roman numerals for a moment. `CXC` (190) is valid because `C` is used first in an additive way and *then* in a subtractive way. But `XCXX` is invalid because `X` is first used in a subtractive way and *then* in an additive way. In Wolfie numerals, examples of similar invalid cases would be expressions like `IFII`, `ETE`, and `ESEE`.

Based on the rules described above, an updated table of the numerals can be described as below:

Symbol:	I	IF	F	IE	E	ET	T	ES	S
Value:	1	3	4	7	8	24	32	56	64
Max # of Appearances:	2	1	1	1	2	1	1	1	2

In `Assign7Answer1.py` complete the function `arabic2wolfie` that takes one parameter, `num`, which is an **Arabic** numeral that you will need to convert to the Wolfie numerals. Your function should return the Wolfie numeral as a string.

Note: You will only be given `num` in the range `[1, 63]` (inclusive of both).

Below are some examples to help you understand. Keep in mind that you want the sequence to be *as short as possible* while following the numeral rules.

Example #1: `num = 10`
Look from the larger Wolfie numerals to the smaller numerals. What is first value you can subtract? It's 8. Now, you have 2 left to worry about, and the only applicable value is 1. Since 1 can appear twice, we have two 1's.

Therefore, 10 = 8 + 1 + 1, and this translates to `EII`.

Example #2: `num = 14`
Look from the larger Wolfie numerals to the smaller numerals. What is first value you can subtract? It's 8. Now, you have 6 left to worry about.

Applying the same technique, 6 can be translated to 4 + 1 + 1. Therefore, 14 = 8 + 4 + 1 + 1, which translates to `EFII`.

Example #3: `num = 22`
Applying the same technique, first we can subtract two 8's because `E` can be repeated. Then, we have 6 to worry about. From the last example we know that it can be expressed by 4+1+1. Therefore, 22 = 8+8+4+1+1, which translates to `EEFII`.

Example #4: `num = 28`
Applying the same technique as above, 28 = 24 + 4, which translates to `ETF`.

Example #5: `num = 30`
Applying the same technique as above, 30 = 24 + 4 + 1 + 1, which translates to `ETFII`.

Example #6: `num = 54`
Applying the same technique as above, 54 = 32 + 8 + 8 + 4 + 1 + 1, which translates to `TEEFII`.

Test Cases:

```
arabic2wolfie(10) returns 'EII'
arabic2wolfie(14) returns 'EFII'
arabic2wolfie(22) returns 'EEFII'
arabic2wolfie(28) returns 'ETF'
arabic2wolfie(30) returns 'ETFII'
arabic2wolfie(54) returns 'TEEFII'
```

Part 2. Wolfie Numerals to Arabic Numerals Converter (3 points)

In `Assign7Answer1.py` complete the function `wolfie2arabic` that takes one string parameter, `numerals`, consisting of properly-formatted Wolfie numerals. Your function should return the integer represented by the Wolfie numerals. The converted integer is guaranteed to be in the range `[1, 63]`.

Process the input string from left-to-right, looking for combinations of numerals (possibly indicating subtraction) or single numerals that are not involved in subtraction. Add the value of the numerals to a running total as you process the string.

To find combinations of numerals, consider two characters simultaneously (e.g., `IE` or `ES`), taking care not to overrun the end of the string while doing this (an "index out of range" error). Proceed in this manner until the rightmost numeral has been processed.

Example Test Cases:

```
wolfie2arabic('EII')    returns 10
wolfie2arabic('EFII')   returns 14
wolfie2arabic('EEFII')  returns 22
wolfie2arabic('ETF')    returns 28
wolfie2arabic('ETFII')  returns 30
wolfie2arabic('TEEFII') returns 54
```

Problem 4: Extra Credit: Generate random phone numbers (2 points)

Complete the function `phone_number`, which returns a random phone number that satisfies the following requirements *and which is no more restrictive than these requirements*:

- Starts with `1`, followed by a space.
- Next, 3 digits that may not start with 0 or 1, and is contained within parentheses.
- Another space.
- 3 digits that form a 3-digit, even integer
- A dash.
- Four digits that form a number in the range 0000 through 5999, inclusive.

For example, `1 (516) 892-3124` satisfies the above requirements.

Start from the provided Assign7Answer4.py file, which contains code to test your solution.