Knock-Out Tournament Bracket Manager (Binary Tree, not BST)

Context. You will model a single-elimination tournament as a **binary tree** (not a BST). **Leaves** represent initial players; **internal nodes** represent matches that produce winners. The tree should be *complete* for the given number of participants

Task

Build a small bracket manager that can:

1. build the tournament tree from a list of player names,
2. record match results with validation and correct propagation,
3. answer structure queries (paths, potential meetings), and
4. print the bracket clearly after each round.

Inputs

- List of N distinct player names (strings).
- If N is not a power of two, **pad to the next power of two with BYE** to form a complete tree.

Core Requirements (implement all)

1. **Build bracket** from the player list (left-to-right leaf fill). Assign a unique matchId to every internal node (your format, but consistent).
2. **Record a result**: recordResult(matchId, winnerName)
   - Only accept if both child participants are known (leaf.name or child's winner).
   - If one participant is BYE, auto-advance the other (define and enforce your rule).
   - Reject re-entering a decided match (unless you implement an explicit "edit" mode; if you do, document it).
3. **Path query**: pathToFinal(player) → return the sequence of matchIds the player would traverse from their first match up to the final. State whether you (a) stop at the first loss or (b) always return the theoretical ladder—be consistent.
4. **Meeting query (LCA)**: wouldMeet(p1, p2) → return the matchId **and** the **round number** where they would meet *if both keep winning*. (You must compute this from the tree structure; do not hardcode.)
5. **Print bracket (level-order)** grouped by rounds. Each internal node line must show: matchId, left participant, right participant, winner (or ?). Leaves must show player names (or BYE).

Example of Scenario (must include in your report)

Use this exact 8-player list (no padding needed):
Anna, Ben, Chou, Dara, Ean, Faye, Gita, Hout

1. **After Round 1**: record four leaf-level match results of your choice; print the bracket.

2. **After Round 2**: record the two semifinal results; print the bracket.
3. **After Final**: record the champion; print the bracket.
4. Show **two** would Meet queries (e.g., Chou vs Hout, Anna vs Dara) with both the matchId and round number.
5. Show **two** pathToFinal queries (e.g., Anna, Faye) with your chosen “stop or theoretical” policy applied.

What to Submit

1. **One PDF** with:
   - Names/IDs, chosen representation (array or pointers) + 1–2 sentence rationale.
   - Screenshots or pasted console **outputs** for all required demo steps.
   - 5–8 sentences total explaining your validation rules and propagation logic.
2. **Source files** (clean, compilable) screenshot and put it in report.