Name: _____

1. List Mutation Tracing

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

a. What will the following code print?

```
numbers = [4, 7, 2]

numbers[0] = numbers[1]
numbers[2] = numbers[0] + numbers[2]
numbers[1] = numbers[2] - 1

print(numbers)
```

b. What will the following code print?

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

temp = values[0]
values[0] = values[2]
values[2] = temp

print(values)
```

c. What will the following code print?

```
scores = [6, 1, 4]

scores[1] = scores[0] - scores[2]
scores.append(scores[1] + scores[2])
scores[0] = scores[3] - scores[1]

print(scores)
```

2. Printing and Loops

a. What will the following code print?

```
count = 1
total = 0
while count <= 5:
    total += count
    if total > 6:
        print(total)
    else:
        print(count)
    count += 1
```

b. What will the following code print?

```
message = "JAMAICA"
for i in range(0, len(message), 2):
    print(message[i] * (i + 1))
```

c. What will the following code print?

```
for row in range(1, 5):
    for col in range(1, row + 1):
        if (row + col) % 2 == 0:
            print("*", end="")
        else:
            print("-", end="")
    print()
```

3. Count Odds

Write an iterative 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 iteration and must not use recursion.

```
def countOdds(numbers):

    count = ________________________

    for ____________________________:

        if ____________________________:

            ________________________

    return ________________________
```

4. All Positives

Write an iterative 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 iteration and must not use recursion.

```
def allPositives(numbers):

    for ____________________________:

        if ____________________________:

            return ________________

    return ________________________
```

5. Count Greater Than

Complete the recursive function `count_greater(numbers, target)` 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.

Your solution must use recursion and must not use loops.

```
def count_greater(numbers, target):

    if ___________________________________:

        return ____________________________

    if ___________________________________:

        return ____________________________

    return ________________________________________
```

6. Is Sorted

Complete the recursive function `is_sorted(numbers)` 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`.

Your solution must use recursion and must not use loops.

```
def is_sorted(numbers):

    if ___________________________________:

        return ____________________________

    if ___________________________________:

        return ____________________________

    return ________________________________________
```

7. Sorting

Suppose we create the following list:

```
numbers = [77, 3, 62, 11, 8, 54, 42, 19]
```

Part I: Selection Sort

Suppose we call `selection_sort(numbers)`. Write the contents of the list after each pass of the outermost loop in the Selection Sort algorithm (i.e., after each element has been selected and placed in its final position).

After Pass 1: _____

After Pass 2: _____

After Pass 3: _____

After Pass 4: _____

After Pass 5: _____

After Pass 6: _____

After Pass 7: _____

Part II: Merge Sort

Now suppose we call `merge_sort(numbers)` on the same list instead. Show each group of sublists produced during the splitting stage and each merge step that produces a larger sorted list.

1st Split: _____

2nd Split: _____

3rd Split: _____

1st Merge: _____

2nd Merge: _____

3rd Merge: _____

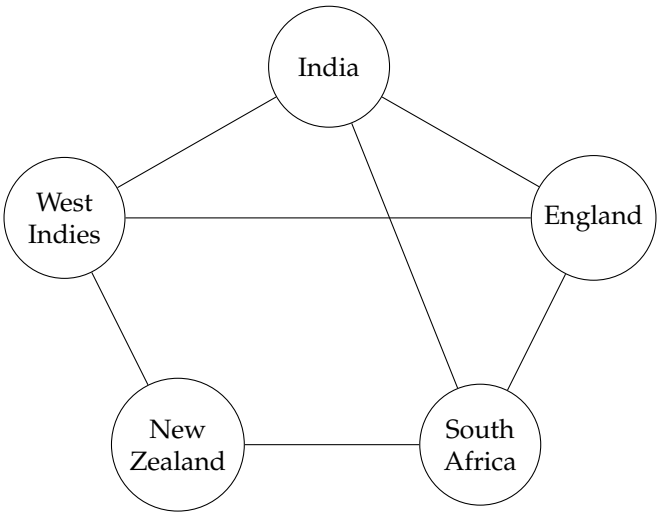
8. Cricket Tournament Graph

Five cricket teams are playing in a tournament:

- West Indies
- India
- England
- South Africa
- New Zealand

An **edge** *(a line)* between two teams means that the teams have played a match against each other.

Consider the following undirected graph:



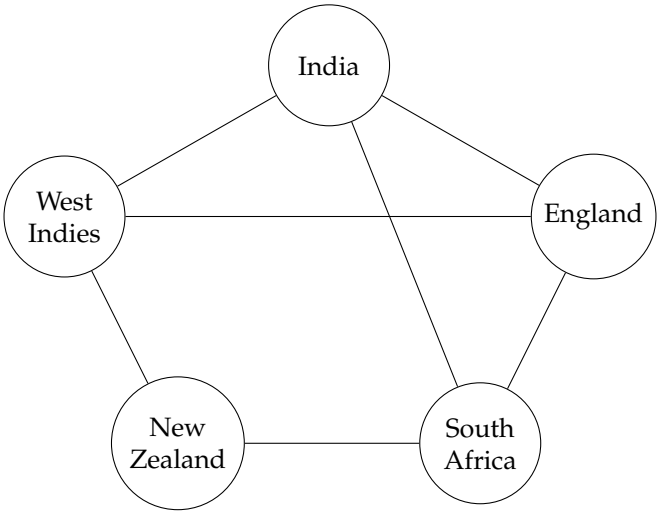
a. Based on the graph, circle each team that the West Indies has played.

India England South Africa New Zealand

b. Complete the adjacency list for the tournament graph.

```
West Indies:   ____________________________________
India:         ____________________________________
England:       ____________________________________
South Africa:  ____________________________________
New Zealand:   ____________________________________
```

The graph is shown again for ease of reference.



c. Complete the adjacency matrix. Write `1` if the two teams have played against each other and `0` otherwise.

	WI	I	E	SA	NZ
WI					
I					
E					
SA					
NZ					

WI = West Indies **I** = India **E** = England

SA = South Africa **NZ** = New Zealand

d. Which representation directly lists every team that a team has played against?

Adjacency List Adjacency Matrix