Given Problem:

Consider the thirsty crow story where a thirsty crow identifies a jug with little water. It puts pebbles into the water to raise the level of water and drinks it. Assume that the initial reading of the jug is 'm1' ml and the crow can drink water if the level of water has come to 'm2' ml. There are two categories of pebbles small and big in the field. Small pebble can raise the level of water by 'x' ml and big pebble can raise the level of water by 'y' ml. There are 'n' small pebbles. Crow prefers to place small pebbles in jug and then only takes big pebbles. Write an PAC, algorithm and draw a flow chart using YED tool to determine the number of pebbles required to raise the water to 'm2' level. For example, if value of 'm1', 'm2','x','y' and 'n' are 54, 300, 10, 20, 10 respectively then the number of pebbles required is 13.

Problem Analysis Chart:

Data	Processing	Output	Solution Alternatives
Initial volume of water, Final Volume of water, Volume of large and small pebbles, Number of small pebbles available as m1, m2, y, x, n respectively	Check if m2 <= m1? • If yes, number of pebbles needed is zero • If no, check if (m2-m1) <= (n*x)? ▪ If yes, number of pebbles needed = round up((m2-m1) / x) ▪ If no, Number of large pebbles used = round up(((m2 –m1)-x*n)//y) Total number of pebbles used = Number of small pebbles used + Number of large pebbles used	Display Total number of pebbles used	1. Define initial volume, final volume, volume of pebbles as constant. 2. Define number of small pebbles as constant. 3. Define all the values as user input.

Algorithm:

- ❖ Get initial and final volume, number of small pebbles available, volume of each pebbles: m1, m2, n, x, y,
- ❖ Check if m2 <= m1?
  - If yes, number of pebbles needed is zero
  - If no, check if (m2-m1) <= (n*x)?
    - If yes, number of pebbles needed = round up((m2-m1) / x)
    - If no, Number of large pebbles used = round up(((m2 –m1)-x*n)//y)
      Total number of pebbles used = Number of small pebbles used + Number of large pebbles used
- ❖ Display the number of needed pebbles.

```mermaid
flowchart TD
    Start([Start])
    Get[Get m1, m2, n, x, y]
    C1{Check if m2<= m1?}
    Zero[Display the number of needed pebbles is Zero]
    C2{Check if (m2-m1)<= (n*x)?}
    T1[Number of pebbles T= math.ceil((m2-m1)/x)]
    Calc[Calculate L= math.ceil(((m2 –m1)-x*n)/y)\nTotal number of pebbles(T) = L+n]
    Disp1[Display number of pebbles T]
    Disp2[Display total number of pebbles T]
    End([End])

    Start --> Get
    Get --> C1
    C1 -->|yes| Zero
    C1 -->|no| C2
    C2 -->|yes| T1
    C2 -->|no| Calc
    T1 --> Disp1
    Calc --> Disp2
    Zero --> End
    Disp2 --> End
    Disp1 --> End
```

Start

Get m1, m2, n, x, y

Check if m2<= m1?
- yes → **Display the number of needed pebbles is Zero**
- no ↓

Check if (m2-m1)<= (n*x)?
- yes → **Number of pebbles T= math.ceil((m2-m1)/x)** → **Display number of pebbles T**
- no ↓

Calculate L= math.ceil(((m2 –m1)-x*n)/y)
Total number of pebbles(T) = L+n

Display total number of pebbles T

End