Ideology to proceed:

- For each circle, check if Abhimanyu can battle or needs to use skips/recharges.
- Update enemy powers if regeneration is needed.
- Return `"Success"` if all circles are processed successfully, otherwise `"Failure"`.

Algorithm

1. **Initialise Variables:**
   - **User Input:**
     - `p`: Initial power of Abhimanyu.
     - `a`: Total skips available.
     - `b`: Total recharge power Abhimanyu can do.
     - `enemy_powers[]`: Array of powers for 11 enemies.
   - **Variables:**
     - Abhimanyu's current power: `current_power = p`
     - Skips remaining: `skips = a`
     - Power recharge left: `recharge_left = b`
     - to store regenerated enemy powers for circles 3 and 7: `regenerate_enemy[]`
2. **Regenerated Enemy Powers will be half of their initial power:**
   - Set `regenerate_enemy[2]` to `enemy_powers[2] / 2`. // For enemy 3
   - Set `regenerate_enemy[6]` to `enemy_powers[6] / 2`. // For enemy 7
   -
3. **Loop from circle `i` from 0 to 10:**
   - `power = enemy_powers[i]`
   - **Check if Abhimanyu can battle:**
     - If `current_power >= power`
       - `current_power -= power`
       - **Regenerated Enemy to attack from behind in iterative next circle:**
         - If `i == 2` or `i == 6` // circle 3 or 7
           - If `i + 1 < 11`
             - Set `enemy_powers[i+1]` to `regenerate_enemy[i]`
     - Else if `current_power` is less than `power`
       - **Skip battle if skips are remaining:**
       - If `skips > 0`:
         - `skips = skips - 1`
         - Continue to the next iteration of the loop.
       - **Else** if `recharge_left > 0`:
         - `recharge_left = recharge_left - 1`
         - Set `current_power = p` // Initial power
         - **Recheck Battle After Recharge // iterative part to check battle can create separate function while writing program to be invoked here.**
           - If `current_power >= power`
             - `current_power -= power`
             - **Regenerated Enemy:**
             - If `i == 2` or `i == 6` // circle 3 or 7
               - If `i + 1 < 11`
               - Set `enemy_powers[i+1]` to `regenerate_enemy[i]`
           - Else (Recharge failed to meet enemy power):
           - Return `"Failure"`
       - Else (No skips or recharges left):
       - Return `"Failure"`.

4. Return `"Success"` // end of function

Test Cases

- **Test Case 1:**
  - [ ] Initial power (p) = 30
  - [ ] Skips (a) = 0
  - [ ] Recharges available (b) = 2
  - [ ] enemy_powers = [25, 15, 25, 20, 10, 30, 18, 20, 9, 33, 12]

    -> Failure: Abhimanyu cannot cross the Chakravyuha due to insufficient power and no skips or recharges left.

- **Test Case 2:**
  - [ ] Initial power (p) = 50
  - [ ] Skips (a) = 6
  - [ ] Recharges available (b) = 2
  - [ ] enemy_powers = [25, 15, 25, 20, 10, 30, 18, 20, 9, 33, 12]

    -> Success: Abhimanyu can cross the Chakravyuha.