CIS 3362 Homework 1
Monoalphabetic Ciphers

1) Given that the encryption function for an affine cipher in a language with 64 alphabet characters is f(x) = (37x + 18) % 64, determine the corresponding decryption function. Please show all of your work.

```
f(x) = 37x + 18 % 64
x = 37y + 18
x-18 = 37y
Need 37^-1 mod 64
Using Extended Euclidean algorithm...

64 = 1 * 37 + 27
37 = 1 * 27 + 10
27 = 2 * 10 + 7
10 = 1 * 7  + 3
7  = 2 * 3  + 1

7 - 2 * 3                = 1
7 - 2 * (10 - 1*7)       = 1
3 * 7 - 2 * 10           = 1
3 * (27 - 2*10) - 2 * 10 = 1
3 * 27 - 8 * 10          = 1
3 * 27 - 8 * (37 - 1*27) = 1
11 * 27 - 8*37           = 1
11 * (64 - 1*37) - 8*37  = 1
11 * 64 - 19*37          = 1 mod 64
-19*37                  = 1 mod 64
So 37^-1 mod 64 = -19
map into range: 64-19 = 45
37^-1 mod 64 = 45
45(x-18)= 45(37y) mod 64
45x - 810 = y
map into range: 64*13=832
      832 - 810 = 22
f^-1(x) = 45x + 22 mod 64
```

2) Encrypt the following message below using the affine cipher function, f(x) = (15x + 20) % 26.
PACKMYBOXWITHFIVEDOZENLIQUORJUGS
LUYOSQJWBMKTVRKXCNWFCHDKAIWPZIGE

For each character in the plain text, we take the integer representation of that char from 0-25, multiply it by a (15), add b (20) and mod by n (26), then we convert it back to the ascii character representation to get our ciphertext

see ‘h1.2.py’
Usage:
execute the program with 4 command line arguments, a, b, n and your plaintext string
ex:

```
$ python h1.2.py 15 20 26 PACKMYBOXWITHFIVEDOZENLIQUORJUGS
LUYOSQJWBMKTVRKXCNWFCHDKAIWPZIGE
```

3) Consider a language with an alphabet size of 145. How many possible affine cipher keys could there be for this language?

Factors of 145: 1, 5, 29, 145
List multiples of 5: 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55,
60, 65, 70, 75, 80, 85, 90, 95, 100, 105, 110, 115, 120, 125,
130, 135.... Count = 28
List multiples of 29: 29, 58, 87, 116.... Count = 4
Total count = 32
144 - 32 = 112
-> Total possible keys: 112 * 145 = 16,240 keys

4) You are attempting to break an affine cipher (in English). You believe that the ciphertext ‘Y’ maps to the plaintext letter ‘e’ and that the ciphertext ‘Z’ maps to the plaintext ‘t’. Determine the decryption function used based on these two pieces of information.

Since ‘Y’ and ‘Z’ are consecutive letters,
we can subtract ‘e’ - ‘t’ = 4-19 = -15
or 15 = a value
f(x) = 15x + b % 26
f(y=24) = 15(24) + b % 26 = 4
b = 8

Check:
f(z=25) = 15(25) + 8 % 26 = 19
True

$f^{-1}$(x)= 15x + 8 mod 64

5) Write a program that prompts the user to enter two uppercase alphabetic strings, where the first is a permutation of all 26 letters, representing a key to a substitution cipher and the second is a plaintext of uppercase letters only. Your program should output the ciphertext produced by encrypting the plaintext with the given key.
(Example, “QWRTYUIOPASDFEGHJKLZXCVBNM” and “CAT” should produce ciphertext “RQZ”.)

Pretty simple one here. The basic idea of a 26-alpha count substitution cipher is to find the location of a letter in the [A,B..Z] alphabet and then replace it with the letter in the same location of our permuted alphabet.

See ‘h1.5.py’
Usage:
execute the program with two command line arguments, your permuted alphabet string and your plaintext
ex:

```
$ python h1.5.py QWRTYUIOPASDFEGHJKLZXCVBNM CAT
RQZ
```

6) Write a program that allows the user to enter a and b for an affine cipher and determines how many times the function must be composed with itself to get back to the identity function. For example, if we have the encryption function f(x) = (5x + 4) mod 26, the answer is 3:

Not too bad here either. We keep track of our original x, the number of times we run the affine function, and x every time it goes through the affine function. After
See ‘h1.6.py’
Usage:
execute the program with 2 command line arguments, a and b
ex:

```
$ python h1.6.py 5 4
3
```

Decode the following ciphertexts. Please use the tools that I have provided off the course webpage and any tools you may create yourself. In your write-up, describe the steps you took and why you took them in decrypting the ciphertext. After your description, reveal the plaintext. PLEASE DO NOT USE ANY OF THE AUTOMATIC DECRYPTING WEBSITES OUT THERE. Due to this possibility, a majority of the credit will be given based upon your description of how you broke the cipher and not the answer, which will only be worth 20% of the total points.

7) (shift)
`yvpaluvufekdrbvzksrukrbvrjrujfexreudrbvzksvkkvi`
`heyjudedontmakeitbadtakeasodsongandmakeitbetter`

Since there are only 25 possible shifts from our current position in the shift, we could easily write a program to find all the permutations and sift through them by hand.

8) (affine)
`mggflgmfovgmgfevwoytgwgmmggflgflotvfkemfevwoytmengekiefzotwoy`
`seethestonesetinyoureyesseethethorntwistinyoursideiwaitforyou`

You could… Try all 312 possible decryption keys.
Or...
Use Arup's online Cryptanalysis Tool
http://www.cs.ucf.edu/~dmarino/ucf/cis3362/cryptotool.html
to find letter frequencies and find repeated N-grams, find something like FLG = THE and look for things such as double vowels (SEE)

The one that matches is encryption keys a = 19, b = 8 and the decryption keys a = 11, b = 16. If you can guess that the first word is "SEE", then you can set up the equations f(12) = 12a + b = 18 and f(6) = 6a + b = 4. Subtracting yields 6a = 14 mod 26, which is equivalent to 3a = 7 mod 13. Multiplying this by 9 we get a = 63 mod 13 or a = 11 mod 13. Thus, we have to try both a = 11 and a = 24. Of these, a = 11 leads to the correct solution when you cross reference with the other letters.

9) Substitution
dcssfntysqteudwsmtxyopqskpxundueopfdjdpymsndqpuncssfnthjpqsd
nqpuutnodulnksqpuljstonktxeknntjdeknmtxyupfsjdosndfscpysqpnq
kduexhvdnkfspjjnkscsqkpuescnprsduehjpqsdvdckdlcssunkshjpqsax
nuttuscsbsynprsufskspyncpulnktxekncnksmoplsoplspvpmkspyncpul
nktxekncnksmoplsoplspvpm

iseemtorecognizeyourfacehauntingfamiliaryeticantseemtoplacei
tcannotfindthecandleofthoughttolightyournamelifetimesarecatc
hingupwithmeallthesechangestakeingplaceiwishidseentheplacebu
tnoonesevertakenmeheartsandthoughtstheyfadefadeawayheartsand
thoughtstheyfadefadeaway

Using the cryptool again, calculate letter frequencies and n-grams.
Normal letter frequency for a single letter in a 26 character alphabet out of completely random text is 1/26, or around 3.8%
In the English language, the most common letters (over 8% avg. freq.) are E, T and A. Letters in the 6-8% range are ONSHR. We can try to match these letters up first and hopefully get lucky and go from there.
Let’s map SPN (our highest frequency letters in the ciphertext) to EAT, and try some mappings for HORNS as well. KTYUC mapped to HORNS (respectively) seem to match up nicely, and we start to see some words forming, ‘see’ ‘these’ ‘hearts’ ‘no one’.
We can fiddle out a few more letter matches from partial words, the C before ‘ANNOT’ a UG in the middle of ‘THO--HTS’.
The C, along with our previous matches, shows us ‘RECOGN--E’ in the first line, which we can guess to be ‘RECOGNIZE’. From here, a quick query to Google for “I see to recognize haunting” will lead to a nice auto-complete query: “i seem to recognize your face haunting familiar” and give us the remaining lyrics of the Pearl Jam song and complete our puzzle!

10) (substitution)
nqryjrkqnolqmbplbnsoblmholwmiylrnyesnmysekcfbrwsenqlsnqlbqme
pjmfoljfxbrnrekmepclnnlbablvilewrlymblwcsylnsmdlbmklmepnqryx
rccoldlbflmyfasbfsimccnsoblmhrkilyyxlccziynqmdlnsxmrnmepyllm
epziynnsnqbsxfsisaamornrccrewciplysjlxlrbpclnnlbyvirwhcfvilr
ncfzsqevirewfmpmjyyqsuurekmnnqlomttmbasbmtfksnlnrjlnsycllutt
tttt
This might be harder to break because it’s not a song lyric. On the other hand maybe my writing and letter frequencies are close to average and this will be very easy for you all to break I guess we’ll just have to wait and see. And just to throw you off a bit I’ll include some weird letters: quickly, quietly, John

Quincy Adams, shopping at the bazzar for a zygote. Time to sleep
zzzzzz

We can begin again by analyzing the ciphertext for letter frequencies and repeated n-grams. Since L has the highest frequency, we can make a reasonable guess at it mapping to E. The next highest letter was N, which we can map to T. Looking at our n-grams, we can see ‘NQL’ occur a few times, and try matching the Q to H to make ‘THE’ in our plaintext. This gives us a bit of a puzzle in the ciphertext with ‘NQLSNQLB’ which translates to THE-THE. Since it is a bit strange to have consecutive articles in an English sentence, we can try translating it to ‘THEOTHER’. Attempting to map the remaining vowels, we can maybe guess that the next highest frequency letter, R, might be I. This gives us an instance of ‘THI-’ in our plaintext, and put an S in that blank space to make ‘THIS’. We can see our plaintext begins to come together, and we can make some decent guesses at missing letters in mostly-complete words to easily map some more substitutions. From ‘-I-HT’ in our plaintext we could guess out ‘MIGHT’, and continue from there as the letters fall into place.