Problem 1

In the discussion on pp. 10-1 to 10-2 of the Bach notes, the author makes "a back of the envelope estimate for how much text is needed to separate E from the next most common letter T." A more precise statement would be that this estimate describes the length of text (denoted n) required to meet the following condition. Under the Gaussian approximation to empirical letter count probabilities (Bach, p 10-1), the empirical letter count frequency for the letter "T" will be less than 0.13 with probability 0.68. In the computations below, you are to impose a more demanding probabilistic requirement, that the desired conditions are met with probability 0.96. Again, you may assume use Gaussian approximation to empirical letter count probabilities.

a) What is the smallest acceptable value of text length, n, to ensure that the empirical letter count frequency for the letter "T" will be less than 0.13 with probability ≥ 0.96. Here the value 0.13 is the "ideal" frequency of occurrence of the letter "E."

Background for 1a and 1b:

n = length of a text

μ = mean # of occurrences of x $\Rightarrow np$

σ = fluctuation about μ (standard deviation) $\Rightarrow \sqrt{npq}$ (note: $q = 1-p$)

$\mu_E = 0.13n$ $\qquad$ $\mu_T = 0.092n$

$\sigma_E = \sqrt{0.13\cdot(1-0.13)\cdot n} = 0.34\sqrt{n}$ $\qquad$ $\sigma_T = \sqrt{0.092\cdot(1-0.092)\cdot n} = 0.29\sqrt{n}$

Note that ~96% of a Gaussian pdf is contained within 2σ

"Back of the envelope" estimate (Bach's notes 10-2)

So within a text of length n, we expect to see: (normalized over n)

E: $0.13 \pm 2\cdot 0.34/\sqrt{n}$ $\qquad$ T: $0.092 \pm 2\cdot 0.29/\sqrt{n}$

1.a.

Since we are finding n s.t. the empirical letter count frequency for "T" will be less than 0.13 w/ probability ≥ 96%, the difference in means will equal the smaller standard deviation when:

$$0.13 - 0.092 = 2\cdot 0.29/\sqrt{n}$$

Why?

Take "T" as our random variable

$$0.038 = 2\cdot 0.29/\sqrt{n}$$

$$n = \left(\frac{2\cdot 0.29}{0.038}\right)^2 \cong 233$$

b) What is the smallest acceptable value of text length, n, to ensure that the empirical letter count frequency for the letter "E" will be greater than 0.092 with probability ≥ 0.96. Here the value 0.092 is the "ideal" frequency of occurrence of the letter "T."

1.b. Here, the difference in means will equal the larger standard deviation

Take "E" as our random variable

$$0.13 - 0.092 = 2 \cdot 0.34/\sqrt{n}$$

$$0.038 = 2 \cdot 0.34/\sqrt{n}$$

$$n = \left(\frac{2 \cdot 0.34}{0.038}\right)^2 \approx 321$$

c) For monoalphabetic ciphers, authorized decryption is made easier by the ability to distinguish between characters based on empirical frequency of occurrence. This suggests why in monoalphabetic ciphers, the space character is not included in the character set (i.e., spaces between words are eliminated). You course notes observe that average word length in plain language English text is approximately 4.5 characters, which indicates that if the space character is included in the set of plaintext characters, its "ideal" frequency of occurrence would be p=(1/4.5)=0.2222.

Supposing that we expand the allowable character set of upper case English letters to include the space character (so we go from $\mathbf{Z}_{26}$ to $\mathbf{Z}_{27}$), and assume the resulting value of probability for the "E" character is reduced slightly, to become p = 0.1252. What is the smallest acceptable value of text length, n, to ensure that the empirical character count frequency for the space character will be greater than 0.1252 with probability 0.96?

$p_{\_} = 0.2222$

$\mu_{\_} = 0.2222n$

$\sigma_{\_} = \sqrt{0.2222(1-0.2222)n} = 0.4157\sqrt{n}$

$p_{E'} = 0.1252$

$\mu_{E'} = 0.1252n$

$\sigma_{E'} = \sqrt{0.1252 \cdot (1-0.1252)n} = 0.3309\sqrt{n}$

$$0.2222 - 0.1252 = \frac{2 \cdot 0.4157}{\sqrt{n}}$$

$$n = \left(\frac{2 \cdot 0.4157}{0.2222 - 0.1252}\right)^2 \approx 74$$

Problem 2

We return to the same construction as used in HW 3, Problem 3 last week. Consider again a limited alphabet of eight English language upper case characters {A, B, C, D, E, F, S, T}; associate the characters of this alphabet with $\mathbf{Z}_8$.

Again, we consider the eighteen-character long (n=18) ciphertext:

'TCEFTCCDSACBSDSACF'

a) Using the definition provided in lecture, construct the 8x8 digram matrix associated with this ciphertext; for reference, denote the resulting 8x8 matrix as DM. Confirm that the sum of entries across any row i of DM equals the count of the character associated with integer i in the set $\mathbf{Z}_8$.

2.a. From lecture, the i,j th entry of DM represents the number of transitions from letter i to letter j. Example: $DM_{0,2} = 2$

Note: i = row, j = column

The resulting digram matrix DM:

```
>> digraph_count8(int_message)

ans =

     0     0     2     0     0     0     0     0
     0     0     0     0     0     0     1     0
     0     1     1     1     1     1     0     0
     0     0     0     0     0     0     2     0
     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     0     2
     2     0     0     1     0     0     0     0
     0     0     2     0     0     0     0     0
```

Our limited character set associated with $\mathbf{Z}_8$. contains only two vowels, "A" and "E." Suppose we take as assumed that the most frequent character in the ciphertext (the character "C," or in the representation in $\mathbf{Z}_8$, the integer 2) will be a vowel correct plaintext. DO NOT assume anything further; in particular, do NOT assume you know that the encryption method is a shift cipher. Your goal below will be to use the DM matrix to try to identify the other character in the ciphertext that corresponds to a vowel in the correct plaintext.

b) Consider two orthogonal partitioning vectors, **u** and **c**; here these will be vectors of eight elements, each entry taking values {0, 1}, satisfying:

$v_i = 1$ if, in the integer representation of ciphertext, i corresponds to a vowel in the correct plaintext, 0 otherwise;
$c_i = 1$ if, in the integer representation of ciphertext, i corresponds to a consonant in the correct plaintext, 0 otherwise;
Based on the assumption above, you already know $v_2=1$, $c_2=0$; you are left with seven other possible choice for the other "1" entry in **v**, and that choice will completely determine the vectors **v** and **c**.

For seven choices of v,c evaluate the quantity: (this will be done in MATLAB)

$$\text{TEST\_DIFFERENCE}=(\mathbf{v}^T\text{DM}\mathbf{v})(\mathbf{c}^T\text{DM}\mathbf{c}) - (\mathbf{v}^T\text{DM}\mathbf{c})(\mathbf{c}^T\text{DM}\mathbf{v})$$

	v	c	TEST_DIFFERENCE
	$[10100000]^T$	$[01011111]^T$	5
	$[01100000]^T$	$[10011111]^T$	0
	$[00110000]^T$	$[11001111]^T$	-13
*	$[00101000]^T$	$[11010111]^T$	0
	$[00100100]^T$	$[11011011]^T$	-13
*	$[00100010]^T$	$[11011101]^T$	-46
	$[00100001]^T$	$[11011110]^T$	5

* Quiz questions.

2.c. TEST_DIFFERENCE is a metric that measures the "closeness" of our guess of v_i in the ciphertext being the other vowel in the plaintext. The choice of v,c with the largest magnitude TEST_DIFFERENCE most likely is the correct choice.
In particular, TEST_DIFFERENCE takes its largest magnitude value for $v=[00100010]$, which indicates with high probability that ciphertext characters C and S decrypt back to vowels in plaintext. Consulting the solution to HW3. problem 3a confirms this: C in ciphertext decrypts to E, while S in ciphertext decrypts to A.

Consider a character space of the uppercase English letters:

{A, B, C, D, E, F, G, H, I, M, N, R, S, T}

with corresponding representation on $\mathbf{Z}_{14}$

{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}

The 56-character (n=56) ciphertext below is constructed based this character set:

NETERECMGTISHENHGATGGCFICHISGSTGEFECMDEHHETHNTIARHEHCICG

This ciphertext was obtained from a plaintext message, over the same character space of $\mathbf{Z}_{14}$, encrypted with a monoalphabetic cipher. You may assume that the plaintext message is natural language English, and represents a grammatically correct, if slightly nonsensical, sentence.

Use any combination of decryption attacks considered thus far in 435, along with any heuristic judgements you may apply from your knowledge of the English language, to identify plaintext that you judge most probably correct (i.e., decrypt the message). Provide a short description of the attack algorithm(s) and any other techniques you employed.

Specify the monoalphabetic cipher employed to encrypt the message.

3. In class, we went over a few types of monoalphabetic ciphers:
   - shift *
   - affine *
   - monoalphabetic substitution

* Quiz question.

Question 7 — 1 pts

Out of the following, which are monoalphabetic ciphers?

- [ ] Vigenère
- [x] Caesar (shift)
- [x] Affine
- [ ] DES

Since we know the message was encrypted with a monoalphabetic cipher, we also know that there is a 1-1 mapping of a character in plaintext to a letter in ciphertext. Starting with a frequency count of the characters is a good starting point.

```
>> text = 'NETERECMGTISHENHGATGGCFICHISGSTGEFECMDEHHETHNTIARHEHCICG'

text =

    'NETERECMGTISHENHGATGGCFICHISGSTGEFECMDEHHETHNTIARHEHCICG'

>> alphabet = {'A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'M' 'N' 'R' 'S' 'T'}'

alphabet =

  14×1 cell array

    {'A'}
    {'B'}
    {'C'}
    {'D'}
    {'E'}
    {'F'}
    {'G'}
    {'H'}
    {'I'}
    {'M'}
    {'N'}
    {'R'}
    {'S'}
    {'T'}

>> counts = [alphabet cellfun(@(x) nnz(ismember(text,x)),alphabet,'un',0)]

counts =

  14×2 cell array

    {'A'}    {[2]}
    {'B'}    {[0]}
    {'C'}    {[6]}
    {'D'}    {[1]}
    {'E'}    {[9]}
    {'F'}    {[2]}
    {'G'}    {[7]}
    {'H'}    {[8]}
    {'I'}    {[5]}
    {'M'}    {[2]}
    {'N'}    {[3]}
    {'R'}    {[2]}
    {'S'}    {[3]}
    {'T'}    {[6]}
```

*Quiz question:

E, H, G, T, C
occur most frequently

Testing shift ciphers:

With our alphabet in $\mathbb{Z}_{14}$, there are only 14 possible shifts. It is fairly easy to check all of them in Matlab:

```
>> hw4_shift_bruteforce

text =

    'NETERECMGTISHENHGATGGCFICHISGSTGEFECMDEHHETHNTIARHEHCICG'

Shift: 0, Plaintext: NETERECMGTISHENHGATGGCFICHISGSTGEFECMDEHHETHNTIARHEHCICG
Shift: 1, Plaintext: MDSDNDBIFSHRGDMGFTSFFBEHBGHRFRSFDEDBICDGGDSGMSHTNGDGBHBF
Shift: 2, Plaintext: ICRCMCAHERGNFCIFESREEADGAFGNENRECDCAHBCFFCRFIRGSMFCFAGAE
Shift: 3, Plaintext: HBNBIBTGDNFMEBHEDRNDDTCFTEFMDMNDBCBTGABEEBNEHNFRIEBETFTD
Shift: 4, Plaintext: GAMAHASFCMEIDAGDCNMCCSBESDEICIMCABASFTADDAMDGMENHDADSESC
Shift: 5, Plaintext: FTITGTREBIDHCTFCBMIBBRADRCDHBHIBTATRESTCCTICFIDMGCTCRDRB
Shift: 6, Plaintext: ESHSFSNDAHCGBSEBAIHAANTCNBCGAGHASTSNDRSBBSHBEHCIFBSBNCNA
Shift: 7, Plaintext: DRGRERMCTGBFARDATHGTTMSBMABFTFGTRSRMCNRAARGADGBHEARAMBMT
Shift: 8, Plaintext: CNFNDNIBSFAETNCTSGFSSIRAITAESEFSNRNIBMNTTNFTCFAGDTNTIAIS
Shift: 9, Plaintext: BMEMCMHARETDSMBSRFERRHNTHSTDRDERMNMHAIMSSMESBETFCSMSHTHR
Shift: 10, Plaintext: AIDIBIGTNDSCRIARNEDNNGMSGRSCNCDNIMIGTHIRRIDRADSEBRIRGSGN
Shift: 11, Plaintext: THCHAHFSMCRBNHTNMDCMMFIRFNRBMBCMHIHFSGHNNHCNTCRDANHNFRFM
Shift: 12, Plaintext: SGBGTGERIBNAMGSMICBIIEHNEMNAIABIGHGERFGMMGBMSBNCTMGMENEI
Shift: 13, Plaintext: RFAFSFDNHAMTIFRIHBAHHDGMDIMTHTAHFGFDNEFIIFAIRAMBSIFIDMDH
```

Results show this probably wasn't encrypted with a shift cipher.

Testing with affine ciphers:

With our reduced alphabet size and relatively small n, we can't really assume the most frequently occurring letter in ciphertext will map back to E in plaintext. We will take the top 5 occurrences in the ciphertext and find two letters in the plaintext that are likely to map back to our ciphertext.

Note that if we want to map two letters from our top 5 occurrences we will have $_5P_2 = 20$ different choices!

Eventually after testing multiple ciphers, you should arrive at the following parameters: $a = 11, b = 4 \longrightarrow e_k(x) = 11x + 4$

Note: E in ciphertext does not map back to E in plaintext!

$$4 = 11x + 4 \mod 14$$
$$x = 0 \longrightarrow A$$

However, H in ciphertext (2nd most frequent letter) does map back to T...

$$7 = 11x + 4 \mod 14$$
$$x = 13 \longrightarrow T$$

Recovering the plaintext:

```
>> text = 'NETERECMGTISHENHGATGGCFICHISGSTGEFECMDEHHETHNTIARHEHCICG'

text =

    'NETERECMGTISHENHGATGGCFICHISGSTGEFECMDEHHETHNTIARHEHCICG'

>> int_message = letter_to_int14(text);
>> a_inv = mod_inv(11,14); b_inv = mod(-a_inv*4,14);
>> pt_ints = mod(a_inv*int_message+b_inv,14);
>> int_to_letter14(pt_ints)

ans =

    'SARAHANDERICTASTEGREENMINTICECREAMANDFATTARTSRIGHTATNINE'
```