Fall 2022 CIS 3362 Homework #7: Random Odds and Ends
Check WebCourses for the due date

1) (50 pts) It was mentioned in class that there is no fast solution to the Discrete Log problem. The only method discussed to solve it was to loop through all possible exponents x to see if $a^x \equiv b$ (mod p), given integers a and b and a prime number p. As long as multiplications and mods are assumed to take constant time, this algorithm solves the problem in O(p) time. However, there is a faster algorithm that is relatively simple (understandable within the scope of ideas taught in this class.) that cuts this run-time down to $O(\sqrt{p})$. Here is how the algorithm works:

Let n = $\lceil\sqrt{p}\rceil$. Then, if there is a valid solution x to the given discrete log problem, then there will exist integers c ($1 \le c \le n$) and d ($0 \le d \le n$) such that:

$$a^{nc-d} \equiv b \ (mod\ p)$$

Now, multiply this equation through by a^d:

$$a^{nc} \equiv ba^d \ (mod\ p)$$

Obviously, doing a double for loop through all possible values of c and d will have the same run time, O(p), as the original algorithm.

But, we can do better by **<u>storing a table (map in Java, dictionary in Python)</u>** which maps each answer of the form $a^{nc}\ (mod\ p)$ to the value of c that achieved it, and store this map in memory.

Then, for each value of d, we can iteratively compute $ba^d\ (mod\ p)$. For each of these answers, see if it is a key in the original map. If so, then this value of d "matches" with the output value c produced by the map, which means that the answer to the given query is simply **<u>nc – d.</u>**

Note: if none of the values produced in the second separate for loop produces a hit in the map, then there is no exponent, x, which satisfies the given query.

Let's look at a quick illustration with p = 11, a = 2 and b = 6

For this example n = $\lceil\sqrt{11}\rceil = 4$.

In our map, we first store 5 → 1, because $2^{4(1)} \equiv 5$ (mod 11).

For each subsequent value, we can take the previous value and multiply it by 5, since this is equivalent to 2^4 mod 11. So the rest of the values in our map would be:

3 → 2, because $2^{4(2)} \equiv 3$ (mod 11).

4 → 3, because $2^{4(3)} \equiv 4$ (mod 11), and

9 → 4, because $2^{4(4)} \equiv 9$ (mod 11).

Next, we start a variable = 6, the result we want. Since 6 isn't in the map, we multiply 6 by the base 2, to get 1 mod 11. This indicates that $6(2^1) \equiv 1$ (mod 11). (Notice that we are trying to match an answer of 5, 3, 4 or 9.) Next, we multiply this again by 2 to get 2, which is also not in the map. Next, when we multiply this by 2, we get 4, which is in the map. This indicates that $6(2^3) \equiv 4$ (mod 11), and our map tells us that $2^{4(3)} \equiv 4$ (mod 11), which means that it must be the case that $2^{4(3)-3} = 2^9 \equiv 6$ (mod 11), solving this instance of the discrete log problem.

Write two functions (in Python or Java) with the following prototypes:

<u>Python</u>

```
def slowDiscLog(base,ans,mod)

def sqrtDiscLog(base,ans,mod)
```

<u>Java</u>

```
public static long slowDiscLog(long base, long ans, long mod)

public static long sqrtDiscLog(long base, long ans, long mod)
```

Your functions should return the smallest non-negative integer x such that $\text{base}^x \equiv$ ans (mod mod). If no such integer exists, it should return -1. The functions will ONLY be tested on cases that work where mod is a prime number less than 2 billion, base is a primitive root of that prime number and ans is in between 2 and mod-2. (So, I will only test them on "regular" cases, so to speak, and no corner cases.)

The first method should run in O(mod) time, just iteratively exponentiating base and checking if the current value is answer.

The second method should run in $O(\sqrt{mod})$ time, using the algorithm described above.

Test them yourself on these test cases and provide a table of correct answer and run times of both of your methods for that test case.

Base	Ans	Mod	X	Time Slow	Time Fast
5	123456	1000000007			
5	87123456	1000000007			
211523205	1036204576	1999999973			
75853221	96317213	1450001227			
1003708272	1820444653	1910003723			
1204331962	505493879	1910003723			

2) (50 pts) Here is a cipher to break. It is encrypted using one of the classical schemes we learned about in class, which was tested in either Quiz 1 or Qujz 2. Good luck!

```
mrsrlbfmmgmxlweamktcfclsaggvgugmvnlxcpkhtkmxmzageaocmdcsrtgr
cvslgcrnpqcoaqhdhgcemhmhdhalgdcsadotblrsgmnhlkcsszhrgvkhgmoc
mncehruahgcogevbchxhungmtrcmrkagirvogmeuageavgqmnxckscimlprp
xrtnvpmdetnovulxxgmlurpmgdhavgrycshvharhgodmcecxcvvpconhcfvz
hafhdw
```

As always, discuss your whole process. What clues did you use to try to figure out what cipher was used. Describe what you did to try to break the cipher, and hopefully, determine both the plaintext and the key! **(If you do, please include these in the write up.)**

3) (50 pts) This is another cipher to break. It doesn't have a key, per se, but the system used is a "fixed" system, with the key embedded in it. In particular, the character at ciphertext position i (1 based) is shifted by f(i+1) characters, where f(i+1) is some function with an integer output. The function is one that is based on a famous number sequence. Good luck!

```
qtlrkvdxptjmvoeyiyhszegnrzovxzphplqojwiatpkiduoxeswjdupyqled
uqgeorzdqmmfgpjxsgzfvvbcljiixjjrewrldfbfaiadcsvgvvhqjemadoiw
mdghrjzqvbvtruwegmriiyftlnqgwaltnmriobayedsoqjjiajqmaqifntbd
shkxgajsovnnvtipwggxrn
```

As always, discuss your whole process. What clues did you use to try to figure out how each character was shifted. Describe what you did to try to break the cipher, and hopefully, determine both the plaintext and what the function f(x) used for creating the shift for each character is! **(If you do, please include these in the write up.)**