Horror Film Night

1. Problem restatement

We're given two sets of days:

- `E`: days Emma likes the movie.
- `M`: days Marcos likes the movie.

Each day can be:

- liked by **both** (good for both → watchable anytime)
- liked by **only Emma**
- liked by **only Marcos**
- liked by **neither** (they'll skip automatically)

Rule:

They can't watch **two consecutive movies disliked by the same person.**

That is, if one of them dislikes a movie, the next watched one must not be disliked by that same person again.

Goal:

Find the **maximum number of movies** they can watch under that rule.

---

2. Problem decomposition

We'll sort all days where at least one person likes the movie (union of `E` and `M`).

⟶ Keep an array of states [0, 2, 1, 2, 1, 1, 0 ...] where

Code	Meaning	Who dislikes the movie	Who likes it
`0`	both like	no one	both
`1`	only Marcos likes	**Emma** dislikes	Marcos
`2`	only Emma likes	**Marcos** dislikes	Emma

So `dislikes[i]` ∈ {0, 1, 2}, where:

- `0` → no restriction for next day,
- `1` → Emma disliked this,
- `2` → Marcos disliked this.

Dimension	Symbol	Meaning
Row	`i`	The **current movie day** we're considering (`dislikes[i]` tells us who dislikes *this* movie)
Column	`p`	Who disliked the **previous watched movie, i.e., the emotional state**

---

3. DP subproblem definition

Let:

$dp[i][p] =$ maximum movies they can watch from day i onward, given the previous movie had dislike type p

DP is n * 3 table

where p∈{0,1,2}.

- p=0: previous movie was liked by both.
- p=1: previous movie was disliked by Emma.
- p=2: previous movie was disliked by Marcos.

DP column (state)	Meaning	Restriction for next movie
`0`	previous movie liked by **both**	no restriction
`1`	previous movie disliked by **Emma**	next movie **cannot** also be disliked by Emma
`2`	previous movie disliked by **Marcos**	next movie **cannot** also be disliked by Marcos

---

4. Recurrence relation

- **At the very end (`i = n`),**

  there *is no future* — no more movies to watch.

  So for any previous state `p`,

  ```
  dp[n][p] = 0
  ```

  (base case).

- **Then for movie `i = n - 1`**,

  we can now *look ahead* to `dp[i + 1][*]`,

  which already tells us "if we were at the next movie, how many could we still watch."

- So at each step, we say:

  > "If I'm at movie i, and I know all possible outcomes starting from i+1,
  >
  > what's the best I can do now depending on who disliked the last one?"

- This repeats backward until we reach `i = 0` —

  at that point, we've reconstructed the *optimal past decisions* starting from the beginning.

At each day `i` and previous dislike state `p`:

- **Option 1:** Skip the movie → `skip = dp[i+1][p]`
- **Option 2:** Watch the movie → only allowed if it doesn't violate fairness:

  $$\text{if } dislikes[i] = 0 \text{ or } dislikes[i] \neq p$$

  This means either both of them liked the previous movie, or the person who dislikes the current movie did not dislike the last movie.

  then:

  $$watch = 1 + dp[i+1][dislikes[i]]$$

At any movie `i`, if the *current movie* is disliked by `dislikes[i]`, and you decide to **watch** it, then **that movie becomes the new "last watched" movie** —

so the *next step* (movie `i+1`) must start under the assumption:

$$\text{previous disliked person} = \text{dislikes[i]}$$

That's why we look up:

$$dp[i+1][dislikes[i]]$$

Combine both:

$$dp[i][p] = \max(\text{skip}, \text{watch})$$

$$dp(i,p) = \begin{cases} 0, & \text{if } i = n; \\ \max\left(dp(i+1,p),\ 1 + dp(i+1, dislikes[i])\right), & \text{if } dislikes[i] = 0 \text{ or } dislikes[i] \neq p; \\ dp(i+1,p), & \text{otherwise.} \end{cases}$$

---

5. Base case

At the end of the days:

$$dp[n][p] = 0 \quad \forall p$$

because no more movies can be watched.

---

6. Evaluation order

Bottom-up — compute backwards from the last day `n-1` down to day 0,

since each state depends only on `dp[i+1][*]`.

The DP table has:

- `n+1` rows (for each day),
- `3` columns (for possible previous dislike states).

At the end, find `dp[0][0]`, which is movie from the first movie onward (all) given they both like the movie (because there is no previous restriction for first movie, so both works)

---

7. Complexity

Metric	Complexity
Time	O(3n)=O(n)
Space	O(3n)=O(n)

Example

Let:

```
E = [1, 3]
M = [2, 3]
```

So:

- Emma likes movies on days 1 and 3.
- Marcos likes movies on days 2 and 3.
- Sorted union → `[1, 2, 3]`
  → 3 movies total (`n = 3`)

Now build `dislikes` array:

Day	In E?	In M?	dislikes[i]	Meaning
1	✅	❌	2	only Emma likes → Marcos dislikes
2	❌	✅	1	only Marcos likes → Emma dislikes
3	✅	✅	0	both like

So:

```
dislikes = [2, 1, 0]
```

DP definition

`dp[i][p]` = max number of movies watchable **from movie i onward**,

given the **previous dislike type = p**.

(`p = 0` = both liked, `p = 1` = Emma disliked, `p = 2` = Marcos disliked)

Initialization (base case)

At the end of the list (`i = 3` → after last movie):

```
dp[3][0] = dp[3][1] = dp[3][2] = 0
```

i=3	prev=0	prev=1	prev=2
dp	0	0	0

Movie 3 (i=2, dislikes[2] = 0 → both like)

If both like, they can **always watch** regardless of previous dislike type.

$$dp[2][p] = \max(dp[3][p],\ 1 + dp[3][0]) = 1 + 0 = 1$$

i=2	prev=0	prev=1	prev=2
dp	1	1	1

Movie 2 (i=1, dislikes[1] = 1 → Emma dislikes)

Can watch if:

- previous dislike type ≠ 1, or

- current = 0 (both like, not the case here)

So allowed when `prev` ∈ {0, 2}.

Compute:

$$dp[1][p] = \max(\text{skip} = dp[2][p],\ \text{watch} = 1 + dp[2][1]) \text{ if allowed}$$

Otherwise, skip.

prev	Allowed?	skip=dp[2][p]	watch=1+dp[2][1]	dp[1][p]
0	✅	1	1+1=2	**2**
1	❌	1	—	**1**
2	✅	1	1+1=2	**2**

i=1	prev=0	prev=1	prev=2
dp	2	1	2

Movie 1 (i=0, dislikes[0] = 2 → Marcos dislikes)

Allowed if previous dislike ≠ 2.

So allowed when `prev` ∈ {0, 1}.

Compute:

$$dp[0][p] = \max(dp[1][p], 1 + dp[1][2]) \text{ if allowed}$$

prev	Allowed?	skip=dp[1][p]	watch=1+dp[1][2]	dp[0][p]
0	✅	2	1+2=3	**3**
1	✅	1	1+2=3	**3**
2	❌	2	—	**2**

i=0	prev=0	prev=1	prev=2
dp	**3**	**3**	**2**

Final result

We start with "no previous dislike," meaning `prev = 0`:

$$\boxed{dp[0][0] = 3}$$

They can watch all three movies:

Day 1 (Marcos dislikes) → Day 2 (Emma dislikes) → Day 3 (both like).

Full DP table summary

i (movie index)	dislikes[i]	dp[i][0]	dp[i][1]	dp[i][2]
0	2 (Marcos dislikes)	**3**	**3**	**2**
1	1 (Emma dislikes)	2	1	2
2	0 (both like)	1	1	1
3	— end —	0	0	0

Interpretation

- Each row = state after processing that movie and all future ones.
- Column index (`prev`) = who disliked the last watched movie.
- Values increase as you move up the table (earlier decisions have more choices).
- Top-left cell `dp[0][0]` = final answer = **3**.