cmput210 a1 sol jan 2017 40 marks

2. (2 + 1 + 1 + 6) a) presumably, like most crytogram solvers, quipquip uses frequency analysis together with some word list. If nulls are inserted, then new words are created that are not in the word list. Eg. here `were less` becomes `werje less`, but the program has no way to know that `werje` should be understood as `were`.

   ```
   b)  abbasidcaliphswerjelessintjerestedthanthejirpredecesjsorsinconqjuest,
   prefjerringtogrowanorgajnizedanjdaffluejntsociety.
   ```

   ```
   c) abbasidcaliphswerelessinterestedthantheirpredecessorsinconquest,
   preferringtogrowanorganizedandaffluentsociety.
   ```

   ```
   abbasid caliphs were less interested than their predecessors in conquest,
   preferring to grow an organized and affluent society.
   ```

   d) There are many ways to solve this: here is one. First, run `freqs.py` to get letter counts:

   ```
    f  q  t  i  m  b  j  s  p  c  d  e  g  h  k  o  a  v  x  ,  r  y  z
   16 10 10  9  9  8  8  8  6  5  4  3  3  3  3  3  2  2  2  1  1  1  1
   ```

   Consider the nulls. If there are too few, then quipquip should recognize many words, and perhaps solve the cryptogram. So let's assume that the number of nulls is relatively large. So, one by one, let's see what happens if we guess that the null is `f q t i m` . . .

   From the above frequency table, we can see the frequencies we would get by removing f,q,t,i,m . . . respectively.

   ```
      f  q  t  i  m  b  j  s  p  c  d  e  g  h  k  o  a  v  x  ,  r  y  z
   f -- 10 10  9  9  8  8  8  6  5  4  3  3  3  3  3  2  2  2  1  1  1  1
   q 16 -- 10  9  9  8  8  8  6  5  4  3  3  3  3  3  2  2  2  1  1  1  1
   t 16 10 --  9  9  8  8  8  6  5  4  3  3  3  3  3  2  2  2  1  1  1  1
   i 16 10 10 --  9  8  8  8  6  5  4  3  3  3  3  3  2  2  2  1  1  1  1
   m 16 10 10  9 --  8  8  8  6  5  4  3  3  3  3  3  2  2  2  1  1  1  1
   ```

   Removing f leaves a distribution that is very even, whereas as the others have the 1st character (maybe plaintext e) significantly larger, which is what we expect. Let's guess that ciphertext `f` is plaintext `e` and that one of `q,t,i,m` is the null.

   For each guess, run the program freqs.py and gather data on letter pair frequency, and compare it to the data listed on the webpage: `http://homepages.math.uic.edu/~leon/mcs425-s08/handouts/char_freq2.pdf` In particular, which of the three guesses gives digrams of the form `?e` closest to that on the webpage, with the most common pair (corresponding to `he`) about 2.3 times the 2nd most common pair (corresponding to `re`)?

   For pairs `?f`, for guesses q,t,i respectively, the 4 most frequent counts are 3 2 2 1, 3 3 2 1, 4 2 2 1, so with the last guess we get something close to what we expect. So what if i is the null?

   ```
   tr -d 'q' < ciphertext > 2i
   ```

   ```
   baabtjcdbkjogtxfqfkfttjmsfqftsfcsgbmsgfjqoqfcfdfttpqtjmdpmrvfts,
   oqfefqqjmhsphqpxbmpqhbmjyfcbmcbeekvfmstpdjfsz
   ```

At this point, we could continue to solve the ciphertext using methods described in Singh (which will be a bit of a challenge, but doable, since we don't know where the blanks are). We could also send our ciphertext (with nulls removed) to quipquip and see what happens. It solves it quickly.

Another solution: for each possible guess of the null character, delete that character, and try quipquip. We will make at most 26 tries before we solve it.

Summary. The null is ciphertext i. Remove all nulls with `tr -d 'i' < ctxt`. Solve the resulting ctxt like you solve any cryptogram (or use quipquip). After adding nulls, the encryption function was `tr 'a-z' 'badcfehgjilknmporqtsvuxwzy' < ptpn`, a Kama Sutra encryption.

3. $(3+2+5)$ a) the only letter following q in English is u. From the letter pair data, we see there is only 1 ciphertext (ctxt) letter with this property: m is followed only by l. Also, the expected frequency of English q is 0.1%, which would be about 12 characters out of 12165. So, based on ciphertext single-letter frequencies, we would expect that one of o,y,r,m,g is ptxt q. This matches our guess that ptxt q is most likely ctxt m.

   b) Assuming ctxt m is ptxt q, the ctxt follower l of m should be ptxt u. Further evidence for this guess is that the ctxt frequency of l is 334/12165 $\approx$ .27, which is close to the expected frequency of u in English.

   c) here are ciphertext uvt-uvt pair frequencies and plaintext English eta-eta pair frequencies (from the table linked in the webnotes). From this data, the best matching is uvt-eta.

```
63  55 73     48  83 110
85  61 33     75  56  59
 0 148  0      0 104   1
```

   Another answer: argue as in the Singh example that the vowel among ctxt uvt is likely ctxt v (based on the number of letters that come before or after ctxt v), and then use the data to show that if ctxt v is ptxt a, then ctxt u is more likely than ctxt t to be ptxt e.

4. (2) frequencies `4t-3h-2e-1c-2a-1i-1n (14total)`, so `14!/(4!3!2!2!) = 151,351,200` transpositions

   frequencies `4t-3h-2e-4_-1c-2a-1i-1n (18total)`, so `18!/(4!4!3!2!2!) = 463,134,672,000` transpositions

5. (5) 26 (or 25 if you ignore the 0-shift, which does not disguise the plaintext at all).

   26! (or 26! - 1) $\approx 4 \times 10^{26}$ permutations of the alphabet $ab \ldots z$

   see the text

   otherwise ciphertext can be cracked by a brute force search of the key space

   there might be methods (eg. using frequency analysis) than can significantly reduce the search space

6. (3) see the text

7. (3)

```
l   y   s   a   n   d   e
 r   o   f   s   p   a   r
  t   a   u   s   e   d   a
   s   c   y   t   a   l   e
lrtsyoacsfuyasstnpeadadlerae

t      s     i     n     r     l
 r      p     t     s     a     e
  a      o     i     s     m
   n      s     o     c     b
tsinrlrptsaeaoismnsocb

l     y     s     a     n     d     e
 r     o     f     s     p     a     r
  t     a     u     s     e     d     a
   s     c     y     t     a     l     e
lysanderofspartausedascytale
```

8. (1) This is the story of the Spanish cryptographers and their ruler, King Philip II of Spain. see the text

9. (2) There is little evidence in Singh, either for or against, so we need to consult other sources, eg. the link *historical references* in the webnotes to the text *Mary Queen of Scots and the Babington plot. Edited from the original documents in the Public Record Office . . .*, also summarized briefly summarized on Wikipedia. Mary's two secretaries Claude Nau de la Boisselière and Gilbert Curle were taken into custody and interrogated: it seems that they told the interrogators what they wanted to hear. So there is some evidence, extracted under torture, that the offical version of events is correct.

   Note that MQS was not allowed to examine evidence or have anyone assist her in the preparation of a defence. Also, as a queen of a foreign land, she believed that — regardless of whether she had knowledge of any plot against Elizabeth — she was not subject to English law or courts.

10. (2) `vkupu zts t jlhwbe xlvepd tv vku kxppxp xi vkubp uyuelvbxf`

    `tr 'thequickbrownfxjmpsvlazydg' 'a-z' < ctxt` or `tr 'a-z' 'vigycnzbfphuqmkrdjsaetloxw' < ctxt`

11. (2) variance. a plaintext message can be considered a sample (with repetition) of words from a corpus. as a sample, there is likely to be some variance of any statistical measure of the corpus.

    something like this: using single-letter frequencies to guide your assignment, but backtracking as necessary, find a mapping of ptxt to ctxt that maximizes the number of ptxt words that appear