Bài 1: Cho hàm đệ quy để tính tổng các số từ 1 đến n. Hãy giải thích từng bước thực hiện của hàm đệ quy này khi n = 7

```
1 def sum_of_numbers(n):
2     if n == 1:
3         return 1
4     else:
5         return n + sum_of_numbers(n-1)
6 print(sum_of_numbers(7))
```

Quy trình các bước như sau:

1. Bước 1: Gọi **sum_of_numbers(7)**
2. Bước 2: Vì **n** không bằng 1, nó đi vào phần else của câu lệnh điều kiện.
3. Bước 3: Hàm gọi **sum_of_numbers(6)** và trả về **7 + sum_of_numbers(6)**
4. Bước 4: Trong **sum_of_numbers(6)**:
   - Vì **n** không bằng 1, nó tiếp tục đi vào phần else.
   - Hàm gọi **sum_of_numbers(5)** và trả về **6 + sum_of_numbers(5)**
5. Bước 5: Quá trình này lặp lại cho đến khi **n** bằng 1. Khi **n** bằng 1, hàm trả về 1 và các lời gọi đệ quy ngưng.

6. Bước 6: Kết quả trả về từ lời gọi **sum_of_numbers(1)** là 1.
7. Bước 7: Khi quay lại từ đệ quy, các giá trị được tính từ **sum_of_numbers(2)** đến **sum_of_numbers(7)** được cộng lại.
8. Bước 8: Kết quả cuối cùng là **1 + 2 + 3 + 4 + 5 + 6 + 7 = 28**.

Vì vậy, kết quả của lời gọi **sum_of_numbers(7)** là 28.

Bài 2: Cho hàm đệ quy để tính số Fibonacci thứ n. Hãy giải thích từng bước thực hiện của hàm đệ quy này khi n = 8.

```
1 def fibonacci(n):
2     if n <= 1:
3         return n
4     else:
5         return fibonacci(n-1) + fibonacci(n-2)
6 print(fibonacci(8))
```

Quy trình thực hiện các bước như sau:

1. Bước 1: Gọi **fibonacci(8)**
2. Bước 2: Vì **n = 8** không nhỏ hơn hoặc bằng 1, nó đi vào phần else của câu lệnh điều kiện.
3. Bước 3: Hàm gọi đệ quy **fibonacci(7)** và **fibonacci(6)**.
4. Bước 4: Trong lời gọi **fibonacci(7)**:

- Vì **n = 7** không nhỏ hơn hoặc bằng 1, nó tiếp tục gọi đệ quy **fibonacci(6)** và **fibonacci(5)**.

5. Bước 5: Quá trình này tiếp tục cho đến khi nó gọi các số Fibonacci nhỏ hơn hoặc bằng 1, khi đó chúng sẽ trả về chính giá trị của **n**.
6. Bước 6: Kết quả của các lời gọi **fibonacci(2)**, **fibonacci(1)**, **fibonacci(0)** là lần lượt 1, 1, và 0.
7. Bước 7: Khi đã tính được **fibonacci(2)** và **fibonacci(1)**, hàm sẽ tính **fibonacci(3)** bằng cách cộng hai số Fibonacci trước đó là **fibonacci(2)** và **fibonacci(1)**, kết quả là 2.
8. Bước 8: Tương tự, nó tính **fibonacci(4)** bằng cách cộng **fibonacci(3)** và **fibonacci(2)**, kết quả là 3.
9. Bước 9: Quá trình này tiếp tục cho đến khi nó tính được **fibonacci(8)** bằng cách cộng **fibonacci(7)** và **fibonacci(6)**, kết quả là 21.

Vậy kết quả của lời gọi **fibonacci(8)** là 21.

Bài 3: Cho hàm đệ quy để tính x mũ n. Hãy giải thích từng bước thực hiện của hàm đệ quy này khi x = 2 và n = 6.

```
1 def power(x, n):
2     if n == 0:
3         return 1
4     else:
5         return x * power(x, n-1)
6 print(power(2, 6))
```

Quy trình thực hiện các bước như sau:

1. Bước 1: Gọi **power(2, 6)**
2. Bước 2: Vì **n = 6** không bằng 0, nó đi vào phần else của câu lệnh điều kiện.
3. Bước 3: Hàm gọi **power(2, 5)** và trả về **2 * power(2, 5)**
4. Bước 4: Trong lời gọi **power(2, 5)**:
    - Vì **n = 5** không bằng 0, nó tiếp tục gọi đệ quy **power(2, 4)** và trả về **2 * power(2, 4)**
    - Quá trình này tiếp tục cho đến khi **n** bằng 0.
5. Bước 5: Khi **n = 0**, hàm trả về 1.
6. Bước 6: Quay lại từ đệ quy, kết quả từ **power(2, 1)** đến **power(2, 6)** được tính.
7. Bước 7: Kết quả cuối cùng là :2 * 2 * 2 * 2 * 2 * 2 = 64

Vậy kết quả của lời gọi **power(2, 6)** là 64.

Bài 4: Cho hàm đệ quy giải bài toán Tháp Hà Nội. Hãy giải thích từng bước thực hiện của hàm đệ quy này chuyển 4 đĩa từ cọc A sang cọc B, với trung gian là cọc C.

```
1  def thap_ha_noi(n, A, C, B):
2      if n == 1:
3          print(f"Chuyển đĩa 1 từ cột {A} sang cột {B}")
4      else:
5          thap_ha_noi(n-1, A, B, C)
6          print(f"Chuyển đĩa {n} từ cột {A} sang cột {B}")
7          thap_ha_noi(n-1, C, A, B)
8
9  # Chuyển 4 đĩa từ cọc A sang cọc B, với trung gian là cọc C
10 thap_ha_noi(4, "A", "C", "B")
```

Quy trình thực hiện các bước như sau:

1. Bước 1: Gọi **thap_ha_noi(4, "A", "B", "C")**
2. Bước 2: Vì **n = 4** không bằng 1, nó đi vào phần else của câu lệnh điều kiện.
3. Bước 3: Hàm gọi **thap_ha_noi(3, "A", "C", "B")**
    - Lúc này, **thap_ha_noi(3, "A", "C", "B")** sẽ thực hiện việc di chuyển 3 đĩa từ cột **A** sang cột **C**, với cột **B** là cột trung gian.

4. Bước 4: Trong **thap_ha_noi(3, "A", "C", "B")**, hàm gọi **thap_ha_noi(2, "A", "B", "C")**
    - Lúc này, **thap_ha_noi(2, "A", "B", "C")** sẽ thực hiện việc di chuyển 2 đĩa từ cột **A** sang cột **B**, với cột **C** là cột trung gian.
5. Bước 5: Tiếp tục quá trình này đến khi nó gọi **thap_ha_noi(1, "A", "C", "B")**
    - Trong **thap_ha_noi(1, "A", "C", "B")**, nó sẽ di chuyển 1 đĩa từ cột **A** sang cột **B**.
6. Bước 6: Quay lại từ đệ quy, các đĩa từ cột **A** được di chuyển sang cột **C**, sau đó đĩa cuối cùng từ cột **A** được di chuyển sang cột **B**.
7. Bước 7: Kế tiếp, các đĩa từ cột **C** được di chuyển sang cột **B**, sử dụng cột **A** làm cột trung gian.
8. Bước 8: Quá trình này hoàn thành, và tất cả các đĩa đã được di chuyển từ cột **A** sang cột **B** theo quy tắc của bài toán tháp Hà Nội.

Kết quả là bạn sẽ thấy dãy thông báo in ra màn hình thể hiện từng bước di chuyển của các đĩa. Đầu ra sẽ cho thấy các bước di chuyển để chuyển 4 đĩa từ cột **A** sang cột **B**.

Bài 5: Cho hàm đệ quy giải bài toán cổ vừa gà vừa chó. Hãy giải thích từng bước thực hiện của hàm đệ quy của bài toán này.

```
1  def cho_ga(tong_so_con, tong_so_chan):
2      if tong_so_con == 0 and tong_so_chan == 0:
3          return 0, 0
4      if tong_so_chan % 2 != 0:
5          return -1, -1
6      for cho in range(tong_so_con + 1):
7          ga = tong_so_con - cho
8          if ga * 2 + cho * 4 == tong_so_chan:
9              return cho, ga
10     cho, ga = cho_ga(tong_so_con -1, tong_so_chan -4)
11     if ga != -1:
12         return cho + 1, ga
13     else:
14         return -1, -1
15
16 tong_so_chan = 100
17 tong_so_con = 36
18 so_cho, so_ga = cho_ga(tong_so_con, tong_so_chan)
19 print("Số gà là:", so_ga)
20 print("Số chó là:", so_cho)
```

Quy trình thực hiện các bước như sau:

1. **Bước1:**

   - **tong_so_con = 36** và **tong_so_chan = 100** được truyền vào hàm **cho_ga**.

- Hàm được gọi với các đối số này: **cho, ga = cho_ga(36, 100)**

2. **Bên Trong cho_ga(36, 100)**:
    - Hàm kiểm tra các trường hợp cơ bản:
        - Vì **tong_so_con** không phải là không, và **tong_so_chan** là số chẵn, nó tiếp tục.
    - Sau đó, nó đi vào một vòng lặp để lặp qua các số gà (**cho**) có thể.
    - Đối với mỗi giá trị của **cho**, nó tính số chó (**ga**) như **tong_so_con - cho**.
    - Nó kiểm tra xem tổng số chân có khớp với **tong_so_chan** bằng cách so sánh **ga * 2 + cho * 4** với **tong_so_chan**.
    - Nếu tìm thấy một kết hợp hợp lệ, nó trả về các giá trị của **cho** và **ga**.
    - Nếu không, nó tiếp tục vòng lặp cho đến khi tìm thấy một kết hợp hợp lệ hoặc tiêu tốn tất cả các khả năng.
3. **Tìm Một Kết Hợp Hợp Lệ**:
    - Sau một số lần lặp, nó tìm thấy một kết hợp hợp lệ với **cho = 14** và **ga = 22**.
    - Điều này có nghĩa là có 14 con gà và 22 con chó, và tổng số chân bằng 100.
4. **Trả Kết Quả**:
    - Hàm trả về **cho = 14** và **ga = 22**.
5. **In Ra Kết Quả**:
    - Các giá trị được trả về **so_cho** và **so_ga** được in ra bên ngoài hàm.
    - **print("Số gà là:", so_ga)** in ra **"Số gà là: 14"**.
    - **print("Số chó là:", so_cho)** in ra **"Số chó là: 22"**.