# MATH-131 (Numerical Methods for Scientists and Engineers) — Worksheet 2

Semester: Spring 2019, Instructor: Nicholas Knight

***Due Feb. 5 at 2359.*** *Please remember to cite your sources, including your collaborators.*

*Deliverable:* Submit a Live script titled `worksheet2.mlx` via CatCourses (under Assignments). Divide this file into sections, one for each of the following questions, plus an extra (final) section containing all the function definitions. Document each function definition to explain the input and output arguments.

1. In lecture we learned that the number $1/10 = 0.1$ cannot be represented exactly in binary floating point (just like $1/3 = 0.333\cdots$ cannot be represented exactly in decimal floating point). However, this fact is hidden by the way that Matlab outputs numbers. To see this, try the commands:

   ```
   format short e;
   x = .1
   format long e;
   x = .1
   ```

   You can display the nearest $(d+1)$-digit decimal floating point number to a binary floating point number $x$ using the command `fprintf('%.*e\n', d, x);`

   How big do you need to make $d$ so that this command shows that $x = 0.1$ is not represented exactly? (Experiment with different values of $d$ and find the smallest.)

2. Write a function that converts a natural number (nonnegative integer) into an array of its decimal digits. It should have the signature

   ```
   function X = uint_to_digits(x)
   ```

   where `x` is a natural number and `X` is an array of natural numbers between 0 and 9. For example,

   ```
   uint_to_digits(12345)
   ```

   outputs

   ```
   ans =
        1 2 3 4 5
   ```

   (*Hint:* The commands `log10` and `floor` (or `ceil`) are useful.)

   Show the results for

   ```
   uint_to_digits(2^53)
   uint_to_digits(2^53 + 1)
   uint_to_digits(2^53 + 2)
   ```

   Explain why two of the three outputs are equal.

3. Write a function that converts an array of decimal digits into a natural number. It should have the signature

   ```
   function x = digits_to_uint(X)
   ```

   where `x` is a natural number and `X` is an array of natural numbers between 0 and 9. For example,

   ```
   digits_to_uint([1 2 3 4 5])
   ```

   outputs

   ```
   ans =
        12345
   ```

   Show the results for

   ```
   format long e;
   digits_to_uint([9 0 0 7 1 9 9 2 5 4 7 4 0 9 9 2])
   digits_to_uint([9 0 0 7 1 9 9 2 5 4 7 4 0 9 9 3])
   digits_to_uint([9 0 0 7 1 9 9 2 5 4 7 4 0 9 9 4])
   ```

   Explain why two of the three outputs are equal.

4. Write a function that implements addition of natural numbers. The function signature should be

   ```
   function s = addn(a, b)
   ```

   where `a`, `b`, and `s` are digit arrays (like those returned by `uint_to_digits`). For example,

   ```
   addn([2 3 4 6], [9 9 9 9])
   ```

   outputs

   ```
   ans =
        1 2 3 4 5
   ```

   Your function should only use the addition table for integers 0 through 9. You *cannot* simply do:

   ```
   s = uint_to_digits( digits_to_uint(a) + digits_to_uint(b) );
   ```

5. Write a function that implements multiplication of natural numbers. The function signature should be

   ```
   function s = multn(a, b)
   ```

   where `a`, `b`, and `s` are digit arrays (like those returned by `uint_to_digits`). For example,

   ```
   multn([1 5], [8 2 3])
   ```

   outputs

   ```
   ans =
        1 2 3 4 5
   ```

   Your function should only use the multiplication table for integers 0 through 9. You *cannot* simply do:

   ```
   s = uint_to_digits( digits_to_uint(a) * digits_to_uint(b) );
   ```