**Homework 2**

Due: Lesson 12

(50 pts)

*1.* [5] We saw in class that the Shift Cipher is very weak, in general. Suppose the 26 keys of the Shift Cipher are used with equal probability 1/26 (yes, even k=0). Prove that if you only encrypt a single message of just one character, then no matter what the plaintext probability distribution is, the Shift Cipher has perfect secrecy. Use English to supplement your math (make your proof understandable). *Hint: How might the equation H(P) = H(P|C) be helpful?*

A cryptosystem has perfect secrecy if H(P|C) = H(P). Therefore, if we calculate H(P|C) and H(P) and they are equal, then the cryptosystem has perfect secrecy. First, we will calculate H(P). Because the key, K and the cipher text, C, have the same probabilities, we can set them equal to each other. This gives H(K)=H(C). Now for H(P|C). First we will find H(P,K,C), or the joint entropy of the plaintext, key, and cipher text, in two different ways. In the first, we find it is equal to H(P) + H(K) because they are independent from each other and knowing the plaintext and the key is equivalent to knowing the cipher text for a one-time pad. Because knowing P and C is also equivalent to knowing K for a one-time pad, we can simplify H(P,K,C) = H(P,C). Through the chain rule, H(P,C) = H(P|C) + H(C). Putting H(P,K) = H(P) + H(K) and H(P,C) = H(P|C) + H(C) along with knowing that H(K)=H(C), we can substitute and equate the equations get H(P|C) = H(P). Thus, the cipher has perfect secrecy.

2. [5] We have a cryptosystem with three plaintext letters ‘*a*’, ‘*b*’, and ‘c’ which occur with probabilities of 0.85, 0.10, and 0.05 respectively. There are three keys *k*1, *k*2, and *k*3 which occur with equal probability. The resulting encryption table for the three keys is as follows:

*a b c*

*k*1 C A B (In other words Ek1(*a*) C)

*k*2 A B C

*k*3 B C A

a. Calculate the entropy of the plaintext H(P).

b. Calculate the entropy of the plaintext, given the ciphertext H(P|C).

c. How much information does the ciphertext give you about the plaintext? What about if *k*3 encrypted ‘*a*’ to ‘*C*’ and ‘*b*’ to ‘*B*’ instead? How would that affect H(P|C) (answer theoretically not numerically)?

1. –(.85\*log2(.85)+.10\*log2(.10)+.05\*log2(.05)) = .748 bits
2. H(P|C) =

P(a|A)log(P(a|A))+P(b|A)log(P(b|A))+P(c|A)log(P(c|A))+ P(a|B)log(P(a|B))+P(b|B)log(P(b|B))+P(c|B)log(P(c|B))+

P(a|C)log(P(a|C))+P(b|C)log(P(b|C))+P(c|C)log(P(c|C)) =

P(a|All) = p(a,All)/p(All) = (.85\*.33)/.33 = .85

P(b|All) = p(b,All)/p(All) = (.1\*.33)/.1 = .1

P(c|All) = p(c,All)/p(All) = (.05\*.33)/.1 = .05

H(P|C) = 3(.85log2(.85) + . 1log2(.1) + .05log2(.05)) = 2.24 bits

1. The ciphertext gives us no information about the plaintext as all three keys don’t overlap in assigning ciphertext letters to plaintext letters. If *k3* encrypted ‘a’ to ‘C’ and ‘b’ to ‘B’ instead H(P|C) would decrease as it would increase the probability that ‘C’ translates to ‘a’ and ‘B’ translates to ‘b’.

3. [5] Find integers *x* and *y* such that 19x +103y = 1.

GCD(19,103) = 1 🡪 19\*x + 103\*y = 1 🡪19\*x = 1 (mod 103) 🡪 x = 38 🡪 19\*38 + 103\*y = 1 🡪103\*y = -721 🡪 y = -7

X = 38, y = -7

4. [5] Find 13-1 (mod 101) using the Extended Euclidean algorithm. Based on your results, what is 101-1 (mod 13)?

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Step-by-Step Solution | | | | | | | q | | | | x0 = 0 x1 = 1  xj =-qj-1 xj-1 + xj-2 | | | | y0 = 1 y1 = 0  yj =-qj-1 yj-1 + yj-2 | | | |
| 101 | = | 13 | x | 7 | + | 10 | | q1 | = | 7 | | X2 | = | -7 | | y2 | = | 1 |
| 13 | = | 10 | x | 1 | + | 3 | | q2 | = | 1 | | X3 | = | 8 | | y3 | = | -1 |
| 10 | = | 3 | x | 3 | + | 1 | | q3 | = | 3 | | X4 | = | -31 | | y4 | = | 4 |
| 3 | = | 1 | x | 3 | + | 0 | | q4 | = | 3 | | X5 | = | 101 | | y5 | = | -13 |

13-1 (mod 101) = -31, 70 101-1 (mod 13) = 4, -9

5. [5] Find gcd(30030,269) using the Euclidian algorithm

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Step-by-Step Solution | | | | | | | q | | |
| 30030 | = | 269 | x | 111 | + | 171 | q1 | = | 111 |
| 269 | = | 171 | x | 1 | + | 98 | q2 | = | 1 |
| 171 | = | 98 | x | 1 | + | 73 | q3 | = | 1 |
| 98 | = | 73 | x | 1 | + | 25 | q4 | = | 1 |
| 73 | = | 25 | x | 2 | + | 23 | q5 | = | 2 |
| 25 | = | 23 | x | 1 | + | 2 | q6 | = | 1 |
| 23 | = | 2 | x | 11 | + | 1 | q7 | = | 11 |
| 2 | = | 1 | x | 2 | + | 0 | q8 | = | 2 |

gcd(30030,269) = 1

6. [5] Find the inverse of 899 (mod 2968). Show all the steps.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Step-by-Step Solution | | | | | | | q | | | x0 = 0 x1 = 1  xj =-qj-1 xj-1 + xj-2 | | | y0 = 1 y1 = 0  yj =-qj-1 yj-1 + yj-2 | | |
| 899 | = | 2968 | x | 0 | + | 899 | q1 | = | 0 | X2 | = | 0 | y2 | = | 1 |
| 2968 | = | 899 | x | 3 | + | 271 | q2 | = | 3 | X3 | = | 1 | y3 | = | -3 |
| 899 | = | 271 | x | 3 | + | 86 | q3 | = | 3 | X4 | = | -3 | y4 | = | 10 |
| 271 | = | 86 | x | 3 | + | 13 | q4 | = | 3 | X5 | = | 10 | y5 | = | -33 |
| 86 | = | 13 | x | 6 | + | 8 | q5 | = | 6 | X6 | = | -63 | y6 | = | 208 |
| 13 | = | 8 | x | 1 | + | 5 | q6 | = | 1 | X7 | = | 73 | y7 | = | -241 |
| 8 | = | 5 | x | 1 | + | 3 | q7 | = | 1 | X8 | = | -136 | y8 | = | 449 |
| 5 | = | 3 | x | 1 | + | 2 | q8 | = | 1 | X9 | = | 209 | y9 | = | -690 |
| 3 | = | 2 | x | 1 | + | 1 | q9 | = | 1 | X10 | = | -345 | y10 | = | 1139 |
| 2 | = | 1 | x | 2 | + | 0 | q10 | = | 2 | X11 | = | 899 | y11 | = | -2968 |

899\*1139 + 2968\*-345 = 1. Thus, the inverse of 899 (mod 2968) is 1139.

7. [5] Find all possible solutions for x:

1. 14x ≡ 35 (mod 147)

14x ≡ 35 (mod 147) 🡪 gcd(14,147) = 7 🡪 14/7 ≡ 35/7 (mod 147) 🡪 2x ≡ 5 (mod 21) 🡪 gcd(2,21) 🡪 2-1\*2x ≡ 5\*2-1 (mod 21) 🡪 2-1 = 11 🡪 11\*2x ≡ 5\*11 (mod 21) 🡪 x ≡ 55 (mod 21) 🡪 X0 = 13 🡪 X0 + (kn)/d = 13 +k(147)/7 = 13 + k(21) = x 🡪 x = (13 + 21k)

1. 14x ≡ 40 (mod 147)

14x ≡ 40 (mod 147) 🡪 gcd(14,147) = 7 🡪 14/7 ≡ 40/7 🡪 40/7 is not an integer, thus the solution DNE.

8. [5] Find all possible solutions for x:

1. 𝑥2 ≡ 5(mod 31)

X = 6, 25

1. 𝑥2 ≡ 1(mod 24)

X = 1, 5, 7, 11, 13, 17, 19, 23

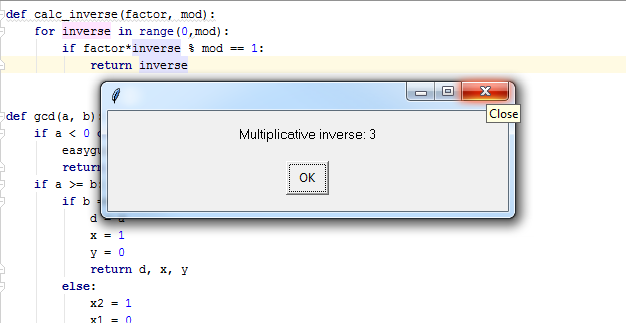
1. 𝑥2 ≡ 14(mod 17)

X DNE as there is no perfect square which has is congruent to 14 mod 17.

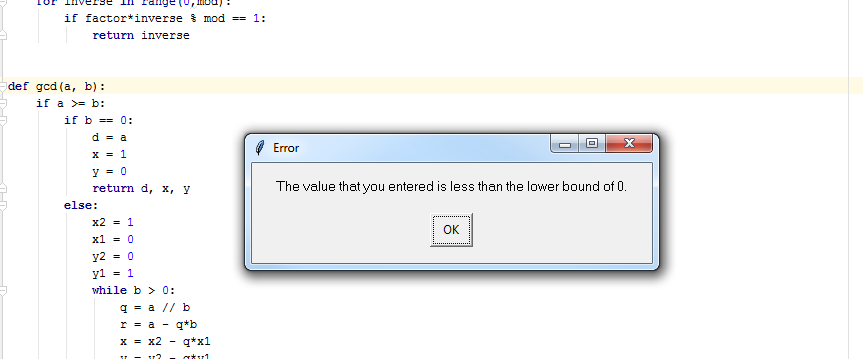
9. [10] Write a small program that implements the Extended Euclidean algorithm to find the greatest common divisor. The program should output the *gcd* and *x* and *y*. Input should be two non-negative integers a and b, with Your program should also provide an option to calculate the inverse of a number with some modulus for the user. Submit a screen shot showing all the working features of your program along with a print out of your code. . ba 

You can use the pseudocode below, but you don’t have to. You can use online sources, just be sure to document. The key is for you to have a tool on your computer that you understand how to use and are confident in its correctness.

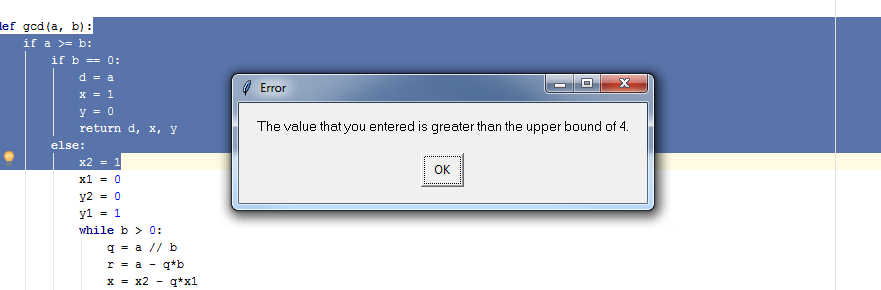
Calculating the multiplicative inverse of 7 mod 20.



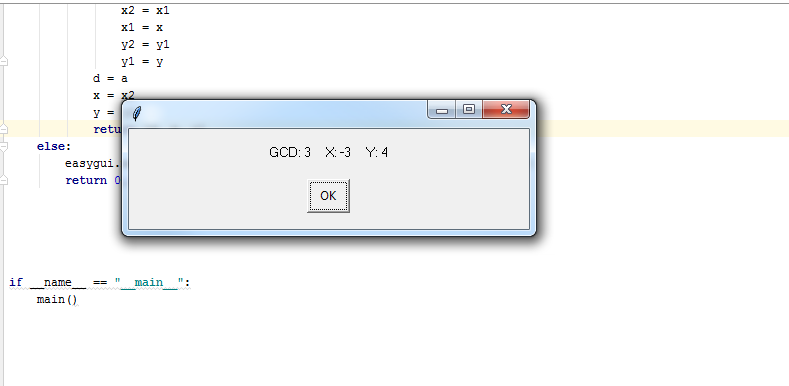
Ensuring the values are non-negative.



Ensuring a >= b



Calculating the GCD, X, and Y of 99 and 75.



**import** easygui  
  
  
**def** main():  
 choice = easygui.choicebox(**"Multiplicate Inverse or Extended Euclidean?"**, **"Main"**, [**"Multiplicative Inverse"**, **"Extended Euclidean"**])  
  
 **if** choice == **"Multiplicative Inverse"**:  
 factor = easygui.integerbox(**"Enter the factor you have"**, **"Multiplicative Inverse"**, **""**, 0)  
 mod = easygui.integerbox(**"Enter the mod integer"**, **"Multiplicative Inverse"**, **""**, 0)  
 easygui.msgbox(**"Multiplicative inverse: {:.0f}"**.format(calc\_inverse(factor, mod)))  
  
 **elif** choice == **"Extended Euclidean"**:  
 a = easygui.integerbox(**"Enter integer 1"**, **"Multiplicative Inverse"**, **""**)  
 b = easygui.integerbox(**"Enter integer 2"**, **"Multiplicative Inverse"**, **""**, 0, a)  
 easygui.msgbox(**"GCD: {:.0f} X: {:.0f} Y: {:.0f}"**.format(gcd(a,b)[0], gcd(a,b)[1], gcd(a,b)[2]))  
  
  
  
**def** calc\_inverse(factor, mod):  
 **for** inverse **in** range(0,mod):  
 **if** factor\*inverse % mod == 1:  
 **return** inverse  
  
  
**def** gcd(a, b):  
 **if** a >= b:  
 **if** b == 0:  
 d = a  
 x = 1  
 y = 0  
 **return** d, x, y  
 **else**:  
 x2 = 1  
 x1 = 0  
 y2 = 0  
 y1 = 1  
 **while** b > 0:  
 q = a // b  
 r = a - q\*b  
 x = x2 - q\*x1  
 y = y2 - q\*y1  
 a = b  
 b = r  
 x2 = x1  
 x1 = x  
 y2 = y1  
 y1 = y  
 d = a  
 x = x2  
 y = y2  
 **return** [d, x, y]  
  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()