Name: _____

1. Code Tracing

a. What will the following code print?

```
word = "computer"

for i in range(len(word)):
    if i % 2 == 0:
        print(word[:i + 1])
    else:
        print(word[i:])
```

b. What will the following code print?

```
def mystery(s):
    if len(s) == 0:
        return

    print(s)
    mystery(s[1:])
    print(s)

mystery("cat")
```

2. Sum Until Zero

Write a recursive function `sumUntilZero(numbers)` that returns the sum of all values in the list until the first occurrence of `0`. If the first value is `0`, return `0`.

For example:

- `sumUntilZero([3, 7, 2, 0, 5, 8])` returns `12`
- `sumUntilZero([5, 4, 1, 0])` returns `10`
- `sumUntilZero([0, 4, 5])` returns `0`

Your solution must use recursion and must not use loops.

```
def sumUntilZero(numbers):

    if ___________________________________:

        return ____________________________

    if ___________________________________:

        return ____________________________

    return ________________________________________
```

3. Alternate Capitalization

Write a recursive function `alternateCase(s, upper)` that returns a new string where every other character alternates between uppercase and lowercase letters. You may assume that the first call will always pass `True` for `upper`.

For example:

- `alternateCase("computer", True)` returns `"CoMpUtEr"`
- `alternateCase("hello", True)` returns `"HeLlO"`
- `alternateCase("python", True)` returns `"PyThOn"`

Your solution must use recursion and must not use loops.

```
def alternateCase(s, upper):

    if ___________________________________:

        return ____________________________

    if ___________________________________:

        return ____________________________________

    return ________________________________________
```

4. Finding Sums

Fill in the `findSum` function that takes a **sorted** list of *distinct* integers `numbers` and a target value `x`. The function returns whether there are two values in the list whose sum equals `x`.

Examples:

- `findSum([1, 3, 4, 6, 8, 11], 10)` returns `True`
- `findSum([2, 5, 7, 9, 12], 20)` returns `False`
- `findSum([1, 2, 3, 4, 5], 9)` returns `True`

```
def findSum(numbers, x):

    for __________________ in ___________________:

        for __________________ in ___________________:

            if ___________________________________:

                return ______________________

    return ______________________
```

This week, you will learn to analyze code and develop more efficient solutions. The solution above uses nested `for` loops, which is not the most efficient approach. This problem can actually be solved using a *single* `while` loop.

Hint: Does the sorted order of the list help?

```
def findSum(numbers, x):

    left = 0
    right = _________________________

    while _______________ < _______________:

        if ________________________________ == x:

            return ______________________

        elif ________________________________ < x:

            ________________________________

        else:

            ________________________________

    return ______________________
```

5. Name Diamond

Fill in the `nameDiamond` function that accepts a string `name` as a parameter and prints it in a "diamond" format as shown below.

For example, the call of `nameDiamond("SUSAN")` should print:

```
S
SU
SUS
SUSA
SUSAN
 USAN
  SAN
   AN
    N
```

The call of `nameDiamond("bob")` should print:

```
b
bo
bob
 ob
  b
```