CSC 110 – Lab 10
Cryptography Algorithms

Names: ____

Introduction

The purpose of this lab is to understand how certain encryption algorithms work and to practice implementing parts of these algorithms in Python.

Exercises

1) In class we looked at the Caesar cipher as a simple encryption algorithm.

   a) Why is the Caesar cipher NOT a good algorithm to use for Internet security?

   b) Given your answer to question a) above, write a function that can break the Caesar cipher. That is, if you are given the ciphertext, see if you can find a way to determine the plaintext. Use a brute force method and assume that a human can look at all of the possible values for the plaintext and decide which one is the most likely. The function signature for this exercise is:

```
def breakCaesar(ciphertext):
   # Given the ciphertext
   # Print all of the possible values for the plaintext
   # so that a human can see what the secret message might be
```

   Note: You may use code that already wrote in class this week that you can find on the Sample Code page of the Sakai site.

Once you have the function working, use it to crack the following secret message:

maxitllphkwbltuktvtwtukt

Instructor Initials: ____________________________

2) For this lab exercise, we will use the following code that simulates a very simple public key encryption algorithm.

`http://www.cs.uri.edu/~cingiser/csc110/labs/public_key.py`

This program is a simulation of how public key encryption would actually work. In a real scenario, the code for message sender A and message receiver B would be on different computers. Here we have it all in one program to demonstrate how it works. In the comments, where it says

```
# B sends the public key to A
```

and

```
# A sends ciphertext to B
```

there is no message actually being sent. This is meant to indicate that a message would be sent from B's computer to A's computer, and vice versa, in an actual scenario.

Take a look at the code to make sure you understand how it is implementing the public key encryption technique that we discussed in class and answer the following questions. For simplicity, we are assuming that the "secret" message is an integer being "sent" from A to B.

a) How is the message receiver (B) computing the private key?

b) What is the mathematical calculation that B is using to compute the public key?

c) What information does the message sender (A) need from B in order to encrypt the message?

d) What is the simple encryption algorithm used to encrypt a message that is made up of a single integer?

e) We have simplified the encryption process quite a bit for this exercise, assuming that the plaintext is a single integer. How could this program be modified so that it could encrypt a message made up of a string? What functions would have to change to do this?

f) Let's run the program and see what it does. When the program prompts you for a secret message, enter an integer.

- What is the ciphertext that A sends to B?

- Suppose this ciphertext were sent along the Internet from A to B, and you are sniffing the packets and you are able to intercept this message. Are you able to figure out what the secret key is using just the public information (public key, ciphertext, encryption algorithm)?

- Why are you able to figure out the secret key? Since you can figure out the secret key, what does this mean about this public key encryption algorithm? What would make a better algorithm?

.

Instructor Initials: ____________________________

Challenge Problems

1) We learned in class that the reason that the RSA algorithm is so powerful is that it is very hard to compute the prime factorization of very large numbers (watch the optional video for this week for more information). In class we wrote a brute-force prime factorization function. Modify that function so that it will be a bit smarter about how much work it does. Provide an analysis of how much work your algorithm does versus the one we wrote in class. You can find that code here:

http://www.cs.uri.edu/~cingiser/csc110/handouts/python/prime_factors.py

Instructor Initials: ____________________________

2) In the simple public key encryption program we looked at in exercise 2 above, we assumed that the input was a single integer. Modify the program so that it can encrypt a string using the same encryption algorithm for each character in the string.

Instructor Initials: ______________________________