CMPSC 177 Homework 3

Task 1

1. I obtained this using a short program that iterated from 2 through the square root of the given value N = 379447. If the current iterator % N was equal to zero, then the value of the iterator would be p and the quotient of N and p was q. In other words, q = N/p. The result of this is (613, 619). However, . So the product of 612 and 618 is 378216.
2. If the RSA public key has an N = 379447 and a public exponent of e = 7, then it's a simple matter of finding d. From the previous part, we know that , and , as given in the lecture. Thus, .

Task 2

Task 3

There are four main problems with the hash procedure provided. The first issue is with the hash function being used. MD5 generates hashes very quickly. This might seem like a good thing, but it's not because this allows a hacker to quickly generate hashes for possible passwords. Slower hashes might be inconvenient for a user, but it can be crippling to a hacker who has to calculate hashes for many possible passwords.

In addition, MD5 is deterministic. If I hash two passwords that are exactly the same, their hashes will be exactly the same. Thus, if there are any repeating patterns, or if multiple hashed passwords have the exact same hash, any hacker will know that those passwords/portions will be the same.

Now, there are ways around the deterministic issue, but this procedure implements none of them. There's no salt generated for the password, which helps protect against dictionary or rainbow table attacks by being tacked onto the password before being hashed. This would change the result of the hash, and since the hash is unique to each password, the resulting hashes would be different from each other and reveal less to any observers.

Finally, the procedure does not hash multiple times. This would increase the time it took to compare the hashes of potential passwords with the actual hash, assuming that they managed to hash the potential passwords the correct number of times. The point of securing passwords through hashes is to make the process of guessing it a slow and arduous slough that eats away at the attacker's time. This process fails to do so.

Task 4

The first password I managed to crack was Daenerys Targaryen's account, and the process I used was a dictionary attack using the word.txt file provided. I created programs in both python and C to do this, and, on their own, both of them failed. The process I used was simple. Go through the txt file one line at a time, use the target account's salt to encrypt it using the same crypt function that was in the example, and comparing the resulting hash with the target account's hash. If the hashes are the same, then we have our password. The problem was, this didn't work out as planned as no password was found, according to my program. So, I had the python version create an output file holding every single has for each word for the given salt. That took about 30 minutes. Once I had this output file, testing it on the Daenerys account information, I found that there was a match between her hash and a hash in the resulting file, on line 275943, by ctrl+f-ing her hash in the output file. The corresponding word in the word.txt file was overargumentative. I tried to log onto her account using the word as the password, and successfully gained entry.