Bài 1

Bước 1: Gọi sum_of_mumbers(7).

Kiểm tra điều kiện cơ sở: n = 7, không thỏa mãn điều kiện cơ sở. Thực hiện câu lệnh trong else: return 7 + sum_of_numbers(6),

Bước 2: Hàm sum_of_mumbers(6) được gọi.

Kiểm tra điều kiện 1 cơ sở: n = 6, không thỏa mãn điều kiện cơ sở.

Bước 3: Hàm sum_of_mumbers(5) được gọi.

Kiểm tra điều kiện cơ sở: n = 5, không thỏa mãn điều kiện cơ sở.

Bước 4: Hàm sum_of_mumbers(4) được gọi.

Kiểm tra điều kiện cơ sở: n = 4, không thỏa mãn điều kiện cơ sở.

Bước 5: Hàm sum_of_mumbers(3) được gọi.

Kiểm tra điều kiện cơ sở: n = 3, không thỏa mãn điều kiện cơ sở.

Bước 6: Hàm sum_of_mumbers(2) được gọi.

Kiểm tra điều kiện cơ sở: n = 2, không thỏa mãn điều kiện cơ sở.

Thực hiện câu lệnh trong else: return 2 + sum_of_mumbers(1).

Bước 7: Hàm sum_of_mumbers(1) được gọi.

Kiểm tra điều kiện cơ sở: n = 1, điều kiện cơ sở được thỏa mãn. Hàm trả về 1.

Các hàm được thực hiện lần lượt theo thứ tự từ dưới lên trên (từ bước 7 lên bước 1)

->sum_of_mumbers(1) trả về 1.

->sum_of_mumbers(2) trả về 2+1 = 3.

->sum_of_mumbers(3) trả về 3+ 3 = 6.

->sum_of_mumbers(4) trả về 4+ 6 = 10.

->sum_of_mumbers(5) trả về 5+ 10 = 15.

->sum_of_mumbers(6) trả về 6+ 15 = 21 ->sum_of_mumbers(7) trả về 7+ 21 = 28.

Vì vậy, kết quả cuối cùng là 28, đây là tổng của các số nguyên dương bé hơn 7.

Bài 2

Bước 1: Gọi hàm fibonacci(8).

Kiểm tra điều kiện cơ sở: n = 8 không thỏa mãn điều kiện cơ sở.

Thực hiện câu lệnh trong else: return fibonacci(7) + fibonacci(6).

Bước 2: Hàm fibonacci(7) và hàm fibonacci(6) được gọi.

Kiểm tra điều kiện cơ sở: n = 7 và n = 6 không thỏa mãn điều kiện cơ sở.

Thực hiện câu lệnh trong else:

return fibonacci(6)+ fibonacci(5)+ fibonacci(5)+fibonacci(4) .

Bước 3: Hàm fibonacci(6) ,fibonacci(5)và hàm fibonacci(4) được gọi.

Kiểm tra điều kiện cơ sở: n = {4,5,6} không thỏa mãn điều kiện cơ sở.

Tiếp tục phân rã cho đến khi gặp điều kiện cơ sở.

...

Bước cuối cùng: Khi n <= 1, hàm trả về n, tức là fibonacci(1) và fibonacci(0) sẽ trả về 1 và 0.

Các hàm được thực hiện lần lượt theo thứ tự từ dưới lên trên, từ các giá trị cơ sở trở lại đến fibonacci(8).

->fibonacci(0) và fibonacci(1) trả về 0 và 1.

->fibonacci(2) trả về 0+1 = 1.

->fibonacci(3) trả về 1+ 1 = 2.

->fibonacci(4) trả về 2+ 1 = 3.

->fibonacci(5) trả về 2+ 3 = 5.

->fibonacci(6) trả về 3+ 5 = 8.

->fibonacci(7) trả về 5+ 8 = 13.

->fibonacci(8) trả về 8+ 13 = 21.

Vì vậy, kết quả cuối cùng là 21, đây là số Fibonacci thứ 8.

Bài 3

Bước 1: Gọi hàm power(2, 6).

- Kiểm tra điều kiện cơ sở: n = 6, không thỏa mãn điều kiện cơ sở.
- Thực hiện câu lệnh trong else: return 2 * power(2, 5).

Bước 2: Hàm power(2, 5) được gọi.

- Kiểm tra điều kiện cơ sở: n = 5, không thỏa mãn điều kiện cơ sở.
- Thực hiện câu lệnh trong else: return 2 * power(2, 4).

Bước 3: Hàm power(2, 4) được gọi.

- Kiểm tra điều kiện cơ sở: n = 4, không thỏa mãn điều kiện cơ sở.
- Thực hiện câu lệnh trong else: return `2 * power(2, 3)`.

Bước 4: Hàm power(2, 3) được gọi.

- Kiểm tra điều kiện cơ sở: n = 3, không thỏa mãn điều kiện cơ sở.
- Thực hiện câu lệnh trong else: return 2 * power(2, 2).

Bước 5: Hàm power(2, 2) được gọi.

- Kiểm tra điều kiện cơ sở: n = 2, không thỏa mãn điều kiện cơ sở.
- Thực hiện câu lệnh trong else: return 2 * power(2, 1).

Bước 6: Hàm power(2, 1) được gọi.

- Kiểm tra điều kiện cơ sở: n = 1, không thỏa mãn điều kiện cơ sở.
- Thực hiện câu lệnh trong else: trả về `2 * power(2, 0).

Bước 7: Hàm power(2, 0) được gọi.

- Kiểm tra điều kiện cơ sở: n = 0, điều kiện cơ sở được thỏa mãn.
- Hàm trả về 1.

Các hàm được thực hiện lần lượt theo thứ tự từ dưới lên trên (từ bước 7 lên bước 1):

->power(2, 0) return 1.
->power(2, 1) return 2 * 1 = 2.
->power(2, 2) return 2 * 2 = 4.
-> power(2, 3) return 2 * 4 = 8.
->power(2, 4) return 2 * 8 = 16.
-> power(2, 5) return 2 * 16 = 32.
-> power(2, 6) return 2 * 32 = 64.

Vì vậy, kết quả cuối cùng là 64, đây là giá trị của 2^6 .

Bài 4

Bước 1: Gọi hàm thap_ha_noi(4, "A", "B", "C").

• Vì n không bằng 1, chúng ta sẽ thực hiện các bước trong else

Bước 2: Gọi hàm thap_ha_noi(3, "A", "C", "B") để chuyển 3 đĩa từ cọc A sang cọc C, với trung gian là cọc B.

• Quy trình này sẽ tiếp tục đệ quy cho đến khi chỉ còn 1 đĩa, sau đó in ra câu lệnh chuyển đĩa đó từ cọc A sang cọc C.

Bước 3: In ra câu lệnh chuyển đĩa lớn nhất (đĩa 4) từ cọc A sang cọc B.

print(f"Chuyển đĩa 4 từ cột A sang cột B")

Bước 4: Gọi hàm thap_ha_noi(3, "C", "B", "A") để chuyển 3 đĩa từ cọc C sang cọc B, với trung gian là cọc A.

• Quy trình này sẽ tiếp tục đệ quy cho đến khi chỉ còn 1 đĩa, sau đó in ra câu lệnh chuyển đĩa đó từ cọc C sang cọc B.

Bước 5: Chuyển đĩa thứ 3 từ cọc C sang cọc B.

print(f"Chuyển đĩa 3 từ cột C sang cột B").

Bước 6: Gọi hàm thap_ha_noi(2, "A", "B", "C") để chuyển 2 đĩa từ cọc A sang cọc B, với trung gian là cọc C.

• Quy trình này sẽ tiếp tục đệ quy cho đến khi chỉ còn 1 đĩa, sau đó in ra câu lệnh chuyển đĩa đó từ cọc A sang cọc B.

Bước 7: Chuyển đĩa thứ 2 từ cọc A sang cọc B.

print(f"Chuyển đĩa 2 từ cột A sang cột B").

Bước 8: Gọi hàm thap_ha_noi(1, "C", "A", "B") để chuyển đĩa cuối cùng từ cọc C sang cọc B, với trung gian là cọc A.

• Đây là bước cuối cùng, và nó sẽ in ra câu lệnh chuyển đĩa cuối cùng từ cọc C sang cọc B.

Bước 9: Chuyển đĩa thứ 1 từ cọc C sang cọc B.

print(f"Chuyển đĩa 1 từ cột C sang cột B").

Chuyển đĩa 1 từ cột A sang cột C

Chuyển đĩa 2 từ cột A sang cột C

Chuyển đĩa 1 từ cột B sang cột A

Chuyển đĩa 3 từ cột A sang cột B

Chuyển đĩa 1 từ cột C sang cột B

Chuyển đĩa 2 từ cột C sang cột B

Chuyển đĩa 1 từ cột A sang cột C

Chuyển đĩa 4 từ cột A sang cột C

Chuyển đĩa 1 từ cột B sang cột A

Chuyển đĩa 2 từ cột B sang cột A

Chuyển đĩa 1 từ cột C sang cột B

Chuyển đĩa 3 từ cột B sang cột C

Chuyển đĩa 1 từ cột A sang cột C

Chuyển đĩa 2 từ cột A sang cột C

Chuyển đĩa 1 từ cột B sang cột A

Quy trình này sẽ lặp lại cho đến khi tất cả các đĩa được chuyển từ cọc A sang cọc B theo đúng quy tắc của bài toán Tháp Hà Nội.

Bài 5

Bước 1: Gọi hàm cho_ga (36, 100).

• Kiểm tra điều kiện cơ sở: tong_so_con không bằng 0 hoặc tong_so_chan không bằng 0, và tong_so_chan chia hết cho 2 và lớn hơn hoặc bằng tong_so_con * 2.

Bước 2: Lặp qua từng số lượng chó có thể (từ 0 đến tong_so_con).

• Tính số lượng gà dựa trên số lượng chó hiện tại: ga = tong_so_con cho

• Kiểm tra nếu tổng số chân của gà và chó bằng tong_so_chan: if ga *2 + cho * 4 == tong_so_chan

Bước 3: Nếu tìm được số lượng gà và chó phù hợp, trả về kết quả.

• Trả về cho, ga

Bước 4: Nếu không tìm được kết quả ở bước hiện tại, gọi đệ quy hàm cho_ga với số lượng con giảm đi 1 và số lượng chân giảm đi 4.

cho, ga = cho_ga (tong_so_con - 1, tong_so_chan - 4)

Bước 5: Kiểm tra kết quả từ lời gọi đệ quy.

• Nếu ga không bằng -1, tức là tìm được kết quả, trả về cho + 1, ga

• Nếu không, trả về -1, -1 để biểu thị không tìm được kết quả phù hợp.

Kết quả cuối cùng sẽ được in ra thông qua print("Số gà là:", so_ga) và print("Số chó là:", so_cho), cho thấy số lượng gà và chó tương ứng với số lượng con và chân đã cho.