Name: _____

1. Code Tracing

Two functions can call ***each other*** recursively instead of calling themselves. It's the World Cup Final, and it's gone to a penalty shootout between Team Spain and Team Argentina, so they alternate turns until the rounds run out.

```
def spain_kicks(rounds):
    if rounds == 0:
        print("Shootout over!")
        return "Sudden death!"
    print("Team Spain steps up to kick. Rounds left:", rounds)
    return argentina_kicks(rounds - 1)

def argentina_kicks(rounds):
    if rounds == 0:
        print("Shootout over!")
        return "Sudden death!"
    print("Team Argentina steps up to kick. Rounds left:", rounds)
    return spain_kicks(rounds - 1)
```

What will be printed when the following code is executed?

```
result = spain_kicks(2)
print(result)
```

Write the complete output below.

2. Count Digits

Write a recursive function called `count_digits` that takes a positive integer `n` and returns how many digits it has.

For example, suppose we execute the following line:

```
print(count_digits(2026))
```

The code above should produce the following output:

```
4
```

Your function ***must use recursion*** and ***MUST NOT USE A LOOP OR THE `str()` FUNCTION***.

Hint: Think about how `n // 10` gets smaller each time. Write clear indentation.

```
def count_digits(n):
    # Write your code below each comment
    # BASE CASE:

    if _______________________________:

        return _______________________________

    # RECURSIVE CASE:

    return _______________________________
```

3. Common Prefix

Write a recursive function called `common_prefix` that takes two strings, `s1` and `s2`, and returns the longest string that ***both*** strings start with.

For example, suppose we execute the following line:

```
print(common_prefix("interview", "internet"))
```

The code above should produce the following output:

```
inter
```

Your function ***must use recursion*** and must ***NOT*** use a loop.

Hint: Compare `s1[0]` and `s2[0]`. If they're the same letter, that letter is part of the shared prefix. If they're different (or either string runs out), the shared portion of the string stops right there. Be careful with indentation.

```
def common_prefix(s1, s2):
    # Write your code below the comment
```

4. *Challenge*: Generating All Subsets

Given a list of elements, a ***subset*** is any group of elements chosen from the list. You may choose none of the elements, some of the elements, or all of the elements. For example, the subsets of `["a", "b"]` are: `[]`, `["a"]`, `["b"]`, and `["a", "b"]`.

Write a recursive function called `generate_subsets` that takes `elements` (the original list), `index` (the index of the element you are currently deciding on), and `current_subset` (the subset you have built so far), and ***prints*** every possible subset.

For example, suppose we make the following function call:

```
generate_subsets(['a', 'b', 'c'], 0, [])
```

This call should print the following output, in any order:

```
['a', 'b', 'c']
['a', 'b']
['a', 'c']
['a']
['b', 'c']
['b']
['c']
[]
```

Your function ***must use recursion*** and must ***NOT*** use a loop.

```
def generate_subsets(elements, index, current_subset):
    # Write your code below the comment
```