(a) def repeat_word(word, RepeatNum):
    """ (string, integer) -> string

    Precondition: RepeatNum >= 0

    Return word repeated the specified number of times.
    
    >>> repeat_word('Marcia ', 3)
    'Marcia Marcia Marcia '
    >>> repeat_word('Buffalo ', 5)
    'Buffalo Buffalo Buffalo Buffalo Buffalo '
    """
    return word*RepeatNum

(b) def function_name(first_name, last_name):
    """ (str, str) -> bool

    Returns True iff last_name is longer than first_name by comparing the values produced
    after calling built-in function len on first_name and last_name.

    >>> function_name("Myrto", "Papadopoulou")
    True
    >>> function_name("Jules", "Verne")
    False
    """
    if len(last_name) > len(first_name):
        return True
    else:
        return False

(c) def failing_grade(g):
    """ (int) -> bool

    Precondition: g must be of type int.

    Return True iff g is less than 50.
    """
    return (g<50)==True

def failed_two_courses(grade1, grade2):
    """
    (int, int) -> bool

    Returns True iff both grade1 and grade2 are less than 50.

    >>> failed_two_courses(10, 20)
    True
    >>> failed_two_courses(99, 45)
    False """
    return (grade1 < 50) and (grade2 < 50)
CSC108H Style Rules (a slightly more detailed list is referenced in Assignment’s 1 handout)

Formatting Style
1. Use Python style conventions for your function and variable names. This means that you should use pothole case: lowercase letters with words separated by underscores (_) to improve readability. For example, the variable name `dog_name` is good but `dogName` is not.
2. Choose meaningful names for your functions and variables. For example, `num_bright_spots` is more helpful and readable than `nbs`.
3. Do not use the tab character to indent code. Instead, use four spaces for each indentation level.
4. Put a blank space before and after every operator. For example, the first line below is good but the second line is not:
   
   ```
   b = 3 > x and 4 - 5 < 32
   b= 3>x and 4-5<32
   ```

5. For each function, write a docstring according to our design recipe. (See below for guidelines on the content of your docstrings.) Put the docstring’s closing quotation marks on their own line.
6. Each line must be less than 80 characters long including spaces. In Wing, the red vertical line indicates 80 characters. See Style Rules on the website for advice on how to break up long lines.
7. Within a docstring, put a blank line between the type contract and description, and between the description and the examples.
8. Put a blank line between the docstring and the function body.

Programming Style
1. Do not compare a Boolean expression to True or to False. For example, the first is good, the second is not:
   
   ```
   if a and b:
       return 100
   
   if (a and b) == True:
       return 100
   ```

2. Replace if statements of this form:

   ```
   if x > 100:
       return True
   else:
       return False
   ```

   with a single-line statement like this:

   ```
   return x > 100
   ```

3. Avoid duplicate code by calling on helper functions.

Docstring: Follow these rules when writing the description portion of the docstring:
1. Describe precisely what the function does.
2. Do not reveal how the function does it.
3. Make the purpose of every parameter clear.
4. Use the name of every parameter.
5. For functions that return values, be clear on what the return value represents.
6. Explain any conditions that the function assumes are true; these are called Preconditions. These conditions should not refer to type requirements, because those are already covered by the type contract. However, sometimes there are conditions not covered by types, and you should include those. For example, if a function requires parameters `x` and `y` to be even, include `x` and `y` must both be even. You can find an example of a Precondition in helper function `get_row` in puzzle_functions.py.
7. Be concise and grammatically correct.
8. Write the description as a command (e.g., Return the first ...) rather than a statement (e.g., Returns the first ...).