Student Name: Jay Prakash **Subject:** Python Programming **Exam Type:** Practical

Question a

Create a function that returns another function to compute a power of a number, which itself returns a function to multiply the result by a constant.

```
# ------------------------------------------------------------
# Program: Nested Functions for Power and Multiplication
# Description: Demonstrates function returning functions
# ------------------------------------------------------------

def power_function(power):
    """
    This function takes a power value and returns another function.
    """
    def compute_power(number):
        """
        This function computes number raised to given power
        and returns another function.
        """
        result = number ** power

        def multiply_by_constant(constant):
            """
            This function multiplies the computed power result
            by a constant.
            """
            return result * constant

        return multiply_by_constant

    return compute_power


# ------------------ Execution ------------------

# Creating power function for square
square = power_function(2)

# Computing square of 5
square_of_5 = square(5)

# Multiplying result by constant 3
final_result = square_of_5(3)

# Displaying output
print("\n========== OUTPUT ==========")
print("Power: 2")
print("Number: 5")
print("Multiplier: 3")
print("Final Result:", final_result)
print("============================")
```

```
========== OUTPUT ==========
Power: 2
Number: 5
Multiplier: 3
Final Result: 75
============================
```

Explanation:

This program demonstrates nested functions.

1. The outer function takes a power value.
2. The second function computes the power of a number.
3. The innermost function multiplies the computed result by a constant.

4. This shows how functions can return other functions in Python.

Question b

Write a function that compresses a string by counting consecutive characters. Example: "aaabbcc" becomes "a3b2c2". Ensure it handles edge cases like single characters or empty strings.

```
# ------------------------------------------------------------
# Program: String Compression
# Description: Compress string by counting consecutive characters
# ------------------------------------------------------------

def compress_string(text):
    """
    Compresses a string by counting consecutive characters.
    """
    # Edge case: empty string
    if not text:
        return ""

    compressed = ""
    count = 1

    # Loop through string characters
    for i in range(1, len(text)):
        if text[i] == text[i - 1]:
            count += 1
        else:
            compressed += text[i - 1] + str(count)
            count = 1

    # Add last character count
    compressed += text[-1] + str(count)

    return compressed


# ------------------ Execution ------------------

input_string = "aaabbcc"
output_string = compress_string(input_string)

print("\n========== OUTPUT ==========")
print("Original String :", input_string)
print("Compressed String:", output_string)
print("============================")
```

```
========== OUTPUT ==========
Original String : aaabbcc
Compressed String: a3b2c2
============================
```

Explanation:

This program compresses a string by counting consecutive characters.

1. If the string is empty, it returns an empty result.
2. It iterates through the string and counts repeated characters.
3. When a new character appears, the previous character and count are added.
4. The final compressed string is returned.

Question c

Create a class Flight to manage flights with attributes like flight number, source, destination, and capacity. Add methods to:

i.Book a ticket.

ii.Cancel a ticket.

iii. Display available seats.

```
# ----------------------------------------------------------
# Program: Flight Management System
# Description: Manage flight booking and cancellation
# ----------------------------------------------------------

class Flight:
    def __init__(self, flight_number, source, destination, capacity):
        """
        Constructor to initialize flight details.
        """
        self.flight_number = flight_number
        self.source = source
        self.destination = destination
        self.capacity = capacity
        self.booked_seats = 0

    def book_ticket(self):
        """
        Books a ticket if seats are available.
        """
        if self.booked_seats < self.capacity:
            self.booked_seats += 1
            print("Ticket booked successfully.")
        else:
            print("No seats available.")

    def cancel_ticket(self):
        """
        Cancels a booked ticket.
        """
        if self.booked_seats > 0:
            self.booked_seats -= 1
            print("Ticket cancelled successfully.")
        else:
            print("No tickets to cancel.")

    def display_available_seats(self):
        """
        Displays available seats.
        """
        available = self.capacity - self.booked_seats
        print("Available Seats:", available)


# ------------------ Execution ------------------

flight = Flight("AI101", "Delhi", "Bangalore", 5)

print("\n========== FLIGHT DETAILS ==========")
flight.book_ticket()
flight.book_ticket()
flight.display_available_seats()
flight.cancel_ticket()
flight.display_available_seats()
print("====================================")
```

```
========== FLIGHT DETAILS ==========
Ticket booked successfully.
Ticket booked successfully.
Available Seats: 3
Ticket cancelled successfully.
Available Seats: 4
====================================
```

Explanation:

This program implements a Flight class.

1. The constructor initializes flight details.
2. The book_ticket() method increases booked seats if capacity allows.

3. The cancel_ticket() method reduces booked seats safely.
4. The display_available_seats() method shows remaining seats.
5. This demonstrates object-oriented programming in Python.