MATLAB Programming NCTU Spring 2017 LAB#12 2017/5/26

For this lab, you will implement a class whose objects are very large positive integers. An example:

```
12345678909876543210
```

The following are the descriptions / tasks:

- Define a class **`BigInt`**. This class should have a single property named **`digits`**, which is a vector where every element is a digit of the big integer. You can set its default value to zero (which will be the only digit of this big integer).
- The constructor method should handle the following cases:
  - No input. In this case, make **`digits`** a scalar zero.
  - A single input that is a string consisting only of digits.
  - A single input that is a scalar non-negative integer. (An easy way to handle this is to convert it to a string first using **`num2str`**.)
  - All other cases should result in an error message.
- A method **`make_str`**, which generates a string representation of the big integer as its output.
- The overloaded method **`disp`**, which displays the big integer as if it's a regular integer. This method can call **`make_str`** to generate a string, which can then be displayed using the regular **`disp`** function.
- The overloaded operator **`plus`** for adding two big integers, or one big integer and one regular integer.
- The overloaded operator **`times`** for multiplying two big integers, or one big integer and one regular integer. We will talk in the class about how to do this easily using the function **`conv`**.
- The overloaded operator **`eq`** for checking whether two big numbers are equal.

Try to test your class with the following code:

```
a = BigInt('5678');  b = BigInt(99);
c = a + a;  disp(c);
d = a + b;  disp(d);
e = a .* b;  disp(e);
f = 2 .* e;  disp(f);
disp(e + e == f);
```

※**Important**: Be sure to use **`clear all`** every time you change your class code, so that you won't have objects in your workspace that are generated using both the old and new classes.

Everything you do in this lab only need to be able to handle operations on individual objects. Extensions to arrays of big integers will be handled in the next lab. Remember to save you code for today somewhere so that you can use it later.