Q3 Cat and Mouse Game

a) Stationary Distributions

Stationary Distribution of Cat Chain :

Let $\pi_1 = P(X_t = \text{Room } 1)$ and $\pi_2 = P(X_t = \text{Room } 2)$, then:

$\pi_1 = P(X_{t-1} = \text{Room } 1) \cdot P(X_t = \text{Room } 1 \mid X_{t-1} = \text{Room } 1) + P(X_{t-1} = \text{Room } 2) \cdot P(X_t = \text{Room } 1 \mid X_{t-1} = \text{Room } 2)$

$\pi_2 = P(X_{t-1} = \text{Room } 1) \cdot P(X_t = \text{Room } 2 \mid X_{t-1} = \text{Room } 1) + P(X_{t-1} = \text{Room } 2) \cdot P(X_t = \text{Room } 2 \mid X_{t-1} = \text{Room } 2)$

Using the transition probabilities:

- $P(X_t = \text{Room } 1 \mid X_{t-1} = \text{Room } 1) = 0.2$
- $P(X_t = \text{Room } 1 \mid X_{t-1} = \text{Room } 2) = 0.8$
- $P(X_t = \text{Room } 2 \mid X_{t-1} = \text{Room } 1) = 0.8$
- $P(X_t = \text{Room } 2 \mid X_{t-1} = \text{Room } 2) = 0.2$

With $\pi_1 + \pi_2 = 1$, we get:

$\pi_1 = \pi_1 \cdot 0.2 + \pi_2 \cdot 0.8$

$\pi_2 = \pi_1 \cdot 0.8 + \pi_2 \cdot 0.2$

Final Stationary Probabilities:

- $\pi_1 = 0.5$
- $\pi_2 = 0.5$

Stationary Distribution of Mouse Chain :

Let $\mu_1 = P(Y_t = \text{Room } 1)$ and $\mu_2 = P(Y_t = \text{Room } 2)$, then:

$\mu_1 = P(Y_{t-1} = \text{Room } 1) \cdot P(Y_t = \text{Room } 1 \mid Y_{t-1} = \text{Room } 1) + P(Y_{t-1} = \text{Room } 2) \cdot P(Y_t = \text{Room } 1 \mid Y_{t-1} = \text{Room } 2)$

$\mu_2 = P(Y_{t-1} = \text{Room } 1) \cdot P(Y_t = \text{Room } 2 \mid Y_{t-1} = \text{Room } 1) + P(Y_{t-1} = \text{Room } 2) \cdot P(Y_t = \text{Room } 2 \mid Y_{t-1} = \text{Room } 2)$

Using the transition probabilities:

- $P(Y_t = \text{Room } 1 \mid Y_{t-1} = \text{Room } 1) = 0.7$
- $P(Y_t = \text{Room } 1 \mid Y_{t-1} = \text{Room } 2) = 0.6$
- $P(Y_t = \text{Room } 2 \mid Y_{t-1} = \text{Room } 1) = 0.3$
- $P(Y_t = \text{Room } 2 \mid Y_{t-1} = \text{Room } 2) = 0.4$

With $\mu_1 + \mu_2 = 1$, we get:

$\mu_1 = \mu_1 \cdot 0.7 + \mu_2 \cdot 0.6$

$\mu_2 = \mu_1 \cdot 0.3 + \mu_2 \cdot 0.4$

Final Stationary Probabilities:

- $\mu_1 = \frac{2}{3}$
- $\mu_2 = \frac{1}{3}$

b) Is it a markov chain?

Yes, the sequence $Z_0, Z_1, Z_2, \ldots$ **is** a Markov chain.

In this case:

- The cat and mouse move independently.
- Each next position depends only on the **current position**, not on the history.
- Hence, the joint transition probability:

$P(Z_{n+1} \mid Z_n) = P(\text{Cat}_{n+1} \mid \text{Cat}_n) \cdot P(\text{Mouse}_{n+1} \mid \text{Mouse}_n)$
depends **only on** Z_n.

Therefore, $\{Z_n\}$ satisfies the Markov property and is a **valid Markov chain** with 4 states.

Q5 Stock Price Model

a) Is the Stock Price Recurrent?

No, it is not recurrent.

- The state space is **infinite** in both directions (prices can go very high in theory)

- Because of the average positive drift, the stock tends to go up over time

b) Does the stationary distribution of the stock price exist?

There is **no** stationary distribution because the stock price has a positive drift
and tends to increase indefinitely over time. This means the process does not settle
into a long-term equilibrium and does not return to states often enough.

c) Does the stock price reach 130 INR befor 1:00 PM?

We will use dynamic programming with mpmath for high precision.

In [1]:
```
from mpmath import mp, mpf

# Set decimal precision
mp.dps = 10

tick_size = 0.01
start_price = 120.00
target_price = 130.00
total_steps = 3*60*60//5

prob_up = mpf('0.1')
prob_same = mpf('0.85')
```

```python
prob_down = mpf('0.05')

# Create DP table p[t][s] where: t is the step number and s is the price index
p = [[mpf('0') for _ in range(2 * total_steps + 1)] for _ in range(total_steps + 1)]
# Initialize starting price index
p[0][total_steps] = mpf('1.0')

# Fill DP table
for step in range(1, total_steps + 1):
    for price in range(2 * total_steps + 1):
        if p[step - 1][price] != mpf('0'):
            # Non zero probability from previous step will propagate
            # to the next step based on the probabilities
            p[step][price - 1] += p[step - 1][price] * prob_down
            p[step][price]     += p[step - 1][price] * prob_same
            p[step][price + 1] += p[step - 1][price] * prob_up

# Sum up final probabilities for prices >= target
index_shift = int((target_price - start_price) / tick_size)
final_probs = p[total_steps][total_steps + index_shift:]
prob_hit = sum(final_probs)
print(f"Estimated probability using high-precision DP: {prob_hit}")
```

```
Estimated probability using high-precision DP: 1.069032254e-432
```

We can also use log-space probabilities for precision, but that was found to be more computationaly expensive than mpmath.