CSC108H Worksheet: unittest

1. Recall our function `collect_underperformers`:

```
def collect_underperformers(nums, threshold):
    """ (list of number, int) -> list of number

    Return a new list consisting of those numbers in nums that are below threshold,
    in the same order as in nums.
    """
```

   (a) We've begun writing a test suite for this function using `unittest`. Complete methods `test_underperformers_high_threshold` and `test_underperformers_mutation`.

```
class TestCollectUnderperformers(unittest.TestCase):

    def test_underperformers_low_threshold(self):
        """ Test collect_underperformers with a threshold for which there
        are no underperformers."""

        actual = collect_underperformers([4, 5, 6], 1)
        expected = []
        self.assertEqual(actual, expected)


    def test_underperformers_high_threshold(self):
        """ Test collect_underperformers with a threshold for which all items
        are underperformers."""









    def test_underperformers_mutation(self):
        """ Confirm that collect_underperformers does not mutate the list it's given."""
```

Note: the test suite above is not complete!

(b) We've changed our mind about the desired behaviour for `collect_underperformers`. We'd like it to modify the list it is given, and not return anything.

```
def collect_underperformers(nums, threshold):
    """ (list of number, int) -> NoneType

    Remove the numbers in nums that are below threshold.
    """
```

Rewrite the first testing method above to work with `collect_underperformers`'s new description.

```
class TestCollectUnderperformers(unittest.TestCase):

    def test_underperformers_low_threshold(self):
        """ Test collect_underperformers with a threshold for which there
        are no underperformers."""
```

2. Complete the two test methods for `most_popular`, described below.

```
def most_popular(company_to_placements):
    """ (dict of {str : list of int}) -> list of str

    Precondition: company_to_placements is not empty

    Return the company (or companies) with the most placements in the race.
    """

class TestMostPopular(unittest.TestCase):

    def test_most_popular_one_item(self):
        """ Test most_popular with a dictionary of length 1."""

    def test_most_popular_mutation(self):
        """ Confirm that most_popular does not mutate the dict it is given."""
```