The discrete exponentiation problem is to determine \( t = g^e \mod n \), that is, the remainder when \( g^e \) is divided by \( n \). For example, \( 2^4 \mod 11 \) is 5 since \( 2^4 = 16 \), which is 5 more than 11.

One (very slow) way to calculate the discrete exponentiation is to simply multiply \( g \) by itself the requisite number of times, at each stage taking the remainder when divided by \( n \). (It can be shown that this is mathematically valid.)

The inverse problem is called the discrete logarithm problem. Here you are given the answer \( t \) and \( g, n \), and must determine \( e \). One way to calculate the discrete logarithm is the same idea: multiply \( g \) by itself taking the remainder at each stage, until the target \( t \) is found. Now, the discrete logarithm does not always exist: so if in the process you reach a product of 1, then one should abort. (It can be shown that this is mathematically valid.)

Create a MATLAB program called \texttt{expoLog.m} that

(a) asks the user whether they want a exponentiation or a logarithm,
(b) prompts for the three parameters, and
(c) then does the calculation, printing out the intermediate values.

Sample runs:

```matlab
>> expoLog
Expo=1 Log=2 1
g is 2
e is 4
n is 11
2,4,8,5,
answer is 5
```

```matlab
>> expoLog
Expo=1 Log=2 2
g is 2
t is 5
n is 11
2,4,8,5,
Log is 4
```

```matlab
>> expoLog
Expo=1 Log=2 2
g is 3
t is 7
n is 11
3,9,5,4,1,
No Log
```