Name: _____

1. Code Tracing

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

a. What will the next piece of code print?

```
my_list = ["ping", "pong", "tennis", "table", "ball", "pool"]

print(my_list[3])
print(my_list[len(my_list) - 5])
print(my_list[-2 + len(my_list)])

for i in range(6):
    print(my_list[(2 * i + 3) % 6])
```

b. What will the following code print?

```
def mystery(n):
    if n == 1:
        return 2

    return mystery(n - 1) + 2 * n

print(mystery(4))
```

2. Python Syntax

What is the purpose of a colon (`:`) in Python? Include at least three examples of where a colon is used.

3. For each expression, fill in the blank with one of the following types: `int`, `list`, `str`, `bool`, or `float`.

```
print(type(18 / 3))              # Printed: _____________________
print(type("Jam" + "Coders"))    # Printed: _____________________
print(type([True, 4, "hi"]))     # Printed: _____________________
print(type(7 == 7.0))            # Printed: _____________________
print(type(19 % 5))              # Printed: _____________________
```

4. For which value(s) of `x` and `y` will the following code print `"Blue_Mountain"`?

```
if x or y:
    print("Portland")
elif not x:
    print("Blue_Mountain")
else:
    print("Negril")
```

Fill in all boxes that apply.

- ☐ `x = True, y = True`
- ☐ `x = True, y = False`
- ☐ `x = False, y = True`
- ☐ `x = False, y = False`

5. What does the following code block print?

```
data = ["hello", ["yes", "no"], [], "nope"]
print(data[1:3][0:1])
```

6. Alternating Sum

Implement the `alternatingSum` function, which accepts a list of integers as its parameter and returns the alternating sum of the list. The alternating sum is computed by adding the elements at even indices and subtracting the elements at odd indices.

For example:

- `alternatingSum([10, 3, 6, 2, 5])` returns 16, since `10 - 3 + 6 - 2 + 5 = 16`
- `alternatingSum([4, 1, 7, 2])` returns 8, since `4 - 1 + 7 - 2 = 8`
- `alternatingSum([9])` returns `9`
- `alternatingSum([])` returns `0`

Part I: Iteration

Complete the iterative implementation below.

```
def alternatingSum(lst):

    total = ________________________

    for i in ______________________________________:

        if ____________________________________________:

            ____________________________________________

        else:

            ____________________________________________

    return ________________________
```

Part II: Recursion

Now complete the recursive implementation below.

```
def alternatingSum(lst):

    if ____________________________________________:

        return ________________________

    if ____________________________________________:

        return ________________________

    return ____________________________________________
```

7. Remove Consecutive Duplicates

Implement the `removeConsecutiveDuplicates` function, which accepts a string as its parameter and returns a new string with all consecutive duplicate characters removed.

For example:

- `removeConsecutiveDuplicates("aaabbccccdaa")` returns `"abcda"`
- `removeConsecutiveDuplicates("hello")` returns `"helo"`
- `removeConsecutiveDuplicates("ababab")` returns `"ababab"`

Part I: Iteration

Complete the iterative implementation below.

```
def removeConsecutiveDuplicates(s):

    result = ________________________

    for ch in ________________________:

        if ____________________________________________:

            ____________________________________________

    return ________________________
```

Part II: Recursion

Now complete the recursive implementation below.

```
def removeConsecutiveDuplicates(s):

    if ____________________________________________:

        return ________________________

    if ____________________________________________:

        return ____________________________________________

    return ____________________________________________
```

8. Consecutive Equal Digits

Implement the `hasConsecutiveEqualDigits` function, which accepts a non-negative integer as its parameter and returns whether the number contains two equal consecutive digits.

For example:

- `hasConsecutiveEqualDigits(122345)` returns `True`
- `hasConsecutiveEqualDigits(4551)` returns `True`
- `hasConsecutiveEqualDigits(12345)` returns `False`
- `hasConsecutiveEqualDigits(7)` returns `False`

Part I: Iteration

Complete the iterative implementation below.

```
def hasConsecutiveEqualDigits(n):

    while ________________________________________:

        lastDigit = ____________________________________
        remainingDigits = ______________________________
        secondToLastDigit = ____________________________

        if ____________________________________________:
            return ________________________

        ____________________________________________

    return ________________________
```

Part II: Recursion

Now complete the recursive implementation below.

```
def hasConsecutiveEqualDigits(n):

    if ____________________________________________:
        return ________________________

    lastDigit = ___________________________________

    remainingDigits = _____________________________

    secondToLastDigit = ___________________________

    if ____________________________________________:
        return ________________________

    return ____________________________________________
```

9. Box Outline

Write a function `drawBox(width, height)` that prints the outline of a box made of asterisks (`"*"`).

For example, the call `drawBox(6, 4)` should print:

```
******
*    *
*    *
******
```

The call `drawBox(3, 6)` should print:

```
***
* *
* *
* *
* *
***
```

You may assume that `width >= 2` and `height >= 2`.

Part I: String Multiplication

Complete the implementation below. You may only use string multiplication (`*`) and string concatenation (`+`) in your solution.

```
def drawBox(width, height):

    print(_____________________________________________)
    for i in range(_____________________________________):
        print(_____________________________________________)

    print(_____________________________________________)
```

Part II: Nested For Loops

Now implement the same function, but this time you may **not** use string multiplication. Instead, print each character individually using nested `for` loops.

```
def drawBox(width, height):

    for row in range(_____________________________________):
        for col in range(_____________________________):

            if (__________________ or __________________ or
                __________________ or __________________):

                print(_________________, end="")
            else:
                print(_________________, end="")
        print()
```