READ THIS FIRST:

Do your best to do every item on your own; if you cannot immediately do an item, go on to others and then come back to it later. Please check the resources section if you have any problems and talk with your professor if there are any further questions.

Due: The following Sunday after this was posted by midnight.

Goals:

- Practice getting around the command line compiling and running Java programs.
- Practice getting around in and using the lab submission site.
- Explain some key concepts we covered in class.
- Get some lab points.

Part One. **Description**:

The famous Hill cryptosystem was an early susbtitution cipher based on linear algebra and modular arithmetic. One of the key components is an encryption matrix that has an important property: it has to be invertible in modulo m. An important step in determining if an $n \times n$ matrix A is invertible is to find if the determinant of A and m are relatively prime.

A common way of finding the determinant of A, `det(A)`, is to use an algorithm of cofactor expansion along the first row. If we denote A in terms of its elements in the i-th row and j-th column as $A = [a_{ij}]$, then we can formally describe the algorithm for the three major cases as follows:

1. For $n = 1$, `det(A)` is simply a, since $A = [a]$.
2. For $n = 2$, `det(A)`$= a_{11}a_{22} - a_{12}a_{21}$, where $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$.
3. For $n \geq 3$, `det(A)`$= a_{11}$`det(`A_{11}`)`$- a_{12}$`det(`A_{12}`)`$+ a_{13}$`det(`A_{13}`)`$- \cdots + (-1)^{n+1}a_{1n}$`det(`$A_{1n}$`)`, where A_{ij} is the $(n-1) \times (n-1)$ submatrix that results when the i-th row and the j-th column are removed from the original matrix A.

Your mission is to write a program that calculates the determinant of any square modular matrix. In this case, we will limit our input to signed 32-bit integers.

Input:

The input starts with a positive integer m, the modulo under which all integer calculations will be performed, followed by a single space and n, the size of the matrix A. Then it is followed by exactly n lines with n integers separated by a single space on each line.

Output:

The output must be the smallest integer greater than or equal to zero corresponding to the `det`(A) in modulo m.

Sample Input 1:

```
12  2
3  2
2  3
```

Sample Output 1:

```
5
```

Sample Input 2:

```
3  2
1  2
9  8
```

Sample Output 2:

```
2
```

Sample Input 3:

```
9  3
7  5  2
0  6  4
8  2  5
```

Sample Output 3:

```
2
```

Requirements: You will create a file `Driver_lab3a.java` that will contain a class with the name `Driver_lab3a` which will contain the `main` method of the program. Inside your driver create reate a method with the following signature `int cofModDet(int m, int[][] A)` that receives an integer $m > 0$, a two-dimensional array (matrix) A, and returns the corresponding determinant in modulo m as an integer. Note that all integer operations must be performed in modulo m as well as the result. Also, make sure you follow the style guidelines that were given for this course.

Part Two. **Description**:

Sometimes you want to process information down to its bits, for example, in cryptography, you often want to process a plaintext P for analysis or encryption. Your mission this time is to process a plaintext formatting it down to its bytes.

If we define P as the plaintext composed of several bytes b, then we could say that $P = b_1 b_2 \dots b_n$ denotes a plaintext of n bytes. In this problem we will use the ASCII code to represent the plaintex for simplicity. You see, every symbol in the ASCII code can be mapped to a hexadecimal value of two digits, which is a byte.

Your mission, is to take a plaintext P composed of n symbols from the ASCII table, and express that plaintext in a matrix of four rows and a variable number of columns, whose elements are hexadecimal bytes placed in column-major order, as follows:

$$P = \begin{bmatrix} b_1 & b_5 & \vdots \\ b_2 & b_6 & \vdots \\ b_3 & b_7 & b_n \\ b_4 & \vdots & \end{bmatrix}$$

However, there is one caveat: all elements in the matrix must have a value and the number of columns in the plaintext matrix P has to be a multiple of four. Thus, for the cases in which the number of columns is less than a multiple of four or there are empty values, we will use the technique known as padding, meaning that we will fill those elements with a chosen value s. We will call this new matrix $\hat{P}_s$.

For example, if the plaintext is $P =$ 'Hola mundo!', the corresponding $\hat{P}_s$ matrix with a chosen value of $s =$ ' $\sim$ ' would be the following:

$$\hat{P}_s = \begin{bmatrix} 48 & 20 & 64 & 7E \\ 6F & 6D & 6F & 7E \\ 6C & 75 & 21 & 7E \\ 61 & 6E & 7E & 7E \end{bmatrix}$$

Your task is to provide the padded matrix $\hat{P}_s$ for any given plaintext P and substitution character s.

Input:

The input consists of two lines. The first contains the substitution character s, where $|s| = 1$. The second line contains the plaintext P, where $|P| \leq 32,768$.

Output:

The output is composed of at least four lines with the possibility of more. The output is organized as follows: Let $\hat{P}_s(1)$, denote the first group of four columns of the matrix $\hat{P}_s$; the program will print $\hat{P}_s(1)$ followed by an empty line, followed by $\hat{P}_s(2)$ which is the second group of four columns followed by an empty line, and so on until the entire matrix $\hat{P}_s$ has been printed in groups of four columns.

Sample Input 1:

```
~
Hola mundo!
```

Sample Output 1:

```
48 20 64 7E
6F 6D 67 7E
6C 75 21 7E
61 6E 7E 7E
```

Sample Input 2:

```
0
Een goede naam is beter dan olie.
```

Sample Output 2:

```
45 67 65 61
65 6F 20 6D
6E 65 6E 20
20 64 61 69

73 74 64 6F
20 65 61 6C
62 72 6E 69
65 20 20 65

2E 30 30 30
30 30 30 30
30 30 30 30
30 30 30 30
```

Requirements: You will create a file `Driver_lab3b.java` that will contain a class with the name `Driver_lab3b` which will contain the `main` method of the program. Inside your driver create a method with the following signature `int[][] getHexMatP(char s, String p)` that receives a susbstitution character s, and a string p whose length is $|p| \leq 16$; the length of p should always be 16 unless the plaintext P is smaller than 16. What I would like you to do is store the full P somewhere and invoke `getHexMatP()` repeatedly in some kind of loop using substrings of P of lenght 16 (i.e., p) and print the matrix returned by the method every time. Also, make sure you follow the style guidelines that were given for this course.

Resources:

- Your textbook (Stanoyevitch)!
- Project submission guidelines for this course (posted on iLearn)
- Coding style guidelines for this course (posted on iLearn)
- "How to" use the command line "shell" (posted on iLearn)
- Piazza for asking questions to professor and classmates use the tag: `lab3`
- The official Java reference: `http://docs.oracle.com/javase/tutorial/collections/TOC.html`
- Stack Overflow Java Tag: `http://stackoverflow.com/questions/tagged/java`

Submission:

- Upload your work to the submission site `https://car.rivas.ai` and submit your `Driver_lab3a.java` and `Driver_lab3b.java` before the due date and make sure they pass all the tests. If they do not pass all the tests, then it means that your programs are incorrect and you need to keep working on them.
- Once you pass all the tests, your professor will review your code for style and then you will receive a grade.