Instructions

- This assignment is due on Tuesday, November 24, 2020 at 2:00 PM EST. Late submissions will **not** be accepted.
- This assignment consists of one problem with two parts. You must submit both parts to receive full credit.
- Your solution needs to be formatted using the LaTeX template available on OWL. Note that there are different templates available for regular assignments and group assignments. You should use the one for group assignments.
- All group members are expected to be working on the solution and every member should attend all group meetings.
- The Scribe will be submitting the assignment on behalf of the group. It is assumed that every member of the group has proofread the submission.
- All solutions must be written in full sentences.
- You are not allowed to use online resources and should only discuss the solution with members of your group.
- This assignment is worth 5 points.

Part 1.

In this assignment, we will investigate the importance of randomness in the choice of element r used in the ElGamal Digital Signature Algorithm. To this end, let A be Samantha's verification key and suppose that she signed two different documents D and D' using the same element r, producing signatures: (S_1, S_2) on D and (S_1', S_2') on D'.

1. Describe an algorithm that given a verification key A and two signatures (S_1, S_2) and (S_1', S_2') determines whether they were produced using the same random element.
2. If (S_1, S_2) and (S_1', S_2') were indeed produced using the same random element, describe an algorithm that finds the secret signing key a.

Part 2.

1. Write a function in Python3 called `solve` that, given the input $(p, g, A, D, D', S_1, S_2, S_1', S_2')$ where
   - p is a large prime;

- $g \in \mathbb{F}_p^*$;
- A is of the form $g^a \bmod p$;
- (S_1, S_2) and (S_1', S_2') are two valid signatures for documents D and D' (respectively) produced using the secret signing key a,

outputs `no` if (S_1, S_2) and (S_1', S_2') were produced using different random elements and outputs the secret signing key a if they were produced using the same random element.

Your program *must* implement the algorithms described in Part 1 of this assignment. All other functions will receive no credit.

2. Download the file `generate_input.py`, and use it to obtain a list of 10 tuples of the form $(p, g, A, D, D', S_1, S_2, S_1', S_2')$ by importing the file

```
from generate_input import generate_input
```

and running the function

```
generate_input("[last three digits of your student number]")
```

(Note the quotation marks.)

3. Run your method `solve` on all these inputs.

As part of your submission, include:

1. The *Python code* implementing your solution;
2. The 10 *inputs you generated*, and the *output of your program* run on these inputs. One input and one output per line.

Examples

Here are some examples of what your function `solve` should do.

```
>>> solve(31, 26, 30, 18, 11, 6, 24, 6, 25)
21
>>> solve(348149, 113459, 185149, 153405, 127561, 208913, 209176, 208913, 217800)
72729
>>> solve(4139, 32, 1644, 3782, 2220, 3776, 1722, 2924, 3616)
no
```

Notes

- Incorrect answers will be penalized more than missing answers. (It is straightforward to verify the correctness of your submission!)
- Make sure that your algorithm terminates on the inputs we provide.
- You may not use any trivial brute-force algorithms. You must implement the algorithm developed in Part 1 of the assignment.
- The file `generate_input.py` is written in Python3, and so should be your solution. Make sure you are using a 64bit version of Python3.
- Your code should not make use of any external libraries such as `numpy` or `math`. All the auxiliary functions should be implemented by you, and should be included in your submission. You should only use the most basic arithmetic operations such as `+`, `-`, `*`, `//`, `%`.
- Comments in the code are not mandatory. However in the case of an incorrect solution, the comments can provide grounds for partial credit.