last name **first names** **id#**

cmput 210 W2016 **solutions: asn 1**

1. **keyphrase generation**

   The variant is more secure, because the original version will map more letters to themselves, and these mappings are easy for humans to notice.

   For the example given: the original maps 5 letters to themselves: `c v w x y z`, while the variant maps only `c` to itself.

2. **smoothing english** (i) homophones and substitutions

```
encipher           decipher      spelling errors after encipher/decipher:
a:       a j       j:   a        j q x z: now k
e:       e q x     q x: e
t:       t z       z:   t
j q x z: k

simple smoothing introduces spelling errors

humans can often correct these errors easier

than special purpose cryptogram crackers such as quipquip
```

   (ii) The original shift was 10, so to decrypt shift -10 or 16 would work. Removing spelling errors as described above gives the plaintext. Notice that to detect spelling errors you would need a dictionary, e.g. you would need to know to replace `frekuency` with `frequency`.

```
smoothing combined with substitution makes cracking harder

but not impossible since high frequency digrams can still be detected
```

   (iii) This is hard. First you need to use your methods for cracking cryptograms, i.e. mono-alphabetic substitution ciphers, knowing that the plaintext has been smoothed as above. If you are lucky, you will crack this, and then recognize that the keyphrase was `Al Kindi from Baghdad`, which gives the substitution below. Decrypting gives the smoothed plaintext. Unsmoothing gives the original. There were no `j q x z` in the original, so there are no final spelling errors to correct.

```
ptxt abcdefghijklmnopqrstuvwxyz
ctxt alkindfrombghejpqstuvwxyzc

qdgjr allan pox txasqd his rqadqrs for yejrs wizh his
     abilizy zo crack zhqir substitution ciphxrs, bqfore finally
     revqaling his mqthod in zhe gold bug

edgar allan poe teased his readers for years with his
     ability to crack their substitution ciphers, before finally
     revealing his method in the gold bug
```

3. **index of coincidence**

```
(i)   (4*3 + 2*1 + 4*3) / (10*9) = 13/45 = 0.289

(ii)  (2*1 + 2*1 + 2*1 + 2*1 + 2*1) / (10*9) = 1/9 = 0.111

(iii) (4*2 + 2*2 + 4*2 + 0*2 + 0*2) / (10*10) = 1/5 = 0.2

(iv)  (4/10)*0.081 + (2/10)*0.001 + (4/10)*0.027 = 0.0434

(v)   .081*.081 + .001*.001 + ... + .020*.020 + .001*.001 = .065
```

4. **cracking Vigenere** (i) any divisor of 96: `1 2 3 4 6 8 12 16 24 32 48 96`.

   (ii) are you joking? it's not even close. length 7 gives an average (over the 7 substrings) IOC of .038, which is almost exactly 1/26, which is what we would expect from random. length 8 gives an average (over the 8 substrings) IOC of .059, which is not far from .065, which is what we would get — with some variance expected — from typical English.

   (iii) If you consider the frequencies of all the letters, the shift which seems to best match English letter frequencies would be 21. This can also be seen by using the method from the lectures, by considering the most common English letters: a e h i n o r s t. Their offsets, starting from a 0, are shown below. The best shift turns out to be 21, matching the keyphrase letter v.

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
*       *     * *         * *     * * *
0 0 0 1 0 0 3 0 1 4 0 0 1 2 5 0 0 1 0 0 0 3 0 0 1 3
    * *         * *     * * *             *       *
```

   It's not too hard to crack using `friedman.py`, which guesses that the best key is `jovslace`. This is close to the correct key, namely `lovelace`. Using `vig.py -d` to decrypt reveals the original plaintext:

```
ada lovelace worked closely with charles babbage on his

analytical enginesi wonder whether she also discussed

cryptography with himher algorithmic notes on babbages

machines are considered by many to be the first computer programs
```