1. Day computations:
10 days, 39 days, 1596 days

2. Code Explanation
This code calculates the number of days between two given dates. It first ensures that the input dates are valid and swaps them if the first date is later than the second. It then computes the total days by summing the remaining days in the first year, the full years in between (assuming 365 days per year), and the days passed in the final year. Finally, it accounts for leap years and prints the total number of days between the two dates.

3. My Pseudocode
```
DEFINE month_days AS LIST [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

FUNCTION day_count(first_month, first_day, first_year, second_month, second_day,
second_year):
    ASSERT first_month BETWEEN 1 AND 12
    ASSERT second_month BETWEEN 1 AND 12
    ASSERT first_day BETWEEN 1 AND month_days[first_month - 1]
    ASSERT second_day BETWEEN 1 AND month_days[second_month - 1]
    ASSERT first_year >= 1753
    ASSERT second_year >= 1753

    IF (first_year, first_month, first_day) > (second_year, second_month,
second_day) THEN
        SWAP first_month AND second_month
        SWAP first_day AND second_day
        SWAP first_year AND second_year
    ENDIF

    SET days_left_in_month = month_days[first_month - 1] - first_day
    SET year_days = days_left_in_month

    FOR i FROM first_month TO 11 DO
        year_days = year_days + month_days[i]
    ENDFOR

    SET year_count = second_year - first_year - 1
    SET day_count = year_count * 365 + year_days

    SET days_passed_in_last_year = SUM(month_days[0] TO month_days[second_month -
2]) + second_day
    day_count = day_count + days_passed_in_last_year

    SET leap_years = 0
    FOR i FROM first_year TO second_year - 1 DO
        IF (i MOD 4 == 0 AND i MOD 100 != 0) OR (i MOD 400 == 0) THEN
            leap_years = leap_years + 1
        ENDIF
    ENDFOR
    day_count = day_count + leap_years

    RETURN day_count
END FUNCTION

PRINT "Month (1-12): "
INPUT first_month

PRINT "Day (1-31): "
INPUT first_day

PRINT "Year: "
INPUT first_year

PRINT "Month (1-12): "
INPUT second_month

PRINT "Day (1-31): "
INPUT second_day

PRINT "Year: "
INPUT second_year

PRINT "The number of days between first_month/first_day/first_year and
second_month/second_day/second_year is "
PRINT day_count(first_month, first_day, first_year, second_month, second_day,
second_year)
```

4. CoPilot's Pseudocode

```
    1. Define a list `month_days` containing the number of days in each month.

    2. Define a function `day_count` that takes six parameters:
    `first_month`, `first_day`, `first_year`, `second_month`, `second_day`, and
`second_year`.

    3. In the function:
    a. Validate input values:
        - Ensure `first_month` and `second_month` are between 1 and 12.
        - Ensure `first_day` and `second_day` are within the valid range for their
respective months.
        - Ensure `first_year` and `second_year` are greater than or equal to 1753.

    b. Check if the first date is later than the second date, and if so, swap the
dates.

    c. Calculate the number of days left in the first year:
        - Subtract `first_day` from the total days in the `first_month`.
        - Add the days from the remaining months of the `first_year`.
    d. Calculate the full years between the two dates:
        - Subtract `first_year` from `second_year` and adjust for the first and
second months.
        - Multiply the number of full years by 365 to get the total days.

    e. Calculate the number of days passed in the last year:
        - Sum the days from the beginning of the year up to the `second_month` and
add `second_day`.

    f. Count leap years between `first_year` and `second_year` and add the extra
days for leap years.

    g. Return the total number of days.

    4. Prompt the user for input values:
    - `first_month`
    - `first_day`
    - `first_year`
    - `second_month`
    - `second_day`
    - `second_year`

    5. Call the `day_count` function with the input values and print the result.
```

5. Compare/Contrast
While CoPilot's pseudocode is more descriptive, I don't think I could call it 'code' anymore. Pseudocode is supposed to resemble code, not complete sentences with English grammer.

6. Update my Solution
I don't think mine needs to be updated when compared to CoPilot's solution.

7. Program Trace

```
        Step    Line Executed   Variable Changes / Computation  Output / Comments
        1       day_count(11, 17, 2002, 4, 6, 2004)     Function call starts
        2       assert checks   Passes all assertions
        3       if (first_year, first_month, first_day) > (second_year,
second_month, second_day)       Condition false, no swap
        4       days_left_in_month = month_days[first_month - 1] - first_day
days_left_in_month = 30 - 17 = 13
        5       year_days = days_left_in_month  year_days = 13
        6       for i in range(first_month, 12) Loop starts (November to December)

        7       i = 11 (December)       year_days += 31 year_days = 13 + 31 = 44
        8       Loop exits
        9       year_count = second_year - first_year - 1       year_count = 2004 -
2002 - 1 = 1
```

10	day_count = year_count * 365 + year_days	day_count = 1 * 365 + 44 = 409	
11	days_passed_in_last_year = sum(month_days[:second_month - 1]) + second_day	days_passed_in_last_year = sum([31, 28, 31, 30]) + 6 = 126 + 6 = 132	
12	day_count += days_passed_in_last_year	day_count = 409 + 132 = 541	
13	for i in range(first_year, second_year)	Loop starts (2002 to 2004)	
14	i = 2002	Leap year check: not a leap year	
15	i = 2003	Leap year check: not a leap year	
16	i = 2004	Leap year check: leap year	leap_years = 1
17	Loop exits	leap_years = 1	
18	day_count += leap_years	day_count = 541 + 1 = 542	
19	return day_count	542	Output: 542 days

8. Algorithmic Efficiency
O(n) where n is the difference in years between the two dates

Step 1 By Hand: 15 minutes
Step 2 Approach: 50 minutes
Step 3 Pseudocode: 30 minutes
Step 4 Copilot: 6 minutes
Step 5 Compare and Contrast: 4 minutes
Step 6 Update: 3 minutes
Step 7 Trace: 30 minutes
Step 8 Efficiency: 15 minutes