Name: _____

1. Variable Tracing

For each of the following questions, write the output without running the code.

a. What will the following code print?

```
numbers = [8, 3, 5]

a = numbers[0]
b = numbers[2]
c = a + b
a = c - numbers[1]

print(a)
print(b)
print(c)
```

b. What will the following code print?

```
values = [10, 4, 7, 2]

x = values[1]
y = values[0] - values[3]
z = x + y
x = z - values[2]

print(x)
print(y)
print(z)
```

2. Printing and Loops

a. What will the following code print?

```
numbers = [3, 8, 1, 6, 9]

for i in range(len(numbers)):
    if numbers[i] % 2 == 0:
        print(i, numbers[i])
    else:
        print(numbers[i] - i)
```

b. What will the following code print?

```
letters = ["A", "B", "C", "D", "E"]

for i in range(len(letters) - 1, -1, -2):
    print(str(i) + ": " + letters[i])
```

c. What will the following code print?

```
words = ["red", "blue", "green"]

for i in range(len(words)):
    for j in range(i + 1):
        print(words[j], end=" ")
    print()
```

3. Function Calls and Recursion

For each of the following questions, write ***everything printed to the console***, in order, without running the code.

```
def countdown(n):
    print(n)

    if n == 0:
        return 1

    return countdown(n - 1) + 1

def mystery(n):
    print(n * 2)
    return countdown(n)
```

a. What will the following code print?

```
countdown(2)
```

b. What will the following code print?

```
print(countdown(2))
```

The following code snippet is given to you again for ease of reference.

```
def countdown(n):
    print(n)

    if n == 0:
        return 1

    return countdown(n - 1) + 1

def mystery(n):
    print(n * 2)
    return countdown(n)
```

c. What will the following code print?

```
print(mystery(2))
```

d. What will the following code print?

```
result = countdown(3)
print(result * 2)
```

4. Count Greater Than

Complete the `count_greater` function that takes a list of numbers and a target value, and returns the number of elements that are strictly greater than the target.

Examples:

- `count_greater([3, 8, 2, 10], 5)` returns 2.
- `count_greater([1, 2, 3], 7)` returns 0.
- `count_greater([5, 5, 5, 5], 5)` returns 0.
- `count_greater([-3, 7, -1, 8, 0], 0)` returns 2.
- `count_greater([12, 9, 15, 20], 10)` returns 3.

Your solution must use iteration *(do not use recursion)*.

```
def count_greater(numbers, target):

    count = __________________

    for ____________ in __________________:

        if _________________________________:

            __________________ += __________________

    return __________________
```

5. Is Sorted

Complete the `is_sorted` function that returns whether a list is sorted in non-decreasing order.

Examples:

- `is_sorted([1,2,2,5])` returns `True`.
- `is_sorted([1,3,2])` returns `False`.
- `is_sorted([7])` returns `True`.
- `is_sorted([5, 5, 5, 5])` returns `True`.
- `is_sorted([2, 4, 6, 8, 10])` returns `True`.

Your solution must use iteration *(do not use recursion)*.

```
def is_sorted(numbers):

    for ____________ in __________________:

        if _________________________________:

            return __________________

    return __________________
```

6. Count Odds

Write a recursive function `countOdds(numbers)` that returns the number of odd values in the list.

For example:

- `countOdds([3, 8, 5, 2, 7])` returns `3`
- `countOdds([2, 4, 6])` returns `0`
- `countOdds([])` returns `0`

Your solution must use recursion and must not use loops.

```
def countOdds(numbers):

    if ___________________________________:

        return ____________________________

    if ___________________________________:

        return ____________________________

    return ________________________________________
```

7. All Positives

Write a recursive function `allPositives(numbers)` that returns `True` if every value in the list is greater than `0`. Otherwise, return `False`.

For example:

- `allPositives([3, 19, 42, 7, 1, 12])` returns `True`
- `allPositives([5, 18, 0, 11, 27, 9])` returns `False`
- `allPositives([])` returns `True`

Your solution must use recursion and must not use loops.

```
def allPositives(numbers):

    if ___________________________________:

        return ____________________________

    if ___________________________________:

        return ____________________________

    return ________________________________________
```

8. Logarithm Review

Evaluate each expression.

a) $\log_2(64)$

b) $\log_3(81)$

c) $\log_5(125)$

d) $\log_{10}(1000)$

e) $\log_4(64)$

f) $\log_7(49)$

9. Exponential vs. Logarithmic Form

Rewrite each expression in the requested form.

a) Rewrite $2^6 = 64$ using logarithms.

b) Rewrite $5^3 = 125$ using logarithms.

c) Rewrite $\log_3(81) = 4$ using exponents.

d) Rewrite $\log_{10}(100) = 2$ using exponents.

e) Rewrite $7^0 = 1$ using logarithms.

f) Rewrite $\log_4\left(\frac{1}{16}\right) = -2$ using exponents.