# MATH 210 Assignment 3

*More Logic, Loops and Functions*

## INSTRUCTIONS

- Create a new Python 3 Jupyter notebook
- Answer each question in the Jupyter notebook and clearly label the solutions with headings
- Functions should include documentation strings and comments
- There are 15 total points and each question is worth 3 points
- Submit the `.ipynb` file to Connect by **11pm Monday, January 30, 2017**
- You may work on these problems with others but you must write your solutions on your own
- Do **not** import any Python packages such as `math` or `numpy` to complete this assignment. These questions require only the standard Python library. Solutions will be given 0 if any Python package/module is used.

## QUESTIONS

1. Write a function called `prime_divisors` which takes one input parameter $N$ (a positive integer) and returns a Python list of prime numbers which divide $N$. For example:

    `prime_divisors(21)` returns `[3,7]`

    `prime_divisors(24)` returns `[2,3]`

    `prime_divisors(1815)` returns `[3,5,11]`

2. Write a function called `prime_factorization` which takes one input parameter $N$ (a positive integer) and returns a Python list of tuples $[(p_1, n_1), \dots, (p_m, n_m)]$ which gives the factorization of $N$ into primes:

    $$N = p_1^{n_1} p_2^{n_2} \cdots p_m^{n_m}$$

    For example:

    `prime_factorization(21)` returns `[(3,1),(7,1)]` since $21 = 3^1 \cdot 7^1$

    `prime_factorization(24)` returns `[(2,3),(3,1)]` since $24 = 2^3 \cdot 3^1$

    `prime_factorization(1815)` returns `[(3,1),(5,1),(11,2)]` since $1815 = 3^1 \cdot 5^1 \cdot 11^2$

3. Given a finite sequence of positive integers $[a_0, a_1, \dots, a_n]$ (of length $n+1$), define a new finite sequence $[b_0, b_1, \dots, b_n]$ (defined recursively) by

$$
\begin{aligned}
b_0 &= a_0 \\
b_1 &= a_1 + \frac{1}{b_0} = a_1 + \frac{1}{a_0} \\
b_2 &= a_2 + \frac{1}{b_1} = a_2 + \frac{1}{a_1 + \frac{1}{a_0}} \\
b_3 &= a_3 + \frac{1}{b_2} = a_3 + \frac{1}{a_2 + \frac{1}{a_1 + \frac{1}{a_0}}} \\
&\vdots \\
b_n &= a_n + \frac{1}{b_{n-1}} = a_n + \frac{1}{\ddots + \frac{1}{a_0}}
\end{aligned}
$$

Write a function called `sequence_to_fraction` which takes one input parameter `integer_list` (a Python list of positive integers $[a_0, a_1, \dots, a_n]$) and returns the last number $b_n$ in the sequence defined above

$$
b_n = a_n + \frac{1}{a_{n-1} + \frac{1}{\ddots + \frac{1}{a_0}}}
$$

For example:

`sequence_to_fraction([1,1])` returns `2.0`

`sequence_to_fraction([1,1,1,1,1,1,1,1,1,1,1])` returns `1.6179775280898876`

`sequence_to_fraction([6,1,1,4,1,1,2,1,2])` returns `2.718279569892473`

`sequence_to_fraction([2,1,1,1,292,1,15,7,3])` returns `3.141592653581078`

4. Define a function called `product` which takes a Python list of numbers and returns the product of the numbers in the list. For example:

`product([1,2,3,4])` returns `24`

`product([2,3,5,7,11,13])` returns `30030`

`product([0.5,0.25,0.125])` returns `0.015625`

5. Write a function called `sequence_to_roots` which takes one input parameter `integer_list` (a Python list of positive integers $[a_0, a_1, \dots, a_n]$) and returns the number

$$
\sqrt{a_n + \sqrt{a_{n-1} + \sqrt{\cdots + \sqrt{a_0}}}}
$$

For example:

`sequence_to_roots([1,1])` returns `1.4142135623730951` (ie. $\sqrt{1 + \sqrt{1}}$)

`sequence_to_roots([2,2,2,2,2])` returns `1.9975909124103448` (ie. $\sqrt{2 + \sqrt{2 + \sqrt{2 + \sqrt{2}}}}$)

`sequence_to_roots([1,2,3])` returns `2.1753277471610746` (ie. $\sqrt{3 + \sqrt{2 + \sqrt{1}}}$)