Assignment 1

Name: Hongzhi Fu Student ID: 1058170

1. (a) $p = D_{(a,b)}(c) = a^{-1}(c - b) \bmod 36$

   (b) Since the condition is $c = p$ for all input p, we can assume that when $p = 0$, then we have $b \bmod 36 = p = 0$. Since $0 \leq b \leq 35$, b should be equal to 0. Thus, we have the equation:

   $$ap \bmod 36 = p \quad \text{for all } p$$

   In this case, the equation is true if and only if $a = 1$. Hence, we conclude that there is only one non-trivial key that satisfies $c = p$ for all input p.

   (c) The cipher is mono-alphabetic cipher, because there is only a single cipher alphabet available per message. In other word, all identical plaintext characters are mapped to the same ciphertext character. E.g. if we fix the keys a and b, and the letter E is mapped to letter Y, then all letters E in a single message will be mapped to letter Y, thus it is a mono-alphabetic cipher.

   (d) Since a large amount of ciphertext characters is given, the attacker can use statistic scheme to count each letter's relative frequency and list those with high frequencies. Then assume that those ciphertext characters are encrypted by high-frequency ones, after which the attacker can simply enumerate all possible keys to match the above assumptions, in this case 432 possibilities (12 possibilities for a, since there are only 12 numbers less than and relative prime to 36; 36 possibilities for b), and finally, use these keys to decrypt ciphertext to observe whether the plaintext has any meaningful information.

   (e) Since the key space is not very large, an efficient way to retrieve the key is obtain some plaintext-ciphertext pairs using the given oracle, then search the goal key from the key space iteratively until all pairs are matched by that key.

2. From my perspective, integrity and availability are the most concerns for the public. In terms of integrity, since the main purpose of COVIDSafe app is to identify people who are tested positive and let other people who have close contact with them protect themselves, the information of active cases must be authentic and reliable, so as not to mislead the public. While for availability, our goal is to find those who have close contact with active cases as quickly as possible to slow down the spread of virus, the app should send out the most recent and timely information about infected people. Confidentiality is also important, because the app must protect privacy of registered users against any opponents who inspect their personal information, but in this app, it is not the most crucial aspect to be considered.

3. (a)

```
def extended_gcd(a, b):
    x_im2, y_im2 = 1, 0
    x_im1, y_im1 = 0, 1  # initialize x_i-2, x_i-1 and y_i-2, y_i-1
    x_i, y_i = 0, 0  # initialize x_i, y_i
    while True:
        r = a % b  # remainder
        q = a // b  # quotient
        if r == 0:
            if b != 1:
                return None  # not solvable
            return x_i, y_i
        x_i = x_im2 - q * x_im1
        y_i = y_im2 - q * y_im1
        x_im2 = x_im1; y_im2 = y_im1
        x_im1 = x_i; y_im1 = y_i  # update x_i-2, x_i-1 and y_i-2, y_i-1
        a = b; b = r  # update a and b
```

(b)

```
def inverse_modulo(a, n):
    if extended_gcd(a, n):
        x, y = extended_gcd(a, n)
        return x % n
    return None  # not solvable
```

(c) My student number is 1058170, so after revoking the function `inverse_modulo` parameterized by `a` and `n` which are 1058170 and 16777259 respectively, the result is 8694250.

4.

(a) The number of possible keys in hill cipher is the number of invertible matrices formed by these numbers. To determine this value, the fact we should know is the condition of an invertible matrix is its determinant should not be 0. The zero determinant implies that each row/column must be linearly independent. Taking the first column into consideration, each entry has 37 possible values to be taken, except all 0 case, thus the number of first column is $37^m - 1$ possibilities. The second column, however, has $37^m - 37$ possible values, since we should avoid being linearly dependent with the first column, which has 37 possibilities. The latter column is similar but have to consider independency or more previous columns, so we conclude a formula:

$$|K| = \prod_{i=0}^{m-1}(37^m - 37^i)$$

Since the key size 37 is a prime number, i.e. no number factors out from a matrix, so there are no redundant matrices. Hence the number of possible keys in this specification is $\prod_{i=0}^{m-1}(37^m - 37^i)$.

(b) The first step is replace ????? with the last five digits of my student ID, which is 58170 and map each character in both plaintext and ciphertext to number, with the rule {A: 0, B: 1, …, 9: 35, _: 36}, then we form two matrices $\mathbf{X}$ and $\mathbf{Y}$, where each row of $\mathbf{X}$, $\mathbf{X}_{i,:}$, which is the plaintext with five characters, corresponds to each row of $\mathbf{Y}$, $\mathbf{Y}_{i,:}$. Since the key for encryption is unknown, we can simply construct an equation:

$$\mathbf{K}_E\mathbf{X} = \mathbf{Y}$$

Concretely, we have:

$$\begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & k_{15} \\ k_{21} & k_{22} & k_{23} & k_{24} & k_{25} \\ k_{31} & k_{32} & k_{33} & k_{34} & k_{35} \\ k_{41} & k_{42} & k_{43} & k_{44} & k_{45} \\ k_{51} & k_{52} & k_{53} & k_{54} & k_{55} \end{bmatrix} \cdot \begin{bmatrix} 23 & 32 & 20 & 7 & 27 \\ 35 & 9 & 4 & 8 & 33 \\ 1 & 0 & 24 & 22 & 26 \\ 32 & 22 & 33 & 31 & 31 \\ 19 & 29 & 5 & 34 & 25 \end{bmatrix} = \begin{bmatrix} 28 & 25 & 27 & 3 & 7 \\ 16 & 27 & 33 & 13 & 8 \\ 31 & 25 & 26 & 24 & 13 \\ 35 & 31 & 20 & 28 & 19 \\ 25 & 34 & 12 & 9 & 18 \end{bmatrix}$$

To find the matrix $\mathbf{K}_E$, we simply take the modulo inverse of $\mathbf{X}$ and multiply matrix $\mathbf{Y}$:

$$\mathbf{K}_E = \mathbf{Y}\mathbf{X}^{-1}$$

To find the modulo inverse of $\mathbf{X}$, we use Python Numpy library function *np.linalg.inv* and *np.linalg.det* to get the inverse and determinant of matrix $\mathbf{X}$, inv($\mathbf{X}$) and det($\mathbf{X}$), then we compute the modulo inverse of det($\mathbf{X}$) by invoking the function `inverse_modulo(a, n)` in question 3 and the result is 1, Finally, we obtain the modulo inverse of $\mathbf{X}$.

$$\mathbf{X}^{-1} = (\det^{-1}(\mathbf{X}) \bmod 37) * \det(\mathbf{X}) * \text{inv}(\mathbf{X})$$

$$= \begin{bmatrix} 14 & 23 & 18 & 6 & 29 \\ 6 & 22 & 0 & 32 & 21 \\ 3 & 1 & 11 & 1 & 2 \\ 14 & 31 & 33 & 20 & 21 \\ 19 & 36 & 31 & 36 & 12 \end{bmatrix}$$

Finally, we obtain key for encryption by using *np.dot* function:

$$\mathbf{K}_E = \mathbf{Y}\mathbf{X}^{-1} \bmod 37$$

$$= \begin{bmatrix} 28 & 16 & 31 & 35 & 25 \\ 25 & 27 & 25 & 31 & 34 \\ 27 & 33 & 26 & 20 & 12 \\ 3 & 13 & 24 & 28 & 9 \\ 7 & 8 & 13 & 19 & 18 \end{bmatrix} \cdot \begin{bmatrix} 14 & 23 & 18 & 6 & 29 \\ 6 & 22 & 0 & 32 & 21 \\ 3 & 1 & 11 & 1 & 2 \\ 14 & 31 & 33 & 20 & 21 \\ 19 & 36 & 31 & 36 & 12 \end{bmatrix} \bmod 37$$

$$= \begin{bmatrix} 21 & 12 & 7 & 12 & 21 \\ 5 & 21 & 33 & 24 & 23 \\ 24 & 22 & 4 & 36 & 27 \\ 9 & 25 & 32 & 24 & 6 \\ 22 & 5 & 31 & 6 & 18 \end{bmatrix}$$

To decrypt a ciphertext by a decryption matrix, all we need to do is take the modulo inverse of encryption matrix $\mathbf{K_E}$ to get $\mathbf{K_D}$.

$$\begin{aligned} \mathbf{K_D} &= \mathbf{K_E}^{-1} \bmod 37 \\ &= (\det^{-1}(\mathbf{K_E}) \bmod 37) * \det(\mathbf{K_E}) * \mathrm{inv}(\mathbf{K_E}) \\ &= \begin{bmatrix} 31 & 6 & 8 & 14 & 32 \\ 30 & 10 & 28 & 31 & 15 \\ 3 & 19 & 31 & 15 & 5 \\ 1 & 19 & 7 & 13 & 7 \\ 12 & 31 & 11 & 14 & 26 \end{bmatrix} \end{aligned}$$

When decryption, we need to partition ciphertext into pieces, each with 5 characters, and multiplied by decryption matrix $\mathbf{K_D}$ to get plaintext of each piece. After calculation, our plaintext is:
98KJ8LUX(space)906S6(space)4AF3LFP2OZQCI3RITF2QMBWA1TMHB01A0WCO1319

5. (a) My student ID is 1058170, with the fourth digit 8 and sixth digit 7 ($x = 8, y = 7$), so the value N is $5x + 6y + 15 = 97$.

(b) Since each digit number of student ID is ranged between 0 to 9 and it must be an integer, we only find all possible x and y such that $5x + 6y + 15 = 97$, and the result shows there is only one combination of x and y, which is my student ID's fourth and sixth digit(8 and 7). For each individual student, there are 100 possible combinations of fourth and sixth digit of his/her ID, so the probability of this student not sharing the same N is $\frac{99}{100}$. Any two students' ID is independent on each other, thus we conclude that the probability of all 230 students not sharing the same N with me is $\left(\frac{99}{100}\right)^{230}$. Finally, the probability of at least one of the 230 students sharing the same N is $1 - \left(\frac{99}{100}\right)^{230}$, which is about 0.901.

(c) Since each ball and bin is not identical, for each ball, it can be put into any of the five different bins. The total number of methods of placing is $5^N = 5^{97} \approx 6.311 \times 10^{67}$.

(d) Since the constraints are all bins are not allowed to be empty and all balls are identical, we can reduce this problem to partition N balls into 5 sections and guarantee each partition has at least 1 ball. Thus, we place N balls in a row and here we have $N - 1$ possible insertion sites for partition and the only job we will do is select 4 insertion sites from $N - 1$ ones, so the possible ways is $C_{N-1}^{5-1} = C_{96}^{4} = 3321960$.

(e) Since in this case, empty bin is tolerable, we can assume that each bin has one ball and allocate the remaining N balls to those bins, i.e. our question here becomes place $N + 5$ balls into 5 bins which subjects to all bins are non-empty. Hence, we have $C_{N+5-1}^{5-1} = C_{101}^{4} = 4082925$ ways of allocation.

(f) Let function $f(n, m)$ be the number of ways of placing n identical balls into m identical bins subject to any empty bins are not tolerable. To solve this function, we have two conditions:

i) $n < m$: When the number of balls is smaller than the number of bins, we can simply ignore superfluous $m - n$ bins, because no matter how n balls are arranged, there are always at least $m - n$ remaining bins, thus we have:

$$f(n, m) = f(n, n) \quad \text{for } n < m$$

ii) $n \geq m$: In this case, we divide it into two cases. One is put one ball into each bins first, and consider how to put $n - m$ balls into m bins. Another case is ignore 1 bin and find how many ways are there to place n balls into $m - 1$ bins, When combining these two cases, we have:

$$f(n, m) = f(n - m, n) + f(n, m - 1) \quad \text{for } n \geq m$$

The initial condition is $f(n, 1) = 1$ and $f(0, m) = 1$, thus we have a recursive relation:

$$f(n, m) \begin{cases} f(n, n) & \text{for } n < m \\ f(n - m, n) + f(n, m - 1) & \text{for } n \geq m \\ 1 & \text{for } n = 0 \text{ or } m = 1 \end{cases}$$

In this specification, it allows us to have at most 2 non-empty bins, so we only consider $f(N - 2, 2)$ and $f(N - 1, 1)$, which is $f(95, 2) + f(96, 1) = 49$.