```
>>> from math import sqrt
>>>
>>> # Jolten Larremore
>>> # Instructor: Dr. Igor Steinmacher
>>> # Course: INF502
>>> # Date: August 29, 2020
>>>
>>> # Problem 1 - Pythagorean Theorem
>>>
>>>
>>> def pythagoreanTheorem(length_a, length_b):
        a = float(length_a)
        b = float(length_b)
        c = sqrt((a**2)+(b**2))
        return c

>>>
>>>
>>> # Example Runs of Problem 1
>>>
>>> pythagoreanTheorem(2, 2)
2.8284271247461903
>>>
>>> pythagoreanTheorem(4, 6)
7.211102550927978
>>>
>>> pythagoreanTheorem(8, 5)
9.433981132056603
>>>
>>> # Problem 2 - List Mangler
>>> # For this problem, I define a new function called list_mangler
that takes in one variable (i.e. the list of integers).
>>> # Within the function, a new variable called 'new_lst' is created
to store the output. Then, I create a 'for' loop that
>>> # reads each number in the list input. Each number will go through
an if-else conditional statement, where a number is
>>> # multiply by 2 if its remainder, when the number is divided by 2,
equals to 0. This is achieved if the number is even.
>>> # Else, the number is indicated to be an odd number, and is
multiply by 3. For each product, it is added to the
>>> # 'new_lst' using the .append() function. The last step is to
return the 'new_lst'.
>>>
>>> def list_mangler(list_in):
    new_lst = []
    for number in list_in:
        if number % 2 == 0:
            number = number * 2
            new_lst.append(number)
        else:
```

```
            number = number * 3
            new_lst.append(number)
    return new_lst

>>>
>>>
>>> # Example Runs of Problem 2
>>>
>>> list_mangler([1, 2, 3, 4])
[3, 4, 9, 8]
>>>
>>> list_mangler([5, 6, 7, 8])
[15, 12, 21, 16]
>>>
>>> list_mangler([9, 10, 11, 12])
[27, 20, 33, 24]
>>>
>>> # Problem 3 - Grade Calculator
>>> # For this problem, I created a function called grade_calc
(abbreviation for grade calculator) that accepts a list
>>> # "grades_in", which contains the integer grades, as well as
another variable in a form of an integer called "to_drop".
>>> # The first steps within the function is to sort the grades_in
list in order from least to greatest using .sort(), and
>>> # create a new variable called deletions, which is set to 0. Then,
a while loop is created, in which a minimum value of
>>> # the grades_in list is stored in the variable "low_v" for "low
variable". After that, the "low_v" is removed from the
>>> # entire grades_in list using .remove(), and the value of the
deletions is added by 1. The "while" loop keeps going
>>> # until the value of "deletions" is greater than the "to_drop"
value; thereby breaking the loop. Then, the average is
>>> # calculated by taking the sum of the remaining grades left and
dividing it by the length of the "grades_it" list. Next,
>>> # I create a dictionary titled "grade_lookup" where I associate
each letter grade (the value) to a logical comparison
>>> # operation involving the calculated average (the keys). Lastly, I
created a for loop that reads each operation (titled
>>> # "equ") in the grade_lookup dictionary. Within the loop, I
established an if-else conditional statement that returns
>>> # the letter grade if "equ" is true; otherwise, the loop
continues.
>>>
>>>
>>> def grade_calc(grades_in, to_drop):
    grades_in.sort()
    deletions = 0
    while deletions <= to_drop:
        low_v = min(grades_in)
        grades_in.remove(low_v)
```

```
        deletions += 1
    avg = sum(grades_in) / len(grades_in)
    grade_lookup = {
        (0 <= avg <= 59): 'F',
        (60 <= avg <= 69): 'D',
        (70 <= avg <= 79): 'C',
        (80 <= avg <= 89): 'B',
        (90 <= avg <= 100): 'A'
    }
    for equ in grade_lookup:
        if equ is True:
            return grade_lookup[equ]
        elif equ is False:
            continue


>>> grade_calc([100, 90, 80, 95], 2)
'A'
>>>
>>> grade_calc([60, 80, 74, 78], 1)
'C'
>>>
>>> grade_calc([89, 85, 90, 100], 0)
'A'
>>>
>>>
>>> # Problem 4 - Odd & Even Filter
>>> # For this problem, I define a new function called odd_even_filter
that takes in one
>>> # variable (i.e. an input list of integers). The variable is
called "numbers". Within
>>> # the function, "numbers" is stored in a new variable called
"lst". Then, two new
>>> # variables called "out_even" and "out_odd" are created, each
assigning to an empty >>> # list.These will store the outputs. Next, I
created a "for" loop that goes over each
>>> # number (num) in "lst". Inside the loop, I placed an
>>> # if-else conditional statement that states that if the remainder
of the num equals 0,
>>> # then that num is added in the out-even list using .append(). If
the remainder does >>> # not equal zero, then the num is added in the
out_odd list using .append(). The last
>>> # step for the function to perform is to return a list with two
sublists. The first >>> # sublist contains contains all even numbers
in the input list (out_even), and the
>>> # second sublist contains all odd numbers (out_odd).
>>>
>>>
>>> def odd_even_filter(numbers):
        lst = numbers
```

```
        out_even = []
        out_odd = []
        for num in lst:
            if num % 2 == 0:
                out_even.append(num)
            else:
                out_odd.append(num)
        return [out_even, out_odd]

>>>
>>> def odd_even_filter(numbers):
    lst = numbers
    out_even = []
    out_odd = []
    for num in lst:
        if num % 2 == 0:
            out_even.append(num)
        else:
            out_odd.append(num)
    return [out_even, out_odd]

>>>
>>>
>>> # Example Runs of Problem 4
>>>
>>> odd_even_filter([1, 2, 3, 4, 5, 6, 7, 8, 9])
[[2, 4, 6, 8], [1, 3, 5, 7, 9]]
>>>
>>> odd_even_filter([2, 4, 42, 8])
[[2, 4, 42, 8], []]
>>>
>>> odd_even_filter([37, 20, 91, 54, 4, 29, 100, 30, 1, 78])
[[20, 54, 4, 100, 30, 78], [37, 91, 29, 1]]
>>>
```