COS 126 Programming Exam 2 Fall 2020

YOU MAY READ ONLY THIS PAGE BEFORE THE EXAM PERIOD BEGINS.
DO NOT READ THE FOLLOWING PAGES UNTIL SO INSTRUCTED.

Instructions. You will have 60 minutes to create and submit two programs. When told to start the exam, download the project `.zip` file. It includes all the files you will need to start.

Resources. You may use your book, your notes, your code from programming assignments, labs and precepts, the code on the COS 126 course website, the booksite and Ed posts. No form of communication is permitted (e.g., talking, texting, etc.) during the exam, except with course staff.

Submissions. Submit your work using the Submit! link. You are responsible for uploading the correct version of your solutions. You must submit your final solution before time is called. Strictly enforced. No exceptions. **You are required to submit the state of your exam every 15 minutes (at the announcements) and also just before starting Part II.**

Grading. Your program will be graded on correctness, clarity (including comments), design, and reasonable efficiency. **You will lose a substantial number of points if your program does not compile**.

Discussing this exam. Some of your peers will take this exam on a different day/time. Do not show this exam to anyone until after next week and do not discuss exam contents with anyone who has not taken the exam.

Honor code pledge. The project `.zip` includes a file named **`HONORCODE.TXT`**. When you are instructed to start, you may "electronically sign" the honor code, **by typing this pledge and then your name in this file**.

I pledge my honor that I have not violated the Honor Code during this examination.
<your name>

You must properly complete and submit `HONORCODE.TXT`

Context of this exam. Random numbers play an essential role in many domains including cryptography, games, science, and art. Unfortunately, completely deterministic machines cannot produce truly random numbers. Instead, to make computers useful for applications in these important domains, computer scientists have developed many algorithms to approximate randomness with "pseudo-randomness." In this exam, we explore one such algorithm.

DO NOT READ FURTHER OR START DOWNLOADING UNTIL SO INSTRUCTED.

Part I: (70%)

The *Middle Square Method* (MSM) is an algorithm invented by John von Neumann for producing pseudorandom numbers (PRNs). Each PRN generated by the MSM is an n-digit number that also serves as MSM's internal state, the value it uses to compute the state/PRN in the next round. MSM computes the next PRN (i.e., the next state) value by 1) squaring its current state to produce a 2n-digit number and 2) returning the middle n-digits of this 2n-digit number. If the squaring yields fewer than 2n digits, view it as having leading zeros.

Write a mutable abstract data type `MiddleSquare4.java` that implements the Middle Square Method for N = 4 according to this `MiddleSquare4` API:

`public class MiddleSquare4`	
`public MiddleSquare4(int seed)`	Creates a pseudorandom number generator with initial state `seed`. If `seed` is not between 1000 and 9999 inclusive, throws an `IllegalArgumentException`.
`public MiddleSquare4(MiddleSquare4 ms)`	Creates a new object with same state and seed as `ms`.
`public void reset()`	Resets the current state to the original state (i.e., `seed`).
`public int step()`	Computes the next PRN by squaring state and extracting digits 3 through 6 (i.e., the middle 4 of 8). Sets the state equal to the PRN and returns the PRN.
`public String toString()`	Returns `String` seed and state as `(<seed>, <state>)` Note the single space after the comma, for example, `(4094, 7609)`

The given `MiddleSquare4.java` contains a test client `main()` method. Feel free to add your own tests. We will not grade you on your `main()` method so long as it compiles.

Notes:

1. To extract digits 3 through 6 of a number, integer divide that number by 100 and use the remainder of dividing that by 10,000. Treating a number as having leading zeros requires no additional work. Examples:
    - 16**7608**36: 16760836 / 100 = 167608; 167608 remainder 10000 = 7608
    - 01**0020**01: 1002001 / 100 = 10020; 10020 remainder 10000 = 20 (thought of as 0020)
    - 00**0004**00: 400 / 100 = 4; 4 remainder 10000 = 4 (thought of as 0004)
2. The expected output of the given `main` is on page 4 of this exam.
3. You may create helper functions, but you may not need to do so.
4. You may use classes used in COS 126 (e.g., `String`, `StringBuilder`, `Character`, `Integer`, etc.)

Do not continue to Part II until you have completed, tested and submitted Part I.

Part II: (30%)

Everyone knows that the proper capitalization of the honor code is:

I pledge my honor that I have not violated the Honor Code during this examination.

Help your instructor create exam context by writing a tool for randomizing the case of letters to produce strings like this:

i pLEDGe My HOnOR tHat I HAVe NOt VIoLaTeD tHe hoNor coDE DUrInG THiS eXAMinaTION.

Using Middle`Square4` as a source of pseudo-randomness, write a mutable abstract data type `RandomCase.java` that implements `RandomCase` API:

`public class RandomCase`

`public RandomCase(int seed)`	Creates a `RandomCase` object using a `MiddleSquare4` object with initial state `seed`.
`public RandomCase(RandomCase rc)`	Creates a new `RandomCase` object with a deep copy of all state associated with the given `RandomCase` object.
`public void reset()`	Resets the state of the PRN generator to initial seed.
`public char convert(char c)`	If character `c` is a letter, get a new PRN. If that new PRN is an even number, return character `c` with the case changed. Otherwise, return `c` unchanged.
`public String convert(String s)`	Returns a new `String` with `convert` applied to all characters in the string `s`.

The given `RandomCase.java` contains a `main()` method test client that performs some tests. Feel free to add your own tests. We will not grade your `main()` method so long as it compiles.

Notes:

1. Note that the PRN generator state is not updated if the character to convert is not a letter.
2. Consider using `Character.toLowerCase` and `Character.toUpperCase`.
3. Two PRN generators with the same state produce the same sequence of PRNs. When both conversions start from the same PRN generator state, converting a converted string returns the original string.
4. The expected output of the given main is on page 4 of this exam.
5. You may create helper functions, but you may not need to do so.
6. You may use classes used in COS 126 (e.g., `String`, `StringBuilder`, `Character`, `Integer`, etc.)

Submit both `RandomCase.java` and `MiddleSquare4.java`.
You are responsible for submitting your solution for grading before time is called!

<u>The output of given `MiddleSquare4.main()` with your correct implementation of `MiddleSquare4`:</u>

```
3314
9825
5306
1536
3592
9024
4325
7056
7871
9526
7446
4429
6160
9456
4159
2972
8327
3389
4853
5516
4262
1646
7093
3106
6472
8867
6236
8876
7833
3558
6593
4676
8649
8052
8347
6724
2121
4986
8601
9772
4919
1965
8612
1665
7722
6292
5892
7156
2083
3388

toString should report "(2309, 3388)":  (note the single space after the comma)
(2309, 3388)
```

<u>The output of given `RandomCase.main()` with your correct implementation of `RandomCase`:</u>

```
i pLEDGe My HOnOR tHat I HAVe NOt VIoLaTeD tHe hoNor coDE DUrInG THiS eXAMinaTION.
I pledge my honor that I have not violated the Honor Code during this examination.
I pledge my honor that I have not violated the Honor Code during this examination.
```