Problem Set 1 -- SOLUTION

Due: Lesson 8

(50 pts)

Help Policy

AUTHORIZED RESOURCES: Any, except another cadet's program.

NOTE:

- Never copy another person's work and submit it as your own.
- Do not jointly create a program.
- You must document all help received from sources other than your instructor.
- **DFCS will recommend a course grade of F for any cadet who egregiously violates this Help Policy or contributes to a violation by others.**

1. [5] (variant of Exercise 5, pg 55)

a) Key is (α =19, β = 12).
b) The decrypted text is "CRUCIO".
c) This is a known plaintext attack.

We know plaintext 'c' (2) encrypts to ciphertext 'Y '(24), and plaintext 'r' (17) encrypts to ciphertext 'X' (23). This gives two equations in two unknowns:

$$2\alpha + \beta \equiv 24$$
$$17\alpha + \beta \equiv 23$$

Subtracting the second equation from the first gives:

$-15\alpha \equiv 1$ $-15 \equiv 11$ *on planet MOD 26 so*
$11\alpha \equiv 1$ *on planet MOD 26 this reads, 'what times 11 MOD 26 equals 1'?*
$\Rightarrow \quad \alpha \equiv 19$ *(found through trial and error)*

Plugging this back into the first equation gives

$2*19 + \beta \equiv 24$
$\beta \equiv -14$ *on planet MOD 26 this is …*
$\Rightarrow \quad \beta \equiv 12$

So the encryption function is

$$C \equiv 19P + 12 \pmod{26}$$

We actually need the decryption function, which must be the inverse of the above:

Since $C \equiv P\alpha + \beta$
$P \equiv \alpha^{-1} (C + \text{-}\beta)$ $\alpha^{-1} = 11$ $\text{-}\beta = 14$ *mult & add inverses*
=> $P \equiv 11 (C + 14)$ *MOD 26*

Which gives our decryption function. Applying this to "YXCYIS" gives "crucio".

2. [5] Exercise 6, page 55.

There is no advantage.

Let f(x) be the first affine cipher, let g(x) be the second. We have

$f(x) = \alpha x + \beta \pmod{26}$
$g(x) = \gamma x + \delta \pmod{26}$

The proposed double encryption function is the composite of the two:

$g(f(x)) = g(\alpha x + \beta) = \gamma(\alpha x + \beta) + \delta$

$= \gamma\alpha x + \gamma\beta + \delta$

$= (\gamma\alpha)x + (\gamma\beta + \delta)$

but this is just another affine cipher, with $\alpha' = \gamma\alpha$ *and* $\beta' = \gamma\beta + \delta$, *both mod 26. Thus this type of operation is equivalent to a single affine cipher, and therefore offers no advantage.*

3. *a. Key length is probably 2, because the most matches are likely to occur at displacements that are multiples of the key length, and a displacement of 2 generated the most matches.*
b. Key is probably {CA}

Ciphertext is C B C A B A A A C A.

At displacement 1:

```
C B C A B A A A C A
  C B C A B A A A C A     2 matches
        *  *
```

Displacement 2:

C B C A B A A A C A

C B C A B A A A C A 4 matches
* * * *

Displacement 3:

C B C A B A A A C A
C B C A B A A A C A 3 matches
* * *

So key length is probably 2.

Now do frequency analysis on the separate ciphers. First position: A:1, B:1, C:3. So 'a' likely encrypts to 'C', which means the first letter of the key is probably C. Second position: A: 4, B: 1, C: 0. So 'a' likely encrypts to 'A', which means the second letter of the key is probably 'A'. So the key is probably CA.

4. [5]
   a. *He is correct.*
   b. *If m and n are relatively prime, the pattern of encryption does not repeat itself until m*n letters. If they are not relatively prime, the pattern will repeat at their least common multiple. But it is indeed more secure, in the sense that in the relatively prime case the key length is longer and it will take slightly longer to crack.*

5. [5] (Exercise 1 page 343)

$H(X_1) = -\Sigma p(x) \log p(x)$
$= -(½ \log ½ + ½ \log ½)$
$= - (½ * -1 + ½ * -1)$
$= - (- ½ + - ½)$
$= 1$
same for $H(X_2)$.

$H(X_1,X_2) = -\Sigma\Sigma p(x,y) \log p(x,y)$
$= - (p(h,h) \log p(h,h) + p(h,t) \log p(h,t) + p(t,h) \log p(t,h) + p(t,t) \log p(t,t))$
$= - (4*(¼ \log ¼))$
$= - (4*(¼ * -2))$
$= - (4*(- ½))$
$= - (-2)$
$= 2$

$H(X_1) + H(X_2) = 2 = H(X_1,X_2)$

Both sides are equal because the two events are independent.

6. [5]
Yes, by removing the independence of the two events in some way. For example, if the first flip is heads, change the rules so that if the second flip is heads, the second coin is flipped again.

No. The joint entropy of two events is maximized when those events are independent, and can never exceed the sum of the individual entropies.

7. [5] (variant of Exercise 4, page 343.)

H(X) ≅ 11.5

Must calculate the probabilities for all the outcomes. Fortunately, it's pretty straightforward. There are 0..2^7-1 = 2^7 outcomes with probability = ½ / 2^7 = 2^{-8}. There are also 2^7..2^{14}-1 = very close to 2^{14} outcomes with probability ½ / 2^{14} = 2^{-15} . So the negative entropy (I prefer to take the negation at the end) is very close to

$$\begin{aligned} -H(X) &= 2^7 * (2^{-8} \log 2^{-8}) + (2^{14})*(2^{-15} \log 2^{-15}) \\ &= ½ * -8 + ½ \log (2^{-15}) \\ &= -4 - 7\,½ \\ &= -11\,½ \end{aligned}$$

So $H(X) \cong 11.5$

8. [5] (variant of Exercise 7, page 344)

(a) 2.741
If we replace the ball, then the two draws are independent and identical. So we can calculate the entropy of one draw and then double it.

$$-H(X) = 6/10 \log 6/10 + 2/10 \log 2/10 + 2/10 \log 2/10$$
$$2*H(X) \approx 2.741$$

(b) Entropy is decreased because we now have more information. Entropy is now 1.436.

Now we need to use the equations for conditional entropy since we are calculating the entropy of the second draw given knowledge of the first. Below are the calculations from a spreadsheet:

X={red, white, black}
Y={red, white, black}

Entropy of X given that Y is red:

x	P(X = x | Y = red)	log (P(X=x | Y=red))	product
red	0.56	-0.85	-0.47

white	0.22	-2.17	-0.48
black	0.22	-2.17	-0.48
			1.436

9. [10] Write a small program that loads in a text file of any size and then prints the frequency (as a percentage) of each character ('a'..'z'). All characters should be made lowercase for counting purposes. Ignore punctuation, spaces, etc. 1. Output should be printed out by expected frequency order (e,t,a,o etc) and look something like:

```
'e' – 13.27%
't' – 9.11%
'a' – 8.47%
'o' – 7.32%
…
'q' – 0.01%
'z' – 0.00%
```

Attach a printout of your code to your submission.

```
--Dr Barry Fagin
--CS431
--Frequency counter
with Ada.Text_Io; use Ada.Text_Io;
with Ada.Float_Text_IO; use Ada.Float_Text_IO;
with Ada.Characters.Handling; use Ada.Characters.Handling;
procedure Freq is
   subtype Char_Range is Character range 'a'..'z';
   Alphabet_Length : constant Integer := 26;
   Filename : constant String := "input.txt";
   Input_File : File_Type;
   C : Character;   --current character read from file
   Counters : array(Char_Range) of Natural := (others => 0);
   Freq : String(1..Alphabet_Length) := "etaoinshrdlcumwfgypbvkjxqz";
   Total_chars : Natural := 0;
begin
   Open(File => Input_File, Mode => In_File, Name => Filename);
   --EXCEPTION HANDLING WOULD GO HERE

   --Read the file and count occurrences of each character
   loop
      exit when End_Of_File(Input_File);
      Get(Item => C, File => Input_File);
      C := To_Lower(C);
      if (C >= 'a' and C <= 'z') then
         Counters(C) := Counters(C) + 1;
         total_chars := total_chars + 1;
      end if;
   end loop;
   --Print out the totals in expected frequency order
   --(easier than sorting)
   for I in 1..Alphabet_Length loop
      Put(Freq(I));
      Put("   ");
      Put(Item => Float(Counters(Freq(I)))/Float(Total_Chars),
         Fore => 0, Aft => 3, Exp => 0);
      New_Line;
   end loop;


end Freq;
```