Function Name: varInfo

Inputs:

1. *(double)* A number or vector of any length
   OR
   (char) A string of any length

Outputs:

1. *(char)* An output string describing the input variable

Function Description:

Write a MATLAB function to identify the class of the input variable and to display its value. Your function should output a string of the form:

```
This variable is of class <type> and has a value of '<value>'.
```

For example, if the input variable is the number 25, then your function should output the string:

```
This variable is of class double and has a value of '25'.
```

If instead the input variable is the string `'CS 1371'`, then your function should output the string:

```
This variable is of class char and has a value of 'CS 1371'.
```

Notes:

- Make sure that your output variable matches the answers to the solution file EXACTLY. Any extra spaces or characters will result in a 0 for this problem.
- The period at the end of the output strings shown above is part of the string.

Hints:

- The `class()`, `num2str()`, and `sprintf()` functions will be useful.
- Using `num2str()` on a string will output the same string. Using `num2str()` on a vector will print the vector to a string with spaces between the elements.
- To get an apostrophe into a string, type it into MATLAB as two apostrophes.

Function Name: `word2int`

Inputs:

1. *(char)* A single word

Outputs:

1. *(double)* A unique number corresponding to the input word

Function Description:

The problem of securely encrypting words is a huge field of research in the computer world. There are many algorithms for encrypting words, but this function will implement a simple method for turning a word into an integer that is sufficiently difficult to decode. Here's how it works:

Every character in the input word is converted into a number and then all of the numbers are summed to generate a single number corresponding to the input word. Each character's number will be computed as follows:

$$num_n = 131^{n-1} * ASCII(char_n)$$

n is the index of a particular character

$ASCII(char_n)$ is the ASCII value of the nth character in the string

So if the input string was 'ABC', the output would be:

$$131^0 * 65 + 131^1 * 66 + 131^2 * 67 = 1158498$$

(65, 66, and 67 are the ASCII values of A, B, and C, respectively)

There is a lot of math that can show that using this method will produce a unique integer for any unique string, but it is very difficult to convert the integer back to the string (you are welcome to try to write such a function). You can imagine it would be even more difficult to decode if you didn't know which prime had been used in the encoding!

Notes:

- ASCII(char) is not a MATLAB function; that was just to show that you should use the ASCII value.

Hints:

- `5.^[1, 2, 3]` will output the vector `[5, 25, 125]`

Function Name: `shortCat`

Inputs:

1. *(char)* A string
2. *(char)* Another string

Outputs:

1. *(char)* A string containing a concatenation of the input strings

Function Description:

This function will concatenate the two input strings, but with a small catch. If one of the strings is longer than the other, you should only concatenate the last N characters of the longer string, where N is the length of the shorter string. For example, if the input strings were `'Hello'` and `'MATLAB'` the output would be `'HelloATLAB'`. `'Hello'` is the shorter of the two words at 5 characters, so the last 5 characters of the other string are concatenated. The first input will always be concatenated in front of the second input.

Notes:

- One or more of the inputs could be the empty string `''`, in which case the output should also be the empty string.

Function Name: `weave`

Inputs:

1. *(double)* 1xM vector to be inserted in the odd position indices
2. *(double)* 1xN vector to be inserted in the even position indices

Outputs:

1. *(double)* 1xQ vector with input vectors weaved

Function Description:

Given two vectors, output a larger vector where the values in the first vector are inserted in the odd indices and the values in the second vector are inserted in the even indices. For example, `weave([1 3 7 1], [2 0 1 6])` should output

[1 2 3 0 7 1 1 6]

If the two vectors are not the same length, the shorter vector should be extended by adding consecutive integers to the end. For example, if the first input vector is [2 7 1] and the second input vector is [3 1 4 1 5 9 2], the first input vector should be extended to [2 7 1 2 3 4 5], and then weaved with the second input vector.

Notes:

- The two vectors are guaranteed to be non-empty.
- Turn all the test cases into one string to reveal a secret message!

Function Name: `checkContour`

Inputs:

1. *(double)* A vector of length N
2. *(double)* Another vector of length N

Outputs:

1. *(logical)* Whether or not the "contour" of the two vectors is the same

Function Description:

For the sake of this problem, we will define the *contour* of a vector to be whether adjacent elements of the vector are increasing or decreasing. Therefore, any vector defines a sequence of up and down contours based on the difference between adjacent values. The contour of two vectors is the same if they both have the same pattern of increasing and decreasing elements. For the contour to be the same, the vectors DO NOT have to have the same values, nor does the amount by which elements increase or decrease have to be the same; only the pattern of increasing and decreasing matters.

Consider the following vectors

```
v1 = [2, 4, 3]
v2 = [-2, 10, 8]
```

These two vectors have the same contour because they both follow the pattern [increasing, decreasing]. Now consider these two vectors:

```
v3 = [5, 3, 1]
v4 = [1, 4, 1]
```

These vectors do not have the same contour. The first one follows the pattern [decreasing, decreasing] while the second one follows the pattern [increasing, decreasing]

The function should output true if and only if the two vectors have the same "contour".

Notes:

- There will never be two equivalent adjacent elements in any of the vectors.
- The `diff()` and `abs()` functions may be helpful.
- Both vectors will always be the same length.
- A single number has the same contour as any other single number.
- Make sure that your output is class logical and not double.

Hints:

- Try to create a new vector that represents the contour of an input vector.