Name: Youssef Salem Anwer Salem Hassan

Problem 1: Calculator Using Match-Case

Task: Create a calculator program that performs basic arithmetic operations. The program should:

- Prompt the user to enter two numbers.
- Prompt the user to enter an operator (+, -, *, /, %).
- Use a match-case statement to perform the operation.
- Display the result.

Example:

Enter the first number: 10
Enter the second number: 5
Enter an operator (+, -, *, /, %): /
Result: 2.0

Additional Requirement:

Handle division by zero and invalid operators gracefully by displaying appropriate error messages.

```python
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
operator = input("Enter an operator (+, -, *, /, %): ").strip()

match operator:
      case '+':
          print(f"Result: {num1 + num2}")
      case '-':
          print(f"Result: {num1 - num2}")
      case '*':
          print(f"Result: {num1 * num2}")
      case '/':
          if num2 == 0:
              print("Error: Division by zero is not allowed.")
          else:
              print(f"Result: {num1 / num2}")
      case '%':
          if num2 == 0:
              print("Error: Division by zero is not allowed.")
          else:
              print(f"Result: {num1 % num2}")
```

```
Enter the first number: 8
Enter the second number: 4
Enter an operator (+, -, *, /, %): *
Result: 32.0
```

Problem 2: Pattern Generation with For Loops

Task:

Write a Python program that generates a number pattern based on a user-defined limit `n`. The pattern should incrementally increase the number of elements in each row, starting from `1`, until the next number would exceed `n`. Use nested for loops to achieve this.

Example: If `n = 10`, the output should be:

1

2 3

4 5 6

7 8 9 10

Requirements:

- Use nested for loops.
- The program should prompt the user for the value of n, (explained below in the hint)
- Ensure that the pattern stops adding numbers when the next number would exceed n.

```
n = int(input("Enter the value of n: "))

current_number = 1

for i in range(1, n + 1):
    row = []
    for j in range(i):
        if current_number > n:
            break
        row.append(current_number)
        current_number += 1
    if not row:
        break
    print(" ".join(map(str, row)))
```

```
Enter the value of n: 28
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
16 17 18 19 20 21
```

```
22 23 24 25 26 27 28
```

Problem 3: List Comparison and Analysis

Task: Write a Python program that, given two lists of integers list_a and list_b, performs the following operations:

- Finds and displays a list of elements that are common to both lists.
- Finds and displays a list of elements that are only in list_a.
- Finds and displays a list of elements that are only in list_b.
- Calculates and displays the sum of the absolute differences between corresponding elements up to the length of the shorter list.

Example:

Given:

list_a = **[1, 2, 3, 4]**
list_b = **[3, 4, 5, 6]**

The program should output:

Common elements: **[3, 4]**
Unique to list_a: **[1, 2]**
Unique to list_b: **[5, 6]**
Sum of differences: **8**

Explanation of Sum of Differences:

Calculate the sum of absolute differences between corresponding elements:

|1 - 3| = 2
|2 - 4| = 2
|3 - 5| = 2
|4 - 6| = 2

Total sum: 2 + 2 + 2 + 2 = 8

Requirements:

- Use list comprehensions not sets.
- Ensure the program handles lists of different lengths.
- The original lists should not be modified.
- Do not use functions; write the code in the main body of the program.

```
list_a = [1, 2, 3, 4]
list_b = [3, 4, 5, 6]
```

```
# common elements
common_elements = [x for x in list_a if x in list_b]
print(f"Common elements: {common_elements}")

# unique to list_a
unique_to_a = [x for x in list_a if x not in list_b]
print(f"Unique to list_a: {unique_to_a}")

# unique to list_b
unique_to_b = [x for x in list_b if x not in list_a]
print(f"Unique to list_b: {unique_to_b}")

# Calculate sum of absolute differences
min_length = min(len(list_a), len(list_b))
sum_of_differences = sum(abs(list_a[i] - list_b[i]) for i in range(min_length))
print(f"Sum of differences: {sum_of_differences}")
```

```
Common elements: [3, 4]
Unique to list_a: [1, 2]
Unique to list_b: [5, 6]
Sum of differences: 8
```

Problem 4: Collatz Conjecture with While Loop

Task

Write a Python program that computes and displays the Collatz sequence for a given positive integer `n`.

Steps to Follow:

- Prompt the user to input a positive integer.
- Check if the input is valid (greater than 0). If not, prompt the user again.
- Compute the Collatz sequence using a while loop:
  - If n is even, the next number is **n / 2**.
  - If n is odd, the next number is **3 * n + 1**.
  - Append each value to a list.
- Continue the process until **n** becomes 1.

Example: If the user inputs 6, the program should output:

Collatz sequence: [6, 3, 10, 5, 16, 8, 4, 2, 1]

Requirments:

- Use a while loop.
- Validate that the input n is a positive integer.

- Do not use functions; write the code in the main body of the program.
- The sequence should be stored in a list and displayed after the computation.

```
while True:
    try:
        n = int(input("Enter a positive integer: "))
        if n > 0:
            break
        else:
            print("The number must be greater than 0.")
    except ValueError:
        print("Invalid input. Please enter a positive integer.")

collatz_sequence = [n]

# Collatz sequence
while n != 1:
    if n % 2 == 0:  # Even number
        n = n // 2
    else:  # Odd number
        n = 3 * n + 1
    collatz_sequence.append(n)

print(f"Collatz sequence: {collatz_sequence}")
```

```
Enter a positive integer: 6
Collatz sequence: [6, 3, 10, 5, 16, 8, 4, 2, 1]
```