CSE 160 Section 9
Final Exam Practice Problem Solutions

Part 1: Reading Code

1. This code will cause an error.

   The function `list.remove(x)` removes the first occurrence of x in the list, however it returns `None`. When the iterating variable of the outer loop takes on the value 2, `item` will be in `items_to_remove` so the body of the if-statement will be executed. After the line `input_list = input_list.remove(item)` the value of `input_list` will then be `None`. Later, when the iterating variable takes on the value 4, `item` will again be in `items_to_remove`. However this time, when Python tries to execute `input_list.remove(item)` the value of `input_list` is `None` and it will throw the error:

   `AttributeError: 'NoneType' object has no attribute 'remove'.`

2. 2

3. This code will cause an error. This function iterates through it's string parameter with a for-loop. As it is written, the iterating variable will take on values character by character instead of word by word. So, when the word "dime" is searched for as a key in the dictionary, it doesn't appear in the dictionary. One way to iterate through the first string word by word would be to change the code to: `for w in words.split():`.

Part 2: Writing Code

4.
```
vowels = {"A", "a", "E", "e", "I", "i", "O", "o", "U", "u"}

def same_number_vowels(string1, string2):
    """
    Returns True if string1 and string2 contain the same number of vowels,
    False otherwise.
    """
    vowels_first_string = 0
    for character in string1:
        if character in vowels:
            vowels_first_string += 1

    vowels_second_string = 0
    for letter in string2:
        if letter in vowels:
            vowels_second_string += 1

    return vowels_first_string == vowels_second_string
```

5.
```
def similar_pairs(list1, list2):
    """
    Given two lists of strings, returns a list of all the similar tuples.
    Each tuple contains one string from list1 and one string from list2.
    The strings are only considered similar if they contain the same number of vowels.
    """
    output = []
    for item1 in list1:
        for item2 in list2:
            if same_number_vowels(item1, item2):
                output.append((item1, item2))
    return output
```

Part 3: Design

6.
```
def read_csv(path):
    """
    Reads the CSV file at the given path and returns a list of dictionaries where the keys
    are: name, type, latitude, longitude
    """

def find_nearby_establishments(known_establishments, current_latitude, current_longitude):
    """
    Given: a list of dictionaries where the keys are name, type, latitude and longitude of
    a particular restaurant or bar; and your current latitude and longitude as floats;
    returns a list of names of the restaurants less than 0.007 degrees latitude/longitude
    from your current location.
    """

def find_bar_near_most_bars(known_establishments):
    """
    Given: a list of dictionaries where the keys are name, type, latitude and longitude of
    a particular restaurant or bar; returns the name of the bar that has the most other
    bars within 0.007 degrees latitude/longitude of its location.
    """
```

7.
- Allows for reuse of the `find_nearby_establishments` function.
- `find_nearby_establishments` doesn't give you any more information about the restaurants or bars that are close to you, aside from their names. The dictionary returned by `read_csv` doesn't distinguish between bars and restaurants, so if you wanted information about one in particular you would have to look through the entire dictionary.

Part 4: Understanding Code

8. For each of the statements below, circle "No error" or "Error", depending on whether execution of it causes an error. Answer assuming that you attempt to execute each statement, even if a previous one caused an error.

```
d = {}          #   "No error"

d[w] = "test"   #   "No error"

d[x] = "test"   #   "No error"

d[y] = "test"   #   "Error"

d[z] = "test"   #   "Error"
```

9. List and sets are mutable. Keys of dictionaries must be immutable values.

10.
```
Global                  funny               silly
funny -> function       x   -> 5            y -> 6
silly -> function       val -> 6            x -> 3
goofy -> function
x     -> 5
```

11. 17