Diffie-Hellman

Figure out the shared secret agreed upon by Alice and Bob. This will be an integer.

The shared secret is 65.

Show your work. Exactly how did you figure out the shared secret?

Users > mshiffman > Desktop > Carleton > Year 4 > CS338 > diffiehellman.py

```
1
2   #tested A & B first using print instead of return to ensure there was only 1 possible i
3   def diffieSecret(g,p, X):
4       for i in range(p):
5           if g**i%p==X:
6               return i
7
8   def diffie(g, p, A, B):
9       x = diffieSecret(g,p,A)
10      y = diffieSecret(g,p,B)
11      aShared = B**x%p
12      bShared = A**y%p
13      if aShared==bShared:
14          return aShared
15      else:
16          return False
17
18
19  def main():
20      g = 7
21      p = 97
22      A = 53
23      B= 82
24      print(diffie(g,p,A,B))
25
26  main()
27
28
```

I first independently figured out Alice and Bob's secret numbers using the diffieSecret. Since the secret key for both must be lower than p and we are only given A and B, I wrote a for loop that checked every value up to p to see if that multiplied by g mod p gave us the A and B values sent. If the if statement on line 5 was true, then the math worked out and the secret key was returned.

I next used the diffie function to calculate the the shared secret by taking the A**secret key and % p and double checked using B**secret key % p to make sure the shared secret was the same, which it was.

Show precisely where in your process you would have failed if the integers involved were much larger.

If the integers involved were much larger, I would not be able to easily calculate Alice and Bob's secret numbers as there could be multiple values where g**i%p ==X.

RSA

Figure out the encrypted message sent from Alice to Bob.

Dear Bob, check this out. https://www.surveillancewatch.io/ See ya, Alice.

Show your work. Exactly how did you figure out the message? (You should include an explanation of how the message from Alice to Bob is encoded. That is, how does Alice's intended message (whatever manner of message it may be) correspond to the integers in the plaintext that you end up with after decrypting the encrypted message?)

Users > mshiffman > Desktop > Carleton > Year 4 > CS338 > RSA.py

```python

import sympy
import numpy as np

def findVals(n):
    for i in range(1,n):
        x = n/i
        if x%1 ==0:
            if sympy.ntheory.primetest.isprime(int(x)) & sympy.ntheory.primetest.isprime(i):
                return (int(x), int(i))


def findD(e, n):
    p, q = findVals(n)
    lambdaVal = np.lcm(p-1, q-1)

    d=0
    while True:
        if (e*d)%lambdaVal == 1:
            return d
        d+=1

def decode(e, n, message):
    d = findD(e,n)
    keyApplied = []
    for i in range(len(message)):
        y = message[i]
        newM = y**d % n
        keyApplied.append(newM)

    decryptedMessage = ""
    for j in range(len(keyApplied)):
        hexCode = hex(keyApplied[j])[2:]
        decryptedMessage += bytes.fromhex(hexCode).decode()
    return decryptedMessage


def main():
    bobE = 13
    bobN = 162991
    message= [17645, 100861, 96754, 160977, 120780, 90338, 130962, 74096,
              128123, 25052, 119569, 39404, 6697, 82550, 126667, 151824,
              80067, 75272, 72641, 43884, 5579, 29857, 33449, 46274,
              59283, 109287, 22623, 84902, 6161, 109039, 75094, 56614,
              13649, 120780, 133707, 66992, 128221]
    print(decode(bobE, bobN, message))

main()
```

I first used the helper function findVals to find the prime numbers that could be the p and q values that made up n. In the findD function I then used the numpy lcm function to find the lcm of (p-1, nq-1) as our lambda value. I then used this lambda value in the while loop to find a possible d value. In the decode function, I used the d value to decrypt the message by reversing the encryption done on the integers. I then used another for loop to loop through the values and convert the integers into hex codes, then add the hex codes converted to bytes converted to strings to decryptedMessage string (source: https://www.geeksforgeeks.org/convert-hex-to-string-in-python/) I originally thought the integers were ascii values but the decrypted message I got was in chinese so I then switched to hex codes, which worked.

Show precisely where in your process you would have failed if the integers involved were much larger.
If the integers involved were much larger, there would be multiple options for p and q values as there could be multiple sets of prime numbers that could be multiplied to create the n value. There could also be many different options for d as my program only goes with the first d value to work, but there could be many others that Bob could have chosen.

Explain, briefly, why the message encoding Alice used would be insecure even if Bob's keys involved larger integers.
Even if Bob's key involved larger integer values, it would still be insecure as Alice encoded the string letter by letter. Given that a pattern in letter frequencies could be guessed, even if larger integer values were used, it would still be an insecure way to encrypt messages.