NYU, Tandon School of Engineering
CS-UY 1114 Introduction to Programming and Problem Solving — Fall 2017

Homework #9
Due by Monday 12/11, 11:55pm

Submission instructions:

1. You should submit your homework in the NYU Classes system. You should turn in 2 '.py' files.
2. Create one '.py' file per question. Each '.py' file may contain multiple functions, and classes definitions.
3. For this assignment, you should turn in 3 '.py' files. Name your files 'YourNetID_hw9_q1.py', 'YourNetID_hw9_q2.py', and 'YourNetID_hw9_q3.py'.

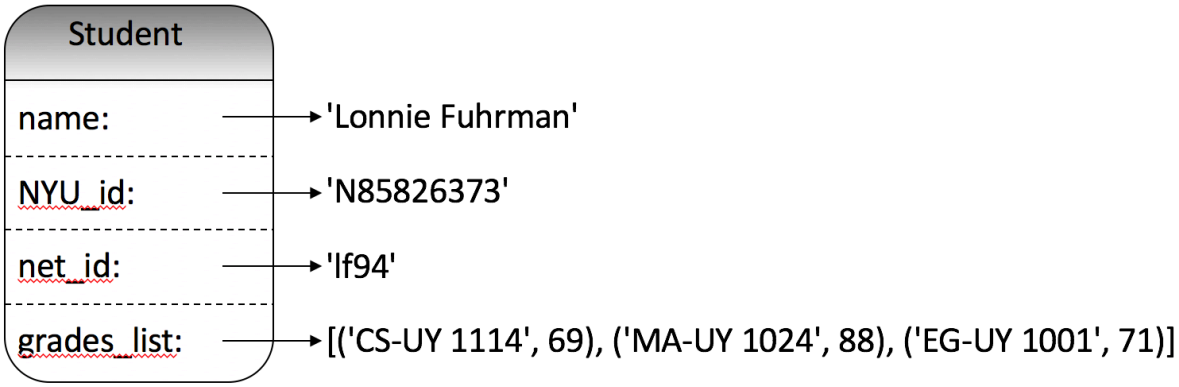
Question 1:

a. Implement a **`class`** `Student`, that will be used to store student's information, and to perform some operations related to the student.

Each Student instance should have the following data-members:

- `name` – Will hold the student's name.
- `NYU_id` – Will hold the student's NYU ID (of type str)
- `net_id` – Will hold the student's NetID (of type str)
- `grades_list` – Will hold the student's grade list. Each element in that list would represent the student's grade in some course, as a pair (of type tuple), where the first component is the course's catalog name, and the second is the student's grade in that course.

For example, if Lonnie Fuhrman is a student with NYU ID: N85826373, NetID: lf94, and with the grades: 69 in CS-UY 1114, 88 in MA-UY 1024, and 71 in EG-UY 1001, then the `Student` object representing him should look like:



A `Student` object should support following methods:

- `__init__(self, name, NYU_id, net_id)` – Will initialize a new `Student` instance with a name, NYU ID, and NetID as given in the parameters. The new student's grade list should be initialized to an empty list.
- `add_grade(self, catalog_name, grade)` – Will **modify** the student's grades list, by adding an entry for the course `cataog_name` with the grade `grade`.
- `average(self)` – Will return the average of the student's grades, rounded to the closest integer.
- `get_email(self)` – Will return the student's email address. Note that an email address should be the student's NetID, followed by '@nyu.edu'.

b. Implement a function:

```
def load_students(students_data_filename)
```

This function gets `students_data_filename`, a name of a '.csv' file that contains information of students. The exact format of the data in that file will be described below.
When called, the function should create and return a list with `Student` objects, representing all the students' data given in the file.

The students' data file would have a header line, followed by a sequence of lines, one for each student. Each such line would contain (in this order): the student's NYU ID, name NetID, and grades in CS-UY 1114, MA-UY 1024, EG-UY 1001, EG-UY 1003, CS-UY 1122, CS-UY 1134, MA-UY 1124.
Note that a student didn't necessarily take all these courses, therefore each student would have only grades for the courses they completed (keeping the grades for courses they didn't take yet empty).

For example, if you open this '.csv' file in a spreadsheet application, the first 6 lines could look like:

NYU ID	Name	NetID	CS-UY 1114	MA-UY 1024	EG-UY 1001	EG-UY 1003	CS-UY 1122	CS-UY 1134	MA-UY 1124
N97604433	Joetta Sotomayor	js454		100	87	70	85	66	
N66238159	Melissia Scroggins	ms643	92		98	66			63
N85826373	Lonnie Fuhrman	lf94	69	88	71				
N38549165	Antoinette Poppell	ap25			81	78	82	81	89
N12000237	Kathern Haddon	kh511	78	98	85	62			

Since it's a '.csv' file, the same first 6 lines, when opened in a plain text editor application would look like:

```
NYU ID,Name,NetID,CS-UY 1114,MA-UY 1024,EG-UY 1001,EG-UY 1003,CS-UY 1122,CS-UY 1134,MA-UY 1124
N97604433,Joetta Sotomayor,js454,,100,87,70,85,66,
N66238159,Melissia Scroggins,ms643,92,,98,66,,,63
N85826373,Lonnie Fuhrman,lf94,69,88,71,,,,
N38549165,Antoinette Poppell,ap25,,,81,78,82,81,89
N12000237,Kathern Haddon,kh511,78,98,85,62,,,
```

Note: We attached to this assignment, a file named 'hw9 - students grades.csv' containing such data (of random students) in the format described above. You may use this file to test your function.

c. Implement a function:

```
def generate_performance_report(students_lst, out_filename)
```

This function gets `students_lst`, a list of `Student` objects, and `out_filename`, a name of a file to where the function should write its output.
When called, the function should write a '.csv' file with a performance report of the students. That is, each student line should have the NYU ID, and the average the student's grades.

Notes:
1. Your file should also include a header line.
2. Use method/s from the `Student` class

For example, if we call this function with the students list created from the file shown in section (b), the first few lines of the output file should look like:

```
NYU ID,Average
N97604433,82
N66238159,80
N85826373,76
N38549165,82
N12000237,81
```

d. Implement a function:

```
def generate_course_mailing_list(students_lst, catalog_name,
                                 out_filename)
```

This function gets `students_lst` - a list of `Student` objects, `catalog_name` - a string with the catalog name of a course, and `out_filename` - a name of a file to where the function should write its output.
When called, the function should write a '.txt' file with the email address of all students from `students_lst` that took the course `catalog_name` (each email address in a separate line).

Notes:
1. A student who took a course, would have a grade for that course.
2. Your file should **not** include a header line.
3. Use methods from the `Student` class

For example, if we call this function with the students list created from the file shown in section (b), and with `catalog_name='CS-UY 1114'`, the first few lines of the output file should look like:

```
ms643@nyu.edu
lf94@nyu.edu
kh511@nyu.edu
lc526@nyu.edu
mh9873@nyu.edu
```

e. Implement a program:

```
def main()
```

When called, it should read the students data from the 'hw9 - students grades.csv' file (attached with this assignment) and generate a performance report, as well as email lists for all the courses.

Question 2:
In this question, we will write a program to play the *Pig-Game*. The Pig-Game is a jeopardy dice game.

The object is to be the first player to reach 100 points.
Each player's turn consists of repeatedly rolling a die. After each roll, the player is faced with two choices: *roll* again, or *hold* (decline to roll again).

- If the player rolls a 1, the player scores nothing in that turn, and it becomes the opponent's turn.
- If the player rolls a number other than 1, the number is added to the player's turn total and the player's turn continues.
- If the player holds, the *turn total*, the sum of the rolls during the turn, is added to the player's score, and it becomes the opponent's turn.

You can read some more about this game at:

- https://en.wikipedia.org/wiki/Pig_(dice_game)
- http://cs.gettysburg.edu/projects/pig/index.html

You should implement 3 classes: `Die`, `PigGamePlayer`, and `PigGame`

1. The `Die` class: Will be used to represent a die
   Each `Die` instance should have the following data-members:
   - `number_of_faces` – Will hold the number of faces that the die has. It should be set to 6 by default, unless a different value is given at construction time.
   - `curr_face_value` – Will hold the current face value.

   A `Die` object should support following methods:
   - `__init__(self, faces=6)` – Will initialize a new `Die` instance.
   - `__repr__(self)` – Will return a string with the current value showed on the die.
   - `roll(self)` – Will modify `curr_face_value` with a random die roll outcome.

2. The `PigGamePlayer` class: Will be used to represent a player of the Pig-Game
   Each `PigGamePlayer` instance should have the following data-members:
   - `name` – Will hold the player's name (of type `str`)
   - `die` – Will hold the die that will be used by the player (of type `Die`)
   - `score` – Will hold the player's score

   A `PigGamePlayer` object should support following methods:
   - `__init__(self, name)` – Will initialize a new `PigGamePlayer` instance, for a player named `name`.
   - `play_turn(self)` – Will interact with the user to play a **single turn** in a Pig-Game.

3. The `PigGame` class: Will be used to represent a Pig-Game game

   Each `PigGame` instance should have the following data-members:

   - `player1` – Will hold the first player (of type `PigGamePlayer`)
   - `player2` – Will hold the second player (of type `PigGamePlayer`)

   A `PigGame` object should support following methods:

   - `__init__(self, player1_name, player2_name)` – Will initialize a new `PigGame` instance, used to play the Pig-Game by 2 players, named `player1_name` and `player2_name`.
   - `play(self)` – Will interact with the user to play the Pig-Game.

You are given the following `main` function:

```
def main():
    name1 = input("Player #1, enter your name: ")
    name2 = input("Player #2, enter your name: ")
    game1 = PigGame(name1, name2)
    game1.play()
```

Your implementation should interact with the user **exactly** as demonstrated in Appendix A.

Question 3:

In this question, we will implement the class `BinaryPositiveInteger`, that will be used to represent positive integer numbers, and preform some operations on them. The `BinaryPositiveInteger` objects will represent positive integers by their binary (base 2) representation. That is, each object will have a data-member of type string, containing the binary representation of the number.

For example, the object representing the number 13, will have `'1101'`, as its string data-member.

Complete the definition of the following `BinaryPositiveInteger` class:

```
class BinaryPositiveInteger:
    def __init__(self, num):
        ''' Initializes a BinaryPositiveInteger object
        representing the value given in the integer num'''

    def __add__(self, other):
        ''' Creates and returns a BinaryPositiveInteger object
        that represent the sum of self and other (also of
        type BinaryPositiveInteger)'''

    def __lt__(self, other):
        ''' returns True if self is less than other, or False
        otherwise'''

    def is_power_of_2(self):
        ''' returns True if self is a power of 2, or False
        otherwise'''

    def largest_power_of_2(self):
        ''' returns the largest power of 2 that is less than or
        equal to self'''

    def __repr__(self):
        ''' Creates and returns the string representation
         of self. The string representation starts with 0b,
         followed by a sequence of 0s and 1s'''
```

For example, after implementing the `BinaryPositiveInteger` class, you should expect the following behavior:

```
>>> n1 = BinaryPositiveInteger(13)
>>> n1
0b1101
>>> n2 = BinaryPositiveInteger(25)
>>> n2
0b11001
>>> n1.is_power_of_2()
False
```

```
>>> n1.largest_power_of_2()
8
>>> n1 < n2
True
>>> n1 + n2
0b100110
```

Implementation Requirements:

1. Each `BinaryPositiveInteger` object should have **only one data-member** (which is the string with the binary representation of the number).
2. You are not allowed to use the `int(…)` and `bin(…)` functions, as well as the `format` method of the `str` class.
3. In the `__init__` method you should convert the given `int` to its binary representation. From that point, all the other methods (`__add__`, `__lt__`, `is_power_of_2`, `largest_power_of_2` and `__repr__`) should make their computations in the binary representation, and should **not** convert the numbers back to `int`s.
   Specifically, when adding two `BinaryPositiveInteger` objects, you should implement the "Elementary School" technique.

Extra Credit:

Support also the multiplication of two `BinaryPositiveInteger` objects (by implementing the "Elementary School" multiplication technique):

```
def __mul__(self, other):
    ''' Creates and returns a BinaryPositiveInteger object
    that represent the multiplication of self and other
    (also of type BinaryPositiveInteger)'''
```

Appendix A.

```
Player #1, enter your name: >? Arya
Player #2, enter your name: >? Sansa

Arya's turn:
You rolled 3.
Your score for this turn is: 3
Roll again? (type 'r' for roll, or 'h' for hold): >? r
You rolled 2.
Your score for this turn is: 5
Roll again? (type 'r' for roll, or 'h' for hold): >? r
You rolled 3.
Your score for this turn is: 8
Roll again? (type 'r' for roll, or 'h' for hold): >? r
You rolled 6.
Your score for this turn is: 14
Roll again? (type 'r' for roll, or 'h' for hold): >? h
You scored 14 points this turn. Your total score is 14

Sansa's turn:
You rolled 6.
Your score for this turn is: 6
Roll again? (type 'r' for roll, or 'h' for hold): >? r
You rolled 2.
Your score for this turn is: 8
Roll again? (type 'r' for roll, or 'h' for hold): >? r
You rolled 3.
Your score for this turn is: 11
Roll again? (type 'r' for roll, or 'h' for hold): >? r
You rolled 4.
Your score for this turn is: 15
Roll again? (type 'r' for roll, or 'h' for hold): >? h
You scored 15 points this turn. Your total score is 15

Arya's turn:
You rolled 6.
Your score for this turn is: 6
Roll again? (type 'r' for roll, or 'h' for hold): >? r
You rolled 1.
You scored 0 points this turn. Your total score is 14

Sansa's turn:
You rolled 6.
Your score for this turn is: 6
Roll again? (type 'r' for roll, or 'h' for hold): >? r
You rolled 5.
Your score for this turn is: 11
Roll again? (type 'r' for roll, or 'h' for hold): >? r
You rolled 6.
Your score for this turn is: 17
Roll again? (type 'r' for roll, or 'h' for hold): >? h
You scored 17 points this turn. Your total score is 32
```

```
Arya's turn:
You rolled 1.
You scored 0 points this turn. Your total score is 14

…
…
…
…
…

Sansa's turn:
You rolled 2.
Your score for this turn is: 2
Roll again? (type 'r' for roll, or 'h' for hold): >? r
You rolled 3.
Your score for this turn is: 5
Roll again? (type 'r' for roll, or 'h' for hold): >? r
You rolled 4.
Your score for this turn is: 9
Roll again? (type 'r' for roll, or 'h' for hold): >? h
You scored 9 points this turn. Your total score is 98

Arya's turn:
You rolled 4.
Your score for this turn is: 4
Roll again? (type 'r' for roll, or 'h' for hold): >? r
You rolled 3.
Your score for this turn is: 7
Roll again? (type 'r' for roll, or 'h' for hold): >? r
You rolled 2.
Your score for this turn is: 9
Roll again? (type 'r' for roll, or 'h' for hold): >? h
You scored 9 points this turn. Your total score is 87

Sansa's turn:
You rolled 4.
Your score for this turn is: 4
Roll again? (type 'r' for roll, or 'h' for hold): >? h
You scored 4 points this turn. Your total score is 102

Sansa won!
```