Suppose a 14-character message has been encrypted as follows: Each character is assigned a number, A=1, B=2, C=3, etc. which is converted to a 5-bit binary string, A=00001, B=00010, C=00011, etc.

The binary is then encrypted using a recursively defined key, creating the ciphertext which is sent. You obtain this ciphertext which has been broken into 5-bit chunks for easy reading:

```
00110 01011 10000 01100 01111 10001 00010
11110 01100 00110 11000 10000 01011 10100
```

You also manage to intercept the first five characters of the original plaintext:

```
ONEDA
```

Decrypt the message as follows:

1. Convert the plaintext characters to numbers to bits. You will have 25 bits.
2. Add these bits, bit-by-bit, mod 2 to the first 25 bits of the ciphertext. This will give you 25 bits worth of key fragment.
3. Use refined brute force to find the corresponding \( \vec{c} \). You don’t need to explicitly give each \( M_m \) but you should be clear about the values of all \( det(M_m) \mod 2 \) that you check, why you stop at some particular \( m \), and how you use that \( M_m \).
4. Use \( \vec{c} \) to create enough key to decrypt the entire message.
5. Decrypt the entire ciphertext and convert to numbers to characters.

Matlab note:

If you have a key fragment stored in a vector then we can extract the matrix \( M_m \) easily as well as do determinants and solve equations all \( \mod 2 \). Here’s an example for \( M_2 \) with a made-up key fragment:

```matlab
>> kf = [1;0;1;0;1;0;1;0;1;0;1;0];
>> m = 2;
>> M = [];for i=1:m;M=[M kf(i:i+m-1)];end;
>> M
M =
     1     0
     0     1
>> mod(det(M),2)
ans =
     1
>> mod(det(M)*inv(M)*kf(m+1:m+m),2)
ans =
     1
     0
```