PHẠM THỊ THU GIANG_23174600108_CA SÁNG

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))
```

Bước 1: Gọi hàm sum_of_numbers(7).

Kiểm tra điều kiện cơ sở: n = 7, không thoả mãn điều kiện cơ sở.

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

Bước 2: Gọi hàm sum_of_numbers(6).

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

Thực hiện câu lệnh trong else: return 6 + sum_of_numbers(5).

Bước 3: Tiếp tục quy trình tương tự cho sum_of_numbers(5), sum_of_numbers(4), sum_of_numbers(3), sum_of_numbers(2) và sum_of_numbers(1).

Bước 4: Khi đến sum_of_numbers(1), điều kiện cơ sở được thoả mãn, nên hàm trả về 1.

Bước 5: Tiến hành tính toán các kết quả trả về từ các lời gọi hàm đệ quy trên.

Bước 6: Tổng các số từ 1 đến 7 sẽ là: 7 + (6 + (5 + (4 + (3 + (2 + 1))))) = 7 + 6 + 5 + 4 + 3 + 2 + 1.

Bước 7: In ra kết quả, tổng của các số từ 1 đến 7 là 28.

Vậy kết quả cuối cùng được in ra 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.

PHẠM THỊ THU GIANG_23174600108_CA SÁNG

```
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))
```

Bước 1: Gọi hàm fibonacci(8).

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

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

Bước 2: Gọi hàm fibonacci(7).

Kiểm tra điều kiện cơ sở: n = 7, không thoả mãn điều kiện cơ sở.

Thực hiện câu lệnh trong else: return fibonacci(6) + fibonacci(5).

Bước 3: Tiếp tục quy trình tương tự cho fibonacci(6), fibonacci(5), fibonacci(4), fibonacci(3), fibonacci(2) và fibonacci(1).

Bước 4: Khi đến fibonacci(1) và fibonacci(2), điều kiện cơ sở được thoả mãn, nên hàm trả về giá trị của n.

fibonacci(1) trả về 1.

fibonacci(2) trả về 1.

Bước 5: Tiến hành tính toán các kết quả trả về từ các lời gọi hàm đệ quy trên.

fibonacci(3) = fibonacci(2) + fibonacci(1) = 1 + 1 = 2

fibonacci(4) = fibonacci(3) + fibonacci(2) = 2 + 1 = 3

fibonacci(5) = fibonacci(4) + fibonacci(3) = 3 + 2 = 5

fibonacci(6) = fibonacci(5) + fibonacci(4) = 5 + 3 = 8

fibonacci(7) = fibonacci(6) + fibonacci(5) = 8 + 5 = 13

Bước 6: Kết quả cuối cùng của fibonacci(8) là tổng của fibonacci(7) và fibonacci(6), tức là 13 + 8 = 21.

Vậy kết quả cuối cùng được in ra 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))
```

Bước 1: Gọi hàm power(2, 6).

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

Thực hiện câu lệnh trong else: return 2 * power(2, 5).

Bước 2: Gọi hàm power(2, 5).

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

Thực hiện câu lệnh trong else: return 2 * power(2, 4).

Bước 3: Tiếp tục quy trình tương tự cho power(2, 4), power(2, 3), power(2, 2), power(2, 1) và power(2, 0).

Khi đến power(2, 0), điều kiện cơ sở được thoả mãn, nên hàm trả về 1.

Bước 4: Tiến hành tính toán các kết quả trả về từ các lời gọi hàm đệ quy trên.

power(2, 1) = 2 * power(2, 0) = 2 * 1 = 2

power(2, 2) = 2 * power(2, 1) = 2 * 2 = 4

power(2, 3) = 2 * power(2, 2) = 2 * 4 = 8

power(2, 4) = 2 * power(2, 3) = 2 * 8 = 16

power(2, 5) = 2 * power(2, 4) = 2 * 16 = 32

Bước 5: Kết quả cuối cùng của power(2, 6) là 2 * 32 = 64.

Vậy kết quả cuối cùng được in ra 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")
```

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

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

Bước 2: Thực hiện lời gọi đệ quy thap_ha_noi(n-1, A, B, C).

Gọi thap_ha_noi(3, "A", "B", "C").

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

Bước 3: Thực hiện lại lời gọi đệ quy thap_ha_noi(n-1, A, C, B).

Gọi thap_ha_noi(2, "A", "C", "B").

Bước 4: Tiếp tục quá trình tương tự cho thap_ha_noi(2, "A", "C", "B"), thap_ha_noi(1, "A", "B", "C").

Khi đến thap_ha_noi(1, "A", "B", "C"), điều kiện cơ sở được thoả mãn, nên in ra lời giải chuyển đĩa 1 từ cột A sang cột B.

Bước 5: Tiếp tục thực hiện các lời gọi đệ quy khác trong hàm thap_ha_noi(2, "A", "C", "B").

In ra lời giải chuyển đĩa 2 từ cột A sang cột B.

Bước 6: Tiếp tục thực hiện các lời gọi đệ quy khác trong hàm thap_ha_noi(3, "A", "B", "C").

In ra lời giải chuyển đĩa 3 từ cột A sang cột B.

Bước 7: Tiếp tục thực hiện các lời gọi đệ quy khác trong hàm thap_ha_noi(4, "A", "C", "B").

In ra lời giải chuyển đĩa 4 từ cột A sang cột B.

Vậy sau khi thực hiện các bước này, bạn sẽ có kết quả in ra các bước chuyển đĩa từ cột A sang cột B theo quy tắc của bài toán Tháp Hà Nội.

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

```
def cho_ga(tong_so_con, tong_so_chan):
    if tong_so_con == 0 and tong_so_chan == 0:
        return 0, 0
    if tong_so_chan % 2 != 0:
        return -1, -1
    for cho in range(tong_so_con + 1):
        ga = tong_so_con - cho
        if ga * 2 + cho * 4 == tong_so_chan:
            return cho, ga
    cho, ga = cho_ga(tong_so_con -1, tong_so_chan -4)
    if ga != -1:
        return cho + 1, ga
    else:
        return -1, -1

tong_so_chan = 100
tong_so_con = 36
so_cho, so_ga = cho_ga(tong_so_con, tong_so_chan)
print("Số gà là:", so_ga)
print("Số chó là:", so_cho)
```

Bước 1: Gọi hàm cho_ga(36, 100).

Bước 2: Kiểm tra điều kiện cơ sở: tong_so_con = 36 và tong_so_chan = 100.

Không thoả mãn điều kiện cơ sở, tiếp tục vào bước tiếp theo.

Bước 3: Kiểm tra điều kiện tong_so_chan % 2 != 0.

PHẠM THỊ THU GIANG_23174600108_CA SÁNG

Điều kiện này kiểm tra xem tổng số chân có phải là số chẵn hay không. Trong trường hợp không phải số chẵn, không thể tạo ra một phân phối hợp lệ của gà và chó. Hàm sẽ trả về -1, -1.

Bước 4: Duyệt qua tất cả các lựa chọn có thể cho số lượng gà và chó:

Với mỗi lựa chọn số lượng chó (cho) từ 0 đến tong_so_con, ta tính số lượng gà tương ứng (ga) bằng cách trừ số lượng chó từ tong_so_con.

Kiểm tra xem có tổng số chân của gà và chó có bằng tong_so_chan không. Nếu có, trả về số lượng chó và gà tương ứng.

Bước 5: Nếu không tìm thấy một phân phối hợp lệ trong vòng lặp for, tiếp tục tìm kiếm bằng cách giảm số lượng con và số chân đi một đơn vị.

Gọi đệ quy cho_ga(tong_so_con - 1, tong_so_chan - 4).

Nếu tìm được một giải pháp hợp lệ, trả về số lượng chó và gà tương ứng, nếu không, trả về -1, -1.

Bước 6: In ra kết quả: số gà và số chó tương ứng.

Vậy sau khi thực hiện các bước này, bạn sẽ có kết quả in ra số lượng chó và gà tương ứng sao cho tổng số chân và tổng số con được cung cấp.