LÊ DUY QUÝ_17A2HN_23174600069_CA CHIỀU

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

```python
def sum_of_numbers(n):
    if n == 1:
        return 1
    else:
        return n + sum_of_numbers(n-1)
print(sum_of_numbers(7))
```

Bước 1: Gọi hàm `sum_of_numbers(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_numbers(6)` được gọi

Kiểm tra điều kiện cơ sở: `n = 6`, không thỏa mãn điều kiện cơ sở.
Thực hiện câu lệnh trong `else`: `return 6 + sum_of_numbers(5)`.

Bước 3: Hàm `sum_of_numbers(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 5 + sum_of_numbers(4)`.

Bước 4: Hàm `sum_of_numbers(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 4 + sum_of_numbers(3)`.

Bước 5: Hàm `sum_of_numbers(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 3 + sum_of_numbers(2)`

Bước 6: Hàm `sum_of_numbers(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_numbers(1)`.

Bước 7: Hàm `sum_of_numbers(1)` được gọi

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

Kết quả cuối cùng

- `sum_of_numbers(7)` trả về 28.

Vì vậy, khi `print(sum_of_numbers(7))`, kết quả sẽ là 28. Đây là tổng các số từ 1 đến 7.

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.

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

Bước 1: Gọi `fibonacci(8)`:

Không thỏa mãn điều kiện cơ sở (`n > 1`).

Trả về `fibonacci(7) + fibonacci(6)`.

Bước 2: Gọi **`fibonacci(7)`**:

Không thỏa mãn điều kiện cơ sở.

Trả về **`fibonacci(6) + fibonacci(5)`**.

Bước 3: Gọi **`fibonacci(6)`**:

Không thỏa mãn điều kiện cơ sở.

Trả về **`fibonacci(5) + fibonacci(4)`**.

Bước 4: Gọi **`fibonacci(5)`**:

Không thỏa mãn điều kiện cơ sở.

Trả về **`fibonacci(4) + fibonacci(3)`**.

Bước 5: Gọi **`fibonacci(4)`**:

Không thỏa mãn điều kiện cơ sở.

Trả về **`fibonacci(3) + fibonacci(2)`**.

Bước 6: Gọi **`fibonacci(3)`**:

Không thỏa mãn điều kiện cơ sở.

Trả về **`fibonacci(2) + fibonacci(1)`**.

Bước 7: Gọi **`fibonacci(2)`**:

Không thỏa mãn điều kiện cơ sở.

Trả về **`fibonacci(1) + fibonacci(0)`**.

Bước 8: Gọi **`fibonacci(1)`**:

Thỏa mãn điều kiện cơ sở (**`n <= 1`**).

Trả về **`1`**.

Bước 9: Gọi **`fibonacci(0)`**:

Thỏa mãn điều kiện cơ sở (**`n <= 1`**).

Trả về **`0`**.

Tổng hợp các kết quả:

`fibonacci(1) + fibonacci(0) = 1 + 0 = 1` ⇒ `fibonacci(2) = 1`.
`fibonacci(2) + fibonacci(1) = 1 + 1 = 2` ⇒ `fibonacci(3) = 2`.
`fibonacci(3) + fibonacci(2) = 2 + 1 = 3` ⇒ `fibonacci(4) = 3`.
`fibonacci(4) + fibonacci(3) = 3 + 2 = 5` ⇒ `fibonacci(5) = 5`.
`fibonacci(5) + fibonacci(4) = 5 + 3 = 8` ⇒ `fibonacci(6) = 8`.
`fibonacci(6) + fibonacci(5) = 8 + 5 = 13` ⇒ `fibonacci(7) = 13`.
`fibonacci(7) + fibonacci(6) = 13 + 8 = 21` ⇒ `fibonacci(8) = 21`.

Kết quả cuối cùng:

`fibonacci(8)` trả về `21`.

Vì vậy, khi `print(fibonacci(8))`, kết quả sẽ 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

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

Bước 1: Gọi hàm `power(2, 6)`

Kiểm tra điều kiện cơ sở: n=6, không thỏa mãn điều kiện cơ sở ($n \neq 0 n \square = 0$).
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 thỏa mãn điều kiện cơ sở ($n \neq 0 n \square = 0$).
Thực hiện câu lệnh trong `else`: `return 2 * power(2, 4)`.

Bước 3: Gọi hàm `power(2, 4)`

Kiểm tra điều kiện cơ sở: n=4, không thỏa mãn điều kiện cơ sở ($n \neq 0 n \square = 0$).
Thực hiện câu lệnh trong `else`: `return 2 * power(2, 3)`.

Bước 4: Gọi hàm `power(2, 3)`

Kiểm tra điều kiện cơ sở: n=3, không thỏa mãn điều kiện cơ sở ($n \neq 0 n \square = 0$).
Thực hiện câu lệnh trong `else`: `return 2 * power(2, 2)`.

Bước 5: Gọi hàm `power(2, 2)`

Kiểm tra điều kiện cơ sở: n=2, không thỏa mãn điều kiện cơ sở ($n \neq 0 n \square = 0$).
Thực hiện câu lệnh trong `else`: `return 2 * power(2, 1)`.

Bước 6: Gọi hàm `power(2, 1)`

Kiểm tra điều kiện cơ sở: n=1, không thỏa mãn điều kiện cơ sở ($n \neq 0 n \square = 0$).

Thực hiện câu lệnh trong `else`: `return 2 * power(2, 0)`.

Bước 7: Gọi hàm `power(2, 0)`

Kiểm tra điều kiện cơ sở: n=0, thỏa mãn điều kiện cơ sở (n=0n=0).
Hàm trả về: `1`

Kết quả cuối cùng

`power(2, 6)` trả về 64.

Vậy, khi `print(power(2, 6))`, kết quả sẽ 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.

```
def tha_ha_noi(n,A, C, B):
    if n == 1:
        print(f"Chuyển đĩa 1 từ cột {A} sang cột {B}")
    else:
        tha_ha_noi(n-1, A, B, C)
        print(f"Chuyển đĩa {n}từ cột {A}sang cột {B}")
        tha_ha_noi(n-1,C, A, B)
#Chuyển 4 đĩa từ cọc A sang cọc B,với trung gian là cọc C
tha_ha_noi(4,"A","C","B")
```

Bước 1: Gọi `tha_ha_noi(4, "A", "C", "B")`

Không thỏa mãn điều kiện cơ sở (n≠1n□=1).

Chia thành hai bước nhỏ hơn:

Bước 1.1: `tha_ha_noi(3, "A", "B", "C")`

Bước 1.2: In ra "Chuyển đĩa 4 từ cột A sang cột B"

Bước 1.3: `tha_ha_noi(3, "C", "A", "B")`

Bước 1.1: Gọi `tha_ha_noi(3, "A", "B", "C")`

Không thỏa mãn điều kiện cơ sở (n≠1n□=1).

Chia thành hai bước nhỏ hơn:

Bước 1.1.1: `tha_ha_noi(2, "A", "C", "B")`

Bước 1.1.2: In ra "Chuyển đĩa 3 từ cột A sang cột B"

Bước 1.1.3: `tha_ha_noi(2, "C", "B", "A")`

Tiếp tục quá trình đệ quy cho đến khi gặp điều kiện cơ sở:

Bước 1.1.1: Gọi `tha_ha_noi(2, "A", "C", "B")`

Không thỏa mãn điều kiện cơ sở ($n \neq 1 n \square = 1$).

Chia thành hai bước nhỏ hơn:

Bước 1.1.1.1: `tha_ha_noi(1, "A", "B", "C")`

Bước 1.1.1.2: In ra "Chuyển đĩa 2 từ cột A sang cột B"

Bước 1.1.1.3: `tha_ha_noi(1, "B", "A", "C")`Bước 1.1.1.1: Gọi `tha_ha_noi(1, "A", "B", "C")`

Thỏa mãn điều kiện cơ sở ($n=1$).

In ra "Chuyển đĩa 1 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.

```python
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 lag:",so_ga)
```

```
print("Số chó là:",so_cho)
```

Bước 1: Gọi `cho_ga(36, 100)`

Không thỏa mãn điều kiện cơ sở (cả `tong_so_con` và `tong_so_chan` không đều bằng 0).

Tiếp tục thực hiện.

Bước 2: Duyệt qua các giá trị có thể của số chó

Với `cho` từ 0 đến `tong_so_con`, tính số gà tương ứng.
Nếu tổng số chân của chó và gà đúng bằng `tong_so_chan`, trả về số chó và số gà.

Bước 3: Gọi đệ quy `cho_ga(tong_so_con - 1, tong_so_chan - 4)`

Giảm số con và số chân đi 4.

Nếu kết quả trả về không phải là `-1`, thì tăng số chó lên 1 và giữ nguyên số gà.

Kết quả cuối cùng:

Số chó là 14 và số gà là 22.