Q1. Write the output of following snippet

```
for num in range(8):
        print("Hello" * num)
```

Q2. Write the output of following snippets

```
my_list = ["a", "b", "c", "d"]
for i in range(2, len(my_list)-1):
        my_list[i] *= i
```

Q3. Write the output of following snippets

```
word = "Hello"
for char in word.lower(): # calling the string method
        print(char)
```

Q4. Write the output of following snippets

```
exp = 4
base = 2
result = 1
for x in range(exp):
  result = base * result

print(result)
```

Q5. Write the output of following snippets

```
counter = 0
while(counter < 10):
  print(counter)
  counter = counter + 1
print('while loop ended')
```

Q6. Trace the execution of the following code and determine the output:

```
def multiply(a, b):
  if b == 0:
    return 0
  elif b % 2 == 0:
    recursive_result = multiply(a, b // 2)
    return recursive_result + recursive_result
  else:
    recursive_result = multiply(a, b // 2)
    return a + recursive_result + recursive_result

result = multiply(3, 5)
print(result)
```

Q7. Walk through the code and determine the output of the program:

```
def fibonacci(n):
    if n <= 1:
        return n
    else:
        return fibonacci(n-1) + fibonacci(n-2)

num_terms = 6
fib_sequence = [fibonacci(i) for i in range(num_terms)]
print(fib_sequence)
```

Q8. Trace the execution of the code and determine the final value of product:

```
def calculate_product(numbers):
    if len(numbers) == 1:
        return numbers[0]
    else:
        recursive_result = calculate_product(numbers[1:])
        return numbers[0] * recursive_result

data = [2, 3, 4, 5]
product = calculate_product(data)

print(product)
```

Q9. Walk through the code and determine the output of the program:

```
def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    mid = len(arr) // 2
    left = merge_sort(arr[:mid])
    right = merge_sort(arr[mid:])

    merged = []
    i, j = 0, 0
    while i < len(left) and j < len(right):
        if left[i] < right[j]:
            merged.append(left[i])
            i += 1
        else:
            merged.append(right[j])
            j += 1

    merged.extend(left[i:])
    merged.extend(right[j:])
    return merged

data = [9, 3, 7, 1, 5]
sorted_data = merge_sort(data)
```

```
print(sorted_data)
```

Q10. Trace the execution of the code and determine the final value of result:

```
def power(base, exponent):
    if exponent == 0:
        return 1
    elif exponent % 2 == 0:
        recursive_result = power(base, exponent // 2)
        return recursive_result * recursive_result
    else:
        recursive_result = power(base, exponent // 2)
        return base * recursive_result * recursive_result

result = power(2, 5)
print(result)
```

Q11. Write the output of following snippets

```
def merge_dicts(dict1, dict2):
    result = dict1.copy()
    for key, value in dict2.items():
        if key in result and isinstance(result[key], dict) and isinstance(value, dict):
            result[key] = merge_dicts(result[key], value)
        else:
            result[key] = value
    return result

dict1 = {'a': 1, 'b': {'c': 2}}
dict2 = {'b': {'d': 3}, 'e': 4}
merged_dict = merge_dicts(dict1, dict2)

print(merged_dict)
```

Q12. Write the output of following snippets

```
def count_values(dictionary):
    value_count = {}

    for value in dictionary.values():
        if isinstance(value, dict):
            nested_count = count_values(value)
            for key, count in nested_count.items():
                value_count[key] = value_count.get(key, 0) + count
        else:
            value_count[value] = value_count.get(value, 0) + 1

    return value_count

data = {'a': 1, 'b': {'c': 1, 'd': {'e': 2}}, 'f': 2}
value_counts = count_values(data)

print(value_counts)
```

Q13. Write the output of following snippets

```
my_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Perform the following operations step by step:

A. Access the element 5 in the nested list.
B. Change the value of 8 to 10.
C. Append a new sublist [11, 12, 13] to the end of the nested list.
D. Print the updated nested list.