成果：

1.加密

```
E(Encryption) or D(Decryption): E
Key: 4 3 1 2 5 6 7
Plain Text: I sit by window waiting for you

I s i t b y w
i n d o w w a
i t i n g f o
r y o u X X X

ciphertext: idiotonusntyIiirbwgXywfXwaoX
```

2.解密

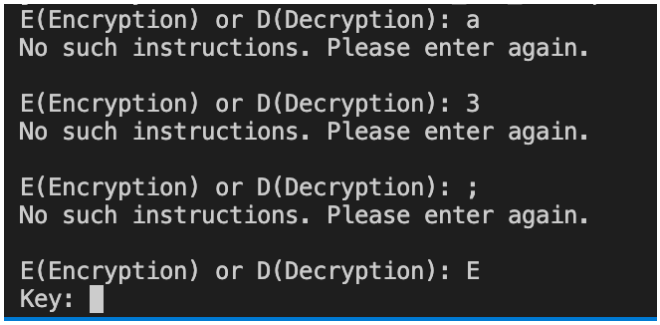
```
E(Encryption) or D(Decryption): D
Key: 3 1 5 6 2 4
Cipher Text: eyyardoystrricgXcappunth

s e c u r i
t y a n d c
r y p t o g
r a p h y X

plaintext: securityandcryptographyX
```

3. 加解密指令防呆：

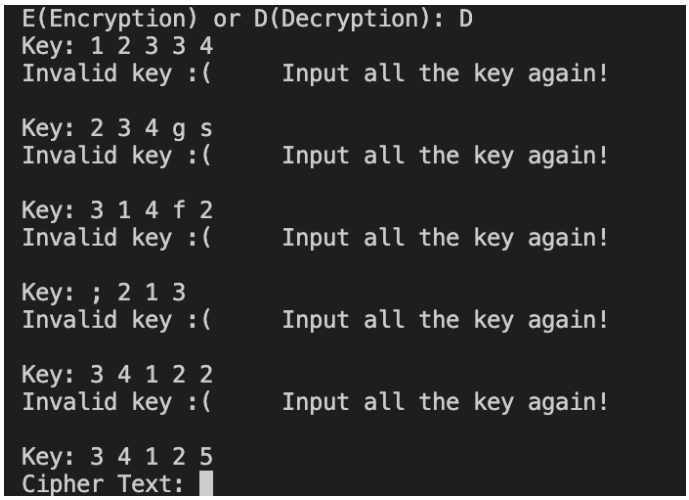
唯有輸入 E 或 D 時 才會進入輸入 key 的步驟



4. key 輸入防呆：

無論要加密/解密時

唯有輸入合法的 key 才能進入輸入明文/密文的步驟



流程：

選擇加密/解密指令⇢輸入合法 key⇢輸入明文/密文⇢產生密文/明文

```
int main()
{
    while (1)
    {
        cout << "E(Encryption) or D(Decryption): ";
        char operation;
        cin >> operation;

        if (operation == 'E')
        {
            encryption();
            break;
        }
        else if (operation == 'D')
        {
            decryption();
            break;
        }
        else
        {
            cout << "No such instructions. Please enter again." << endl
                 << endl;
        }
    }

    return 0;
}
```

加密：輸入密碼、明文⇢放入二維陣列⇢輸出陣列檢查是否放正確⇢產生密文

```
void encryption()
{
    /*------   ↓input the key↓   ------*/
    keylen = InputKey();

    /*------   ↓input the plain text↓   ------*/
    cout << "Plain Text: ";
    getline(cin, plaintext);

    textlen = 1;
    for (int i = 0; i < plaintext.length(); i++)
    {
        if (plaintext[i] != ' ') textlen++;
    }cout << endl;

    /*------   ↓construct text[][]↓  ------*/
    depth = ceil((float)textlen / keylen);
    int index = 0;
    for (int row = 0; row < depth; row++)
    {
        for (int col = 0; col < keylen; col++)
        {
            while (plaintext[index] == ' ')
            {
                index++;
            }

            if (plaintext[index] != '\0')
            {
                text[row][col] = plaintext[index++];
            }
            else
            {
                text[row][col] = 'X';
            }
        }
    }
    /*------   ↓output text[][]↓  ------*/
    outputText();

    /*------  ↓generate cipher text ↓  ------*/
    for (int i = 1; i <= keylen; i++) //find the key from 1
    {
        int col = -1;
        do
        {
            col++;
        } while (key[col] != i);

        if (key[col] == i)
        {
            for (int row = 0; row < depth; row++)
            {
                ciphertext += text[row][col];
            }
        }
    }

    cout << "ciphertext: " << ciphertext << endl<< endl;
}
```

解密：輸入密碼、密文⇢放入二維陣列⇢輸出陣列檢查是否放正確⇢產生明文

```
void decryption()
{

    /*------   ↓input the key↓   ------*/
    keylen = InputKey();

    /*------   ↓input the cipher text↓   ------*/
    cout << "Cipher Text: ";
    getline(cin, ciphertext);

    textlen = 0;
    for (int i = 0; i < ciphertext.length(); i++)
    {
        ++textlen;
    }
    cout << endl;

    /*------   ↓construct text[][]↓  ------*/
    depth = ceil((float)textlen / keylen);
    int index = 0;
    for (int i = 1; i <= keylen; i++) //find the key from 1
    {
        int col = -1;
        do
        {
            col++;
        } while (key[col] != i);

        if (key[col] == i)
        {
            for (int row = 0; row < depth; row++)
            {
                text[row][col] = ciphertext[index++];
            }
        }
    }

    /*------   ↓output text[][]↓  ------*/
    outputText();

    /*------  ↓generate plain text ↓  ------*/
    for (int row = 0; row < depth; row++)
    {
        for (int col = 0; col < keylen; col++)
        {
            plaintext += text[row][col];
        }
    }
    cout << "plaintext: " << plaintext << endl;
}
```