Question 1.  [5 marks]

Fill in the boxes below with what you would see in your Python shell if you were to type the following expressions. If that code would result in an error, then write ERROR and provide a brief explanation in the corresponding box.

(a) >>> word = 'breathe'
    >>> len(word)
    7

(b) >>> s = 'book'
    >>> (s < 'article') and (s[4] == 'k')
    False

(c) >>> l1 = ['a', 'b', 'c']
    >>> l1.isalpha()
    ERROR isalpha is a string method. Cannot be called on a list object.

(d) >>> y = 10 // 4
    >>> x = y
    >>> y = 20
    >>> x
    2

(e) >>> planets = ['earth', 'uranus']
    >>> planets[1][2]
    'a'

(f) # Assume the following function definitions:
    # def f(x):
    #     return 2 + x
    #     print("Function f")
    def g(y):
        print(f(y))
    >>> f(2)
    >>> g(10)

Common Mistakes/Reminders:

- In (b), string comparison uses dictionary order. Also, Python does lazy evaluation (short circuiting). False and anything evaluates to False, so Python will not even evaluate the (s[4] == 'k') expression which would have resulted in a string out of index error.

- In (f), the print statement in the body of function f after the return will not be executed.
Question 2.  [4 marks]

See below the definition for a function named mystery. Complete the two docstring examples below by filling in the boxes with the return values for these function calls. Also, fill in this function’s type contract. The function description has been left out on purpose.

```python
def mystery(val1, val2):
    """
    (int, int) -> int
    # Function description not shown on purpose.
    """

    for i in range(val1, val2, 1):
        if i % 4 == 0:
            return i
    return -1
```

Reminders:

- The range function only takes integers as arguments.
- i will never get a value equal to val2. The end is not part of the list-like sequence of integers that range generates.
Question 3.  [6 marks]
Write the body of the following function according to its docstring description.

```python
def compute_points(word):
    """ (str) -> int

    Return the number of points earned for word as follows.
    - Every character of word that is NOT a digit, earns 1 point.
    - Every character of word that is a digit, earns 10 points.
    - On top of that, every digit in word that is either 8 or 9, earns 5 more points.

    >>> compute_points('8')
    15
    >>> compute_points('year')
    4
    >>> compute_points('53year')
    24
    >>> compute_points('year-539')
    40
    ""

    points = 0
    for char in word:
        if char.isdigit():
            points = points + 10
            if int(char) == 8 or int(char) == 9:
                points = points + 5
        else:
            points = points + 1
    return points
```

Comments/Reminders:

- You need to use a numerical accumulator.
- Make sure that you don’t limit the functionality of your code. For example, do not assume that
  each character in word is either a digit or an alphabetic character; the last provided example shows
  otherwise.
- Pay attention to the type of a variable. For example, in the above solution, char is of type str.
  Therefore, it would be incorrect to compare it to an int without converting it first. Other correct
  solutions would have been:
    - if int(char) in [8, 9]: # this only works when char is digit. See next note.
    - if char == '9' or char == '8':
    - if char in '89':
- Note that it safe to do int(char), only if char is a digit. See nested if in the solution. Otherwise,
  this statement would result in an error.
Question 4. [4 marks]

Write the body of the following function according to its docstring description.

```python
def replicate_elements(my_list, repl_times):
    """ (list of str, int) -> NoneType
    Precondition: repl_times > 0
    Modify every element of my_list by replicating it repl_times times.
    >>> words = ['hi', '120']
    >>> replicate_elements(words, 2)
    >>> words
    ['hihi', '120120']
    """
    for i in range(len(my_list)):
        my_list[i] = my_list[i] * repl_times
```

Comments/Reminders:

- Always pay attention to the type contract/description/docstring examples. This function should NOT be returning a new list. This function should be modifying my_list.
[Use the space below for rough work. This page will not be marked unless you clearly indicate the part of your work that you want us to mark.]
Question 5. [6 marks]

Write the body of the following function according to its docstring description.

```python
def compare_word_length(list1, list2):
    """ (list of str, list of str) -> list of str

    Precondition: list1 and list2 both contain the same number of elements.

    Return a new list where every element is either 'same' or 'diff' depending on whether
    the words at the corresponding positions in list1 and list2
    have the same length or different length.
    >>> compare_word_length(['abc'], ['csc'])
    ['same']
    >>> compare_word_length(['code', 'in', 'Python'], ['read', 'many', 'books'])
    ['same', 'diff', 'diff']
    """
    result = []
    for i in range(len(list1)): # could be len(list2) too.
        if len(list1[i]) != len(list2[i]):
            result.append('diff')
        else:
            result.append('same')
    return result
```

Comments/Reminders:

- Since we want to access elements of the two lists in parallel, we need access to their indices. Pick the right for loop type! It makes coding easier.

- Doing `result = result + 'diff'` is incorrect. You cannot add a list and a string. If you do not use the append list method, you should write the following: `result = result + ['diff']`