Problem 1. Simple questions

1. Given ciphertext "EVIRE" encrypted by Caesar cipher, assume the key is x bounded from 0 to 25. Then I wrote following codes to see if I'll see some meaningful results.

```
input = "EVIRE"
outputs = []

for key in range(0,26):
    plaintext = ""
    for character in input:
        number = ord(character) - key
        if number < 65:
            number += 26
        plaintext += chr(number)
outputs.append(plaintext.lower())
```

   And now we have 26 possible results corresponding to the conditions where key = 0 to 25, listed as follows:

```
['evire', 'duhqd', 'ctgpc', 'bsfob', 'arena', 'zqdmz', 'ypcly',
↪  'xobkx', 'wnajw', 'vmziv', 'ulyhu', 'tkxgt', 'sjwfs', 'river',
↪  'qhudq', 'pgtcp', 'ofsbo', 'neran', 'mdqzm', 'lcpyl', 'kboxk',
↪  'janwj', 'izmvi', 'hyluh', 'gxktg', 'fwjsf']
```

   Finally, we can deduce that the secret locaton has to be "arena" or "river", since the arena is kind of a public place, I suggest Bob go to the river to meet Alice.

2. Given plaintext is "dont" and ciphertext to be "ELNI", we know the encrption matrix can have the size 2x2. Therefore we set the key to be

$$K = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

   According to encryption of Hill Cipher, we have

$$\begin{pmatrix} 3 & 14 \\ 13 & 19 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} \equiv \begin{pmatrix} 4 & 11 \\ 13 & 8 \end{pmatrix} \mod 26$$

   Multiply A inverse on the two sides,

$$K \equiv \begin{pmatrix} 3 & 14 \\ 13 & 19 \end{pmatrix}^{-1} \begin{pmatrix} 4 & 11 \\ 13 & 8 \end{pmatrix} \mod 26$$

   Then we compute the inverse of A as

$$A^{-1} = \frac{Adj(A) = \begin{pmatrix} 19 & -14 \\ -13 & 3 \end{pmatrix}}{\det \begin{pmatrix} 3 & 14 \\ 13 & 19 \end{pmatrix} = -125}$$

We find that

$$(-125) \cdot (-5) \equiv 1 \bmod 26$$

Therefore, we have finally

$$K \equiv \begin{pmatrix} -95 & 70 \\ 65 & -15 \end{pmatrix} \begin{pmatrix} 4 & 11 \\ 13 & 8 \end{pmatrix} \bmod 26 = \begin{pmatrix} 530 & -485 \\ 65 & 595 \end{pmatrix} \bmod 26$$

$$K = \begin{pmatrix} 10 & 9 \\ 13 & 23 \end{pmatrix}$$

3. Given that $n \mid ab$, we have $n = abk$ where k is an integer.
Suppose $n \nmid b$, we let $b = pn + q$, where p and q are two integers. Then we have

$$a(pn + q) = kn$$

$$aq = (k - ap)n = \frac{aq}{n}n$$

Since $k - ap$ is an integer, $\dfrac{ad}{n}$ must be an integer. However, given $\gcd(a, n) = 1$, we know here is a contradiction, thus our assumption is wrong, and we have $n \mid b$.

4.

$$\begin{aligned} 30030 &= 116 \times 257 + 218 \\ 257 &= 1 \times 218 + 39 \\ 218 &= 5 \times 39 + 23 \\ 39 &= 1 \times 23 + 16 \\ 23 &= 1 \times 16 + 7 \\ 16 &= 2 \times 7 + 2 \\ 7 &= 3 \times 2 + 1 \\ 2 &= 2 \times 1 \end{aligned}$$

From above, we have gcd(30030,257)=1.
Since $\sqrt{257} = 16.03$, so the factors of 257 are 2, 3, 5, 7, 11, 13. We can simply show all the results that $257 \bmod 2 = 1$, $257 \bmod 3 = 2$, $257 \bmod 5 = 2$, $257 \bmod 7 = 5$, $257 \bmod 11 = 4$, $257 \bmod 13 = 10$. Therefore, 257 is prime.

5. For OTP, the key can be obtained by XOR operation on plaintext and ciphertext. Therefore, if the attacker obtain the plaintext and ciphertext, he/she can also get the key. If this key is used for the second time, the attacker can easily fake the ciphertext, which is very dangerous.

6. Since secure means that the attacker has to compute at least 2^{128} operations to break the encryption, for the base of 2, it suffices to calculate

$$\sqrt{n \log n} \geqslant 128$$

$$n \geqslant 1546.4$$

Therefore, a graph with a size of 1547 should be used to be secure.

Problem 2. Vigenère cipher

1. Vigenère cipher is a encrypting method based on Caesar cipher, what differs from it is that the plaintext is encrypted not by shifting the same interval of original character, but with various shift values determined by corresponding key character. The full transformation sequence can be looked up with the help of Vigenère Table. Notice that the key length has to be the same with the plaintext length, when multiple keys are chosen, the effective key length is the least common multiple of the lengths of the individual keys.
   Speaking of decryption, the key is to determine the length of the key and then the problem becomes the one similar as the Caesar one, one can also validate the decryption by using Vigenère Table, just simply go to the row in the table corresponding to the key, find the position of the ciphertext letter in that row and then using the column's label as the plaintext.

2. (a) Because the key and one loop of ciphertext encrypted by Vigenère cipher has the same length if the plaintext is the same letter, and it's obvious to find that the ciphertext is looping over for several hundreds times, thus Eve can suspect that the plaintext must have also looped several hundreds times.

   (b) The key length simply equals the length of the loop occurs in the ciphertext, which is 6.

   (c) We know that in english, no word of length six is a shift of another English word, this ensures the key we want to find is unique. We can pick the loop in the ciphertext, and test 26 characters to look up the corresponding 6-letter word by Vigenère Table. When Eve see there is a meaningful word, she finds the key.

Problem 3. Programming

1. We first use GMP random number functions to create two random 4096 bit integers.

```
#include <gmpxx.h>
#include <iostream>
#include <ctime>

int main(){
    gmp_randclass r1(gmp_randinit_default);
    r1.seed((unsigned long)(time(NULL)));

    mpz_class a = r1.get_z_bits(4096);
    mpz_class b = r1.get_z_bits(4096);

    return 0;
}
```

2. The Implementation of Extended Euclidean Algorithm.

```
// Input: a,b. Two positive mpz_class integers(4096 bit)
// Output: r1 = gcd(a,b) and <s1,t1>, Bezout coefficient

pair<mpz_class, pair<mpz_class, mpz_class> >EEA(mpz_class &a,
    mpz_class &b){
    mpz_class r0 = b, r1t, r1 = a, s0 = 0, s1t, s1 = 1, t0 = 1,
        t1t, t1 = 0, q;
    while (r0 != 0){
        q = r1 / r0;
        r1t = r1, r1 = r0, r0 = r1t - q * r0;
        s1t = s1, s1 = s0, s0 = s1t - q * s0;
        t1t = t1, t1 = t0, t0 = t1t - q * t0;
    }
    return make_pair(r1, make_pair(s1,t1));
}
```

3. Finally, we put above two functions together to compare the results.

```
int main(){
    cout << "Given two random 4096 bit number: " << a << " and "<<
        b << ", their gcd given by GMP function is: " << gcd(a,b)
        << ", and the result of extended euclidean algorithm is: "
        << EEA(a,b).first <<"."<< endl;

    if(gcd(a,b) == EEA(a,b).first) cout<< "We can see that two
        results are identical."<<endl;
    else cout<< "Two results are different."<<endl;

    return 0;
}
```