Question 1
This code will cause an error.
The function list.remove(x) removes the first value of x in the list, however it returns None. When the loop outer loop gets to the value of item = 2, item will be in items_to_remove, so after the line "somelist = somelist.remove(item)" somelist will then be None. Later, when item = 4, item will again be in items_to_remove. However, this time Python tries to do somelist.remove(item) somelist is None and it will therefore throw the error: "AttributeError: 'NoneType' object has no attribute 'remove'"

Question 2
2

Question 3
This code will cause an error.
The function histogram is given two strings. It then iterates through the first string with a for loop. When it does this, it goes through the loop letter by letter, not word by word. So, when the word "dime" is search for as a key in the dictionary, it doesn’t appear in the dictionary. Note: One way to go through the first given string word by word would be to slightly change the code to: "for w in words.split():"

Question 4
```python
def similar_pairs(list1, list2, similar):
    output = []
    for items in list1:
        for values in list2:
            if similar(items, values):
                output.append((items, values))
    return output
```

Question 5
```python
def similar_number_vowels(string1, string2):
    vowels = ["A", "a", "E", "e", "I", "i", "O", "o", "U", "u"]
    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
```

print similar_pairs(states, capitals, similar_number_vowels)

Question 6
```python
## Iterative version

def contains(list_of_items, item_to_find):
    """Return True if item_to_find is in list_of_items. Otherwise, return False.""
    for element in list_of_items:
        if element in item_to_find:
            return True
    return False
```

return False
Recursive version

def contains(list_of_items, item_to_find):
    '''Return True if item_to_find is in list_of_items. Otherwise, return False.'''
    if list_of_items == []:
        return False
    elif list_of_items[0] == item_to_find:
        return True
    return contains(list_of_items[1:len(list_of_items)], item_to_find)

Question 7

a)

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, a float value of your current
    latitude and longitude returns a list of name of the restaurants less than
    0.007 degrees latitude/longitude of your current location.
    '''

def find_population_location_of_bar(known_establishments):
    '''
    Given a list of dictionaries where the keys are name, type, latitude and
    longitude of a particular restaurant or bar, examines the altitude and
    longitude of each bar to find a bar less than 0.007 degrees latitude/longitude
    of its location.
    '''

b) Allows for reuse of the find_nearby_establishments function.

c) find_nearby_establishments doesn’t give you any more information about the
   restaurants/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.

Question 8

a)

d = {}           # "No error"
d[w] = "test"    # "No error"
d[x] = "test"    # "No error"
d[y] = "test"    # "Error"
d[z] = "test"    # "Error"

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

Question 9
Global
gcd -> function

gcd
a  -> 15
b  -> 10

gcd
a  -> 5
b  -> 10

gcd
a  -> 10
b  -> 5

gcd
a  -> 5
b  -> 5

gcd
a  -> 0
b  -> 5

gcd
a  -> 5
b  -> 0