1. [5 points] What does q1( ) print?

```python
def q1():
    x = 8
    y = 2
    x = x + y
    y = y - 2
    z = x + y
    print(z)
```

10

2. [5 points] What does q2( ) print? (Recall that // is integer division.)

```python
def q2():
    y = 5432
    x = 3
    while (x > 0):
        y = y // 10
        x = x - 1
    print(y)
```

5
3. [5 points] What does q3() print?

```python
def mx_mag(li):
    mx = 0
    for x in li:
        if x < 0:
            x = 0 - x
        if x > mx:
            mx = x
    return mx

def normalize(m):
    mag = mx_mag(m)
    for i in range(len(m)):
        m[i] = m[i] / mag

def q3():
    x = [ 10, -50, -100, 75 ]
    normalize(x)
    y = x[2]
    print(y)
```

-1.0 (or an approximation thereof)

4. [5 points] What does q4() print?

```python
def dbl(depth):
    if depth <= 0:
        return 1
    else:
        return 2 * dbl(depth - 1)

def q4():
    print(dbl(3))
```

8
5. [15 points] Complete the function `score_word`, consistent with its docstring. The Python quick reference sheet includes a reminder of how to use a `dict` structure like `VALUES`.

```python
# Values of individual letters in the game SCRABBLE
VALUES = {'a': 1, 'b': 3, 'c': 3, 'd': 2, 'e': 1, 'f': 4,
          'g': 2, 'h': 4, 'i': 1, 'j': 8, 'k': 5, 'l': 1,
          'm': 3, 'n': 1, 'o': 1, 'p': 3, 'q': 10,
          'r': 1, 's': 1, 't': 1, 'u': 1, 'v': 4, 'w': 4,
          'x': 8, 'y': 4, 'z': 10}

def score_word(word):
    ""
    Calculates the value of a word in the game SCRABBLE
    as the sum of its letter values (1 for 'a', 3 for 'm', etc).
    Args:
        word, a string containing only lower case letters
    Returns:
        the sum of the SCRABBLE values of the letters in word
    Examples:
        score_word("apple") = 1 + 3 + 3 + 1 + 1 = 9
        score_word("pear") = 3 + 1 + 1 + 1 = 6
        score_word("quince") = 10 + 1 + 1 + 3 + 1 = 17
        score_word("") = 0
    """
    score = 0
    for letter in word:
        score += VALUES[letter]
    return score
```
6. [15 points] Finish the function `max_step` below, consistent with the docstring.

```python
def max_step(li):
    """
    Determine the largest positive distance between adjacent elements of `ar`.
    Args:
        li: A list of integers
    Returns:
        The largest positive step from one element to the next; that is, the largest positive integer \( d \) such that, for some \( i \), \( d = li[i+1] - li[i] \), or 0 if there is no such positive integer.
    Examples:
        max_step([1, 2, 5, 6]) = 3 (biggest step is from 2 to 5)
        max_step([-5, -3, -2, -1]) = 2 (biggest step is from -5 to -3)
        max_step([5, 4, 3, 2, 1]) = 0 (there are no positive steps)
        max_step([42]) = 0 (there are no steps, therefore no positive steps)
        max_step([]) = 0 (there are no steps, therefore no positive steps)
    """
    if len(li) == 0:
        return 0
    prev = li[0]
    max = 0
    for el in li:
        delta = el - prev
        if delta > max:
            max = delta
        prev = el
    return max
```