ReckonQuest — Flashcards

A Spark & Anvil printable flashcard deck. Quiz yourself on the ReckonQuest cast's curriculum — one card at a time.

How to use these cards

Fold each sheet down the middle line so the answers are hidden. Read the **question** on the left, say your answer out loud, then **unfold** to check the right side. Testing yourself — then checking — is one of the most powerful ways to remember. Get one wrong? That's the best kind of practice: try it again tomorrow.

Want real flashcards? Cut along the fold line and the rows into cards — question on one side, answer on the other.

---

1. systematic guess-and-check → 2-variable systems (the chickens-and-cows riddle)

✂ Question (fold →)	Answer
Tad says "try a smaller case first." For 2 heads and 6 legs, how many chickens and cows?	**1 chicken and 1 cow** — 1 chicken (2 legs) + 1 cow (4 legs) = 6 legs, 2 heads. A small case makes the structure easy to see before scaling up.
How many legs do 4 chickens have in total?	**8** — 4 chickens x 2 legs = 8 legs.
Swapping ONE chicken for ONE cow changes the total leg count by:	**+2 legs (a cow has 2 more legs than a chicken)** — A cow (4 legs) replaces a chicken (2 legs), adding 4 - 2 = 2 legs. This is the key to the shortcut.
A pen holds 5 chickens and 2 cows. How many heads are there?	**7** — 5 chickens + 2 cows = 7 animals = 7 heads.
30 heads, 100 legs, 20 cows. How many chickens?	**10** — 30 animals - 20 cows = 10 chickens. Check: 10 x 2 + 20 x 4 = 20 + 80 = 100 legs. ✓
Lettie says: "Let c = the number of chickens and w = the number of cows." For a riddle with 10 heads, which equation counts the HEADS?	**c + w = 10** — Each animal is 1 head, so chickens + cows = total heads: c + w = 10.
CHECK the answer: 6 chickens and 4 cows — how many legs is that?	**28** — 6 chickens x 2 = 12, 4 cows x 4 = 16; 12 + 16 = 28 legs. It matches — so the answer is confirmed (Vetta's check).
A pen holds 5 chickens and 2 cows. How many legs are there?	**18** — 5 chickens x 2 = 10 legs, 2 cows x 4 = 8 legs; 10 + 8 = 18 legs.
A cow has how many legs?	**4** — A cow has 4 legs.
12 heads and 40 legs. How many chickens and cows?	**4 chickens and 8 cows** — All-chickens = 24 legs; 40 - 24 = 16 extra; 16 / 2 = 8 cows; 12 - 8 = 4 chickens. Check: 4x2 + 8x4 = 8 + 32 = 40. ✓
So for 10 heads and 28 legs, how many chickens and cows are there?	**6 chickens and 4 cows** — 4 cows + (10 - 4) = 6 chickens. Check next!
You guessed all 10 chickens = 20 legs, but the riddle says 28 legs. You need 8 MORE legs. Each cow-swap adds 2 legs. How many cows are there?	**4** — You need 8 more legs; each cow adds 2, so 8 / 2 = 4 cows.
"Guess and check" means:	**make a smart guess, check it against the riddle, then adjust** — Systematic guess-and-check: make an educated guess, check it, and use what you learn to make a better next guess.
The chickens-and-cows riddle is really an example of a:	**system of two equations with two unknowns** — Two equations (heads: c + w = H, legs: 2c + 4w = L) with two unknowns (c, w) — a system. Guess-and-check is the bridge to solving it with algebra.
Why is "guess all chickens first" a smart STARTING guess?	**it gives the fewest legs, a clear baseline to add cows from** — All-chickens gives the minimum legs; from there you only ADD legs (2 per cow), so the gap tells you exactly how many cows to swap

	in.
There are 10 heads. If you GUESS they are all chickens, how many legs would that be?	**20** — All-chickens is a great starting guess: 10 x 2 = 20 legs.
Solve it: 7 heads and 20 legs. How many chickens and cows?	**4 chickens and 3 cows** — All-chickens = 14 legs; 20 - 14 = 6 extra; 6 / 2 = 3 cows; 7 - 3 = 4 chickens. Check: 4x2 + 3x4 = 8 + 12 = 20. ✓
Those 40 extra legs come from cows (each swap adds 2 legs). How many cows?	**20** — 40 / 2 = 20 cows.
With c = chickens and w = cows, which equation counts the LEGS?	**2c + 4w = total legs** — Chicken legs (2 each) + cow legs (4 each) = total: 2c + 4w = total legs.
How many legs do 3 cows have in total?	**12** — 3 cows x 4 legs = 12 legs.
Each animal — chicken OR cow — has how many heads?	**1** — Every animal has exactly 1 head, so the number of heads equals the total number of animals.
30 heads, 100 legs. Your all-chickens guess gave 60 legs. How many MORE legs are needed?	**40** — 100 - 60 = 40 more legs are needed.
A chicken has how many legs?	**2** — A chicken has 2 legs.
If a guess gives you TOO MANY legs, what should you do?	**swap some cows back to chickens (that removes legs)** — Too many legs means too many cows — swap cows back to chickens (each swap removes 2 legs) until it matches.
The riddle: 30 heads and 100 legs. If you GUESS all chickens, how many legs is that?	**60** — All-chickens guess: 30 x 2 = 60 legs.

2. name the unknowns and write the relationships (let x = ...)

✂ Question (fold →)	Answer
"A number and its double add to 18" becomes:	**x + 2x = 18** — 'A number (x) and its double (2x) add to 18' is x + 2x = 18.
Solve x + 2x = 18. What is x?	**6** — x + 2x = 3x = 18, so x = 6.
"5 less than a number" is written as:	**x - 5** — '5 less than x' takes 5 away from x: x - 5 (order matters — not 5 - x).
The key difference between '5 less than x' and '5 minus x' is:	**the order: x - 5 versus 5 - x** — '5 less than x' = x - 5; '5 minus x' = 5 - x. Word order changes the expression.
Naming BOTH unknowns in a two-thing problem is smart because it lets you:	**write a relationship connecting them** — Naming both unknowns lets you write the equation that ties them together — the heart of setting up a system.
From q16, solving x + (x + 4) = 20 gives the smaller number:	**8** — x + (x + 4) = 2x + 4 = 20, so 2x = 16 and x = 8.
"3 times a number, decreased by 5, is 16" becomes:	**3x - 5 = 16** — '3 times a number' is 3x; 'decreased by 5' subtracts 5; 'is 16' sets it equal: 3x - 5 = 16.
Which phrase needs PARENTHESES: "twice the sum of a number and 3"?	**2(x + 3)** — 'The sum of a number and 3' is (x + 3); 'twice' that whole thing is 2(x + 3) — the parentheses keep the sum together.
"7 more than twice a number" is:	**2x + 7** — 'Twice a number' is 2x; '7 more than' adds 7: 2x + 7.
"The number of legs is twice the number of heads" becomes:	**legs = 2 x heads** — 'Legs is twice heads' translates directly to legs = 2 x heads.
"A number equals another number plus 4; together they are 20." Using x for the smaller, the second number is:	**x + 4** — The second number is 4 more than the smaller x, so it is x + 4 (and x + (x+4) = 20).
"Twice a number" is written as:	**2x** — 'Twice a number' means 2 times x = 2x.
Lettie's motto is best stated as:	**"name it, then write what the words say"** — Lettie names the unknown and writes exactly what the words say as an equation — structure, not key-word tricks.
"A number increased by 7" is written as:	**x + 7** — 'Increased by 7' means add 7 to the number: x + 7.
"Half of a number is 12" becomes:	**x/2 = 12** — 'Half of x is 12' is x/2 = 12 (so x = 24).
"The sum of two numbers is 15" with one number x means the other is:	**15 - x** — If the two add to 15 and one is x, the other is 15 - x — a neat way to use ONE letter for a two-number problem.
Solve 3x - 5 = 16. x = ?	**7** — 3x - 5 = 16 -> 3x = 21 -> x = 7.
Lettie writes 'let c = number of chickens.' The number of chicken LEGS is then:	**2c** — With c chickens at 2 legs each, chicken legs = 2c.
Setting up the equation correctly matters because:	**a wrong set-up leads to the wrong answer no matter how well you solve it** — Solving perfectly can't rescue a wrong set-up — the equation must faithfully match the problem's relationships.

If x is the number of apples and there are 3 more oranges than apples, the number of oranges is:	**x + 3** — '3 more oranges than apples' means oranges = x + 3.
The first step Lettie takes on any word problem is to:	**name the unknown with a letter (let x = ...)** — Naming the unknown (let x = the thing you don't know) is the first move — it makes the relationships writable.
If a rectangle's length is 3 more than its width w, its length is:	**w + 3** — Length is 3 more than width w, so length = w + 3.
"The product of a number and 4" is:	**4x** — 'Product of a number and 4' means multiply: 4x.
Why give the unknown a LETTER instead of just guessing a number?	**a letter lets you write the relationships exactly, then solve** — A letter holds the unknown so you can write the relationships as an equation and solve for the exact value.
Big idea: setting up the equation turns a story into:	**a math sentence you can solve** — Setting up translates the story into a solvable math sentence (an equation) — the bridge from words to answer.

3. schema-based instruction — recognize the problem type and label every number (anti-key-word)

✂ Question (fold →)	Answer
Two problems: (A) 'Tom has 3 bags of 4 apples' and (B) 'Tom has 3 apples and 4 apples.' They differ because:	**A is a multiply structure (3x4), B is an add structure (3+4)** — The STRUCTURE differs: equal groups (3 bags x 4) multiply; two separate amounts add. Recognizing the type is Kindra's skill.
Recognizing that coins and chickens-and-cows share a schema helps because:	**the same solving method works for both** — Same schema means the same set-up-and-solve method transfers — recognizing type lets one method solve many problems.
Tagg's habit is to write a UNIT next to every number. For '6 chickens' he labels:	**6 chickens (the unit is 'chickens')** — Tagg labels each number with what it counts (6 chickens), so numbers never get confused for the wrong thing.
Kindra sorts a new problem by asking what is UNKNOWN and how the knowns RELATE. This is:	**identifying the schema (structure) of the problem** — Finding what's unknown and how the knowns relate reveals the problem's structure (schema) — Kindra's core move.
"Sara had some marbles. She gave away 4 and has 9 left. How many did she start with?" The word 'left' here means you should:	**ADD (9 + 4) to undo the giving-away** — Despite the word 'left', you ADD (9 + 4 = 13) because you're finding the start — proof that key words mislead; structure decides.
In the chickens-and-cows problem, the number '20' could be heads OR legs. Labeling units prevents:	**mixing up which number means what** — Labeling (20 heads, not just 20) stops you from plugging a head-count where a leg-count belongs — Tagg's safeguard.
"A tank loses 3 liters each hour and starts at 24 liters." This is which type?	**a rate-of-change problem (subtract 3 per hour)** — A per-hour loss is a rate-of-change structure (24 minus 3 each hour) — recognized by the 'per hour' relationship.
Big idea: instead of hunting for key words, a strong solver:	**recognizes the problem's type and labels what each number means** — The reliable approach is recognizing the schema (type) and labeling each number's unit — structure over key-word tricks.
"A ribbon is cut into 4 equal pieces of 5 cm." The total length uses which structure?	**equal groups -> multiply (4 x 5)** — Four equal 5 cm pieces is an equal-groups (multiply) structure: 4 x 5 = 20 cm — despite the word 'cut'.
The BEST reason to ignore the word 'altogether' as an automatic 'add' signal is:	**'altogether' can appear in multiply or subtract problems too** — 'Altogether' can head an add, multiply, or comparison problem — the operation comes from the structure, not the word.
Labeling units also helps you CHECK an answer because:	**the answer's unit should match what the question asked for** — If the question asks for chickens and your answer is in legs, the unit mismatch flags an error — units double as a check (Vetta's ally).
"There are 6 legs." Tagg reminds you 6 is a count of:	**legs, not animals** — 6 counts legs, not animals — Tagg's labeling keeps you from treating a leg-count as a head-count.
Getting a word problem 'wrong' most	**misreading the structure, not from bad arithmetic** — Most word-problem errors are structural (wrong set-up), not arithmetic — which is

often comes from:	why Kindra and Tagg focus there. (And being wrong is fine — the round moves on.)
A 'coin problem' schema usually has two facts:	**a count of coins AND their total value** — Coin problems pair a count relationship with a value relationship — recognizing that dual structure is the schema.
Kindra's first question on a word problem is:	**"what TYPE of problem is this?"** — Kindra recognizes the problem's TYPE (its schema) first — that structure, not a key word, guides the solution.
A 'rate problem' schema involves:	**an amount per unit of something (like miles per hour)** — Rate problems pair two quantities as a ratio (per) — recognizing 'per' structure is the schema, not a key word.
A student solves a rate problem by adding because they saw the word 'and.' The real fix is to:	**identify the structure (a rate) and use it, ignoring the word cue** — The fix isn't a better key word — it's recognizing the rate structure and using the matching method (Kindra over key words).
"12 students share 36 stickers equally." The structure is:	**sharing equally -> divide (36 / 12)** — Sharing equally is a division structure: 36 / 12 = 3 stickers each.
Kindra and Tagg work as a pair because recognizing the type AND labeling units together:	**make sure the right method uses the right numbers** — Kindra's type recognition chooses the method; Tagg's unit labels feed it the right numbers — together they beat key-word guessing.
The chickens-and-cows schema has which two relationships?	**total heads (count) and total legs (value)** — Chickens-and-cows has a count relationship (heads) and a value relationship (legs) — the SAME schema as coin problems.
Schema-based thinking is powerful because once you know the TYPE you:	**already know the method that solves it** — Recognizing the type immediately suggests the matching method — schema knowledge turns many problems into familiar ones.
Why are 'key word' tricks (like 'total means add') UNRELIABLE?	**the same word can mean different operations in different problems** — Key words fail because the same word signals different operations depending on the problem's structure — schema beats key words.
The best label for the number in '15 miles per hour' captures:	**miles per hour (a rate, two units linked)** — 15 miles per hour is a rate linking two units; labeling it fully (miles/hour) preserves its meaning — Tagg's careful labeling.
Two word problems have identical numbers but different structures. The right approach is to:	**solve each by its structure, not by the shared numbers** — Same numbers, different structures need different operations — Kindra reads the structure, not the digits.
Tagg says a bare number like '8' in a problem is 'homeless.' He means:	**it needs a unit label so you know what it counts** — A number without a unit is 'homeless' — Tagg gives it a home (8 pens, 8 legs) so it can't be misused.

4. draw a bar/tape diagram of the relationships

✂ Question (fold →)	Answer
From q11, the number of APPLES is:	**12** — Apples are 2 units, each unit is 6, so apples = 2 x 6 = 12.
"A number and its double" as a bar diagram is:	**a short bar and a bar exactly twice as long** — 'Its double' draws as a bar twice as long — a multiplicative comparison, distinct from '2 more' (an additive one).
Bar diagrams are especially loved in early algebra because they:	**make the jump from pictures to equations feel natural** — Unit bars become variables, so bar diagrams bridge naturally into algebraic equations — a gentle on-ramp.
"Ann has 12 stickers; she has 5 more than Ben." A comparison bar for Ben is:	**a bar 5 shorter than Ann's (Ben = 12 - 5 = 7)** — Ann is 5 more than Ben, so Ben's bar is 5 shorter: 12 - 5 = 7 — the diagram shows the comparison directly.
For '30 is shared so that A gets twice B,' the bars are:	**B is 1 unit, A is 2 units, total 3 units = 30 (B=10, A=20)** — B (1 unit) + A (2 units) = 3 units = 30, so a unit is 10; B = 10, A = 20.
Doodle's diagrams are useful across MANY problem types because a picture:	**shows structure the same way regardless of the story** — Bar diagrams reveal part-whole and comparison structure regardless of the story's surface — a universal tool.
Why draw a diagram BEFORE computing?	**it shows the relationships so you pick the right operation** — The diagram makes the structure visible, so the correct operation becomes obvious — Doodle draws to decide.
"Twice as many apples as pears; 18 fruit total." A bar diagram (pears = 1 unit, apples = 2 units) shows total units:	**3 units = 18, so 1 unit = 6** — Pears (1 unit) + apples (2 units) = 3 units = 18, so 1 unit = 6 (pears 6, apples 12) — the unit-bar method.
Equal groups (like '4 boxes of 6') are drawn as:	**4 equal bars each labeled 6** — Equal groups draw as equal bars (4 bars of 6), a picture of multiplication (4 x 6 = 24).
A diagram and an equation are related because both:	**capture the same relationships in different forms** — A diagram and an equation are two representations of the same relationships — Doodle's picture and Lettie's equation agree.
A common mistake a diagram PREVENTS is:	**confusing 'more than' (add) with 'times as many' (multiply)** — A diagram distinguishes '3 more' (a small extra piece) from '3 times' (a bar three times as long) — preventing a classic mix-up.
In a 'part-part-whole' diagram, the whole bar equals:	**the two part-bars joined end to end** — The whole bar is the two parts placed end to end, so part + part = whole — a picture of addition/subtraction.
For chickens-and-cows, a helpful diagram draws:	**a bar for total heads and a bar showing legs stacked by animal type** — Diagramming heads and legs separately reveals how the leg-total splits by animal type — turning the riddle visual.
"After spending 8 dollars, Mia has half her money left." A bar for her original money split in half shows one half equals:	**8 dollars (the amount spent), so she started with 16** — If spending 8 leaves half, then 8 IS one half, so the whole is 16 — the diagram makes this leap clear.
	relationships between quantities that are hard to hold in your

A diagram helps most when a problem has:	**head** — Diagrams shine when several quantities relate — the picture holds the relationships so your mind can reason about them.
The FIRST thing to label on a bar diagram is:	**what each bar represents (its unit)** — Label what each bar represents first (Tagg's units) so the picture stays meaningful — unlabeled bars mislead.
A bar split into 3 equal parts with the whole labeled 15 shows each part is:	**5** — The whole 15 split into 3 equal parts gives 15 / 3 = 5 per part — a division picture.
Doodle's habit models which good problem-solving move?	**represent the problem before rushing to compute** — Doodle models representing (drawing) the problem before computing — understanding first, arithmetic second.
"Sam is 3 times as old as Kim; together 24." Unit bars (Kim 1, Sam 3) give total units:	**4 units = 24, so 1 unit = 6 (Kim 6, Sam 18)** — Kim (1) + Sam (3) = 4 units = 24, so 1 unit = 6; Kim is 6, Sam is 18.
A bar diagram for '15 is 3 more than a number' locates the number as:	**a bar 3 shorter than 15 (the number is 12)** — 15 is 3 more than the number, so the number's bar is 3 shorter: 15 - 3 = 12.
If a diagram shows the whole as 3 equal parts and TWO of those parts are shaded (30 total), the shaded amount is:	**20** — Each of the 3 parts is 30 / 3 = 10; two shaded parts = 20 — a fraction/part picture.
Big idea: a diagram turns a word problem into:	**a picture of relationships you can reason about** — A diagram converts words into a visible map of relationships, making the right operation clear — Doodle's whole method.
The 'unit bar' method turns a comparison into:	**counting equal units that make up the whole** — The unit-bar method expresses each quantity in equal units, then finds one unit's value — a powerful diagram technique.
A bar (tape) diagram represents a quantity as:	**a rectangle whose length stands for the amount** — A bar diagram draws each quantity as a rectangle whose length represents the amount — a picture of the relationships.
When a diagram and your arithmetic disagree, the smart move is to:	**re-check both — the mismatch flags an error to find** — A diagram-vs-arithmetic mismatch signals an error somewhere; re-checking both catches it — the diagram doubles as a check.

5. translate 'in N years / N years ago' into equations

✂ Question (fold →)	Answer
Why is the age GAP a useful anchor in age problems?	**it stays the same past, present, and future** — The age gap is constant across time, giving a reliable relationship to anchor the equations.
Ages must be reasonable: a solution giving a NEGATIVE age means:	**an error somewhere — re-check the set-up** — A negative age is impossible, so it flags a set-up or arithmetic error to find — reasonableness matters (Vetta).
Ana is 12 and Ben is 8. The difference in their ages in 10 years will be:	**4 (the same as now)** — Both gain 10 years, so the gap stays 12 - 8 = 4 — age differences are constant.
"In 4 years, Ben (now x) will be..." is written as:	**x + 4** — Ben's age in 4 years is x + 4.
From q21, solving (x + 3) + (x + 7) = 30 gives Ben's current age:	**10** — (x + 3) + (x + 7) = 2x + 10 = 30 -> 2x = 20 -> x = 10 (Ben 10, Ann 14).
Solve x + (x + 4) = 20 for Ben's age:	**8** — 2x + 4 = 20 -> 2x = 16 -> x = 8 (Ben 8, Ann 12).
Now Ben is x. "In 2 years Ben will be 15" means:	**x + 2 = 15, so x = 13** — x + 2 = 15 gives x = 13 (Ben is 13 now).
"Ana is 3 times Ben's age. Ben is 6." Ana's age is:	**18** — Ana is 3 times Ben's 6 years: 3 x 6 = 18.
"The sum of Ann's and Ben's ages is 20; Ann is 4 older." Using x for Ben:	**x + (x + 4) = 20** — Ben (x) + Ann (x + 4) = 20 is the set-up.
The KEY fact about ages over time is that everyone's age changes by:	**the same amount** — Time adds the same number of years to everyone, so the difference between two ages never changes.
The reason age problems are great practice for equations is they:	**force careful translation of 'past/future' language into +N and -N** — Age problems drill precise translation of time phrases into +N and -N across multiple people — excellent equation practice.
If Sam is x years old now, 3 years ago he was:	**x - 3** — '3 years ago' subtracts 3: x - 3.
Lettie sets 'let x = Ben's age now.' If Ann is 5 years older, Ann's age now is:	**x + 5** — Ann is 5 years older than Ben (x), so Ann = x + 5.
"5 years ago, Ann (now a) was 10." This means:	**a - 5 = 10, so a = 15** — a - 5 = 10 gives a = 15 (Ann is 15 now).
To describe THREE people's ages when the middle one is x, being careful means:	**writing each other age in terms of x using the given relationships** — Express every age in terms of the one letter x using the relationships, so a single equation captures the problem.
"In 6 years Ben will be twice as	**x + 6 = 2x, so x = 6** — x + 6 = 2x -> 6 = x, so Ben is 6 now (12 in six years).

old as he is now." With x = now:	
A common age-problem trap is forgetting that 'in N years':	**adds N to BOTH people, not just one** — 'In N years' adds N to everyone's age — a frequent error is aging only one person.
Which is TRUE about two people's ages?	**their difference is constant, but their ratio changes over time** — The age DIFFERENCE stays fixed, but the RATIO ('twice as old') changes as both grow — a subtle, important age fact.
"Ann is twice Ben's age now." Using x for Ben, Ann is:	**2x** — 'Twice Ben's age' is 2x.
Big idea: age problems are solved by:	**naming one age as x, writing the others and the time-shifts, then solving** — Name one age x, express the others and any past/future shifts in terms of x, then solve — Lettie's set-up for ages.
After solving an age problem, a good check is:	**plug the ages back into the story and see if it fits** — Substituting the found ages back into the story verifies they satisfy every stated relationship (Vetta's look-back).
"Ann is 4 older than Ben. In 3 years their ages sum to 30." The future-sum equation is:	**(x + 3) + (x + 4 + 3) = 30** — In 3 years Ben is x + 3 and Ann is (x + 4) + 3; their sum is 30 — add 3 to BOTH.
If Mia is x years old now, in 5 years she will be:	**x + 5** — 'In 5 years' adds 5 to the current age: x + 5.
Solve x + 2x = 27 for Ben:	**9** — 3x = 27 -> x = 9 (Ben 9, Ann 18).
"Ann is twice Ben's age; their ages add to 27." Set up:	**x + 2x = 27** — Ben (x) + Ann (2x) = 27 is the set-up.

6. count + value constraints, unit-labeling (coins, stamps)

✂ Question (fold →)	Answer
The value equation for coins multiplies:	**each coin count by that coin's value, then adds** — The value equation sums each count times its coin value (10d + 25q = total cents).
The count equation for coins is usually:	**the numbers of each coin type add to the total number of coins** — The count equation adds the numbers of each coin type to the total coin count (e.g., d + q = 12).
Coin/stamp problems are a great place to practice systems because they naturally give:	**two equations (count and value) in two unknowns** — The count and value relationships give two equations in two unknowns — a natural system to practice on.
Using the count equation to write one coin in terms of the other lets you:	**turn two unknowns into one, so a single equation solves it** — Expressing quarters as 12 - d turns two unknowns into one, so the value equation alone can be solved — the substitution method.
A quick reasonableness check for q17: 8 nickels (40c) + 7 dimes (70c) =	**110 cents (matches!)** — 8x5 + 7x10 = 40 + 70 = 110 cents, matching the problem — a Vetta-style check confirms the answer.
"Total value is $2.10" should be entered in the cents-based equation as:	**210 (cents)** — To stay in cents, convert $2.10 to 210 cents before using the value equation — consistent units (Tagg).
Ticket problems ('adult $8, child $5, 30 tickets, $200') are:	**the SAME schema as coins (count + value)** — Ticket problems share the coin schema: a count equation (tickets) and a value equation (dollars) — recognize and reuse.
A coin problem has the SAME structure as chickens-and-cows because both have:	**a count equation and a value/leg equation** — Coins (count + value) mirror chickens-and-cows (heads + legs) — Kindra sees the shared schema and reuses the method.
Tagg's unit-labeling is ESPECIALLY important in coin problems because:	**counts (coins) and values (cents) are easy to swap by accident** — Coin problems mix counts and values, which are easy to confuse; Tagg's labels (coins vs cents) keep the two straight.
Big idea: coin, stamp, and ticket problems are all solved by:	**writing a count equation and a value equation, carefully labeled** — These count-and-value problems all yield to two labeled equations (count and value) — the shared method Kindra and Tagg apply.
From q16, substituting d = 15 - n into 5n + 10d = 110 gives n =	**8 (nickels)** — 5n + 10(15 - n) = 110 -> 5n + 150 - 10n = 110 -> -5n = -40 -> n = 8 nickels (7 dimes).
"20 stamps worth 5 cents and 10 cents each; total 150 cents." If f = five-cent stamps:	**5f + 10(20 - f) = 150** — Value = 5f + 10(20 - f) = 150 — the same count-plus-value schema for stamps.
Solve 5f + 10(20 - f) = 150. Number of five-cent stamps:	**10** — 5f + 200 - 10f = 150 -> -5f = -50 -> f = 10 five-cent stamps (and 10 ten-cent).
Solve 10d + 25(12 - d) = 210. How many dimes?	**6** — 10d + 300 - 25d = 210 -> -15d = -90 -> d = 6 dimes (and 6 quarters).
The VALUE of d dimes (in cents)	

is:	**10d** — Each dime is worth 10 cents, so d dimes are worth 10d cents.
A coin problem always links two facts:	**how many coins there are, and how much they are worth** — Coin problems pair a count relationship (how many) with a value relationship (how much) — the coin schema.
If a problem gives only the count (not the value), you:	**cannot find each coin type — you need the value relationship too** — With only the count you have one equation and two unknowns — the value relationship is needed to pin both down.
"A jar has nickels and dimes, 15 coins, worth 110 cents." First set up:	**count: n + d = 15; value: 5n + 10d = 110** — The system is n + d = 15 (count) and 5n + 10d = 110 (value) — the two coin relationships.
Tagg warns that '5' in a coin problem could mean 5 coins OR 5 cents. The fix is:	**label it: 5 coins vs 5 cents** — Labeling '5 coins' vs '5 cents' prevents mixing a count with a value — Tagg's core safeguard in coin problems.
The value equation for d dimes and (12 - d) quarters totaling 210 cents is:	**10d + 25(12 - d) = 210** — Value = 10d (dimes) + 25(12 - d) (quarters) = 210 cents — the value equation.
If a coin answer comes out as a FRACTION of a coin (like 3.5 dimes), it means:	**an error — coin counts must be whole numbers** — Coin counts must be whole numbers, so a fractional result signals a set-up or arithmetic error (Vetta's reasonableness).
The value of q quarters (in cents) is:	**25q** — Each quarter is 25 cents, so q quarters are worth 25q cents.
"12 coins, all dimes and quarters." If d is dimes, the number of quarters is:	**12 - d** — With 12 coins total, quarters = 12 - d — the count relationship using one letter.
Why keep VALUE in cents rather than mixing dollars and cents?	**one consistent unit avoids arithmetic mistakes** — Using one consistent unit (cents) avoids mixing dollars and cents mid-equation — a Tagg discipline that prevents errors.
Recognizing 'this is a coin problem' immediately tells Kindra to:	**set up a count equation and a value equation** — The coin schema tells Kindra the method up front: write the count equation and the value equation.

7. combined rates — work done per unit time (painters, hoses, machines)

✂ Question (fold →)	Answer
"3 painters paint a room in 4 hours." The TOTAL work is:	**12 painter-hours** — Total work is painters times hours: 3 x 4 = 12 painter-hours (a useful invariant).
More workers means LESS time. This is an example of:	**inverse variation** — Workers and time vary inversely — their product (the total work) is constant.
From q17, how long would 4 workers take?	**3 hours** — 12 worker-hours / 4 workers = 3 hours — doubling workers halves the time.
If A alone takes 3 hours and B alone takes 6 hours, together they take:	**2 hours (rates 1/3 + 1/6 = 1/2)** — Rates 1/3 + 1/6 = 1/2 job/hour, so together 1 / (1/2) = 2 hours — faster than either alone.
If a hose fills a tank in 6 hours, its RATE is:	**1/6 of the tank per hour** — Rate is the fraction of the job done per hour: one tank in 6 hours = 1/6 tank per hour.
The reason 'time = amount / rate' is so useful is it works for:	**any rate problem — filling, driving, printing, painting** — The rate-time-amount relationship applies to any per-unit-time process — a universal rate tool.
If the room is 12 painter-hours of work, how long for 6 painters (same speed)?	**2 hours** — 12 painter-hours / 6 painters = 2 hours — doubling the painters halves the time (inverse relationship).
"2 workers finish in 6 hours." How many worker-hours is the job?	**12 worker-hours** — Total work = 2 workers x 6 hours = 12 worker-hours.
Driving 150 miles at 50 miles per hour takes:	**3 hours** — Distance is the 'amount' and speed the 'rate': time = 150 / 50 = 3 hours — same rate schema.
The relationship time = job / rate means faster rate gives:	**shorter time** — Since time = job / rate, a bigger rate makes the time smaller — rate and time are inversely related.
Big idea: work-rate problems are solved by thinking in:	**rates (job per hour), which ADD when workers combine** — Think in per-hour rates that add when workers combine, then use time = job / combined rate — the core work-rate method.
Hose A fills 1/6 per hour, hose B fills 1/3 per hour. Together their rate is:	**1/2 per hour** — 1/6 + 1/3 = 1/6 + 2/6 = 3/6 = 1/2 tank per hour combined.
The invariant that stays FIXED when you change the number of workers is:	**the total worker-hours (the size of the job)** — The job's total worker-hours is fixed; only how you split it (workers vs time) changes — a handy invariant.
Rate problems are a subtype Kindra recognizes by the word cue:	**'per' (per hour, per day) plus a job to complete** — 'Per' hints at a rate, but Kindra confirms the rate-time-amount structure rather than trusting the word alone.
Lettie labels the three quantities in every rate problem as:	**rate, time, and amount (rate x time = amount)** — Rate problems link rate, time, and amount by rate x time = amount — labeling all three (Tagg-style) organizes the set-up.
Worker A does 1/4 job/hour, worker	**1/2 job/hour, so 2 hours for the job** — 1/4 + 1/4 = 1/2 job/hour, so the job

B does 1/4 job/hour. Together:	takes 1 / (1/2) = 2 hours.
Why does a draining pipe SUBTRACT its rate?	**it removes work, opposing the filling** — A drain works against the fill, so its rate is subtracted — opposing rates combine by subtraction.
A sanity check for a together-time is that it must be:	**less than the faster worker's solo time** — With help the job finishes faster than the quickest worker alone — a reasonableness check (Vetta).
To find how long to make 300 parts at 60 parts/hour:	**300 / 60 = 5 hours** — Time = amount / rate = 300 / 60 = 5 hours.
The most common WRONG move in a together-problem is:	**adding the two times (6 + 3 = 9) instead of the rates** — Adding times (6 + 3 = 9) is the classic error — together must be faster than the fastest worker alone, so add RATES.
When two workers work TOGETHER, you combine them by:	**adding their rates** — Working together, rates ADD (per-hour outputs stack) — a common trap is wrongly adding the times.
"A machine makes 60 parts per hour." In 3 hours it makes:	**180 parts** — Amount = rate x time = 60 x 3 = 180 parts.
A student answers a together-problem as SLOWER than one worker alone. Vetta's check says:	**that's impossible — help can't slow the job, re-check** — Two workers can't be slower than one; a slower together-time signals an error (likely adding times instead of rates).
If the combined rate is 1/2 tank per hour, filling ONE tank takes:	**2 hours** — Time = 1 job divided by the rate: 1 / (1/2) = 2 hours.
One pipe fills in 2 hours, another drains in 4 hours. Net filling rate is:	**1/2 - 1/4 = 1/4 tank per hour** — The drain opposes the fill, so net rate = 1/2 - 1/4 = 1/4 tank/hour (fills in 4 hours net).

8. weighted averages / concentration mixing

✂ Question (fold →)	Answer
A blend's concentration is ALWAYS:	**between the two ingredient concentrations** — A mixture's concentration always lands between the two inputs — a great reasonableness check (Vetta).
The blend price $6/lb is NOT the simple average of $3 and $8 ($5.50) because:	**there is more of the $8 nut, pulling the average up** — There's more of the $8 nut (3 lb vs 2 lb), so the weighted average leans toward $8 — landing at $6, not the simple $5.50.
In a SALT-solution problem, the amount of PURE SALT is:	**the concentration times the volume** — Pure salt = concentration x volume (e.g., 10% of 20 L = 2 L of salt) — the conserved 'pure' part.
Mixture and coin problems are related because both:	**conserve a total (value or pure ingredient) across the combination** — Both conserve a total across a combination (value for coins, pure ingredient for mixtures) — a shared structural idea.
The conserved quantity that makes mixture problems solvable is:	**the total amount of the pure ingredient (salt, or dollars of value)** — The pure ingredient (or total value) is conserved: what goes in equals what's in the blend — the key mixture invariant.
The PRICE PER POUND of that mixture is:	**$6/lb ($30 / 5 lb)** — Blend price = total cost / total weight = 30 / 5 = $6/lb — a weighted average.
The closer the blend price is to one ingredient's price, the:	**more of that ingredient is in the mixture** — A weighted average sits nearest the value with the largest amount — proximity reveals which ingredient dominates.
A 'weighted average' weights each value by:	**how much of it there is** — A weighted average multiplies each value by its amount before averaging, so larger amounts pull harder.
From q08, the blend's concentration (0.8 L salt in 5 L) is:	**16% (0.8 / 5)** — Concentration = pure salt / total = 0.8 / 5 = 0.16 = 16% (between 10% and 20%, weighted toward the larger 20% part).
In a price mixture, the conserved quantity is:	**total dollars of value** — For a price mixture, the total dollar value is conserved (sum of each part's value equals the blend's value).
Mix 2 L of 10% salt with 3 L of 20% salt. Total pure salt is:	**0.8 L (0.2 + 0.6)** — Salt = 0.10x2 + 0.20x3 = 0.2 + 0.6 = 0.8 L of pure salt.
Mixing 2 lb of nuts at $3/lb with 3 lb at $8/lb: the total COST is:	**$30 (2x3 + 3x8 = 6 + 24)** — Cost = 2x3 + 3x8 = 6 + 24 = $30 total.
Vetta checks a mixture answer by asking:	**'is the blend between the two ingredients, and does the conserved total match?'** — Vetta verifies the blend lies between the inputs and the conserved total (salt or value) balances — two reasonableness checks.
From q02, the total WEIGHT of the mixture is:	**5 lb** — Total weight = 2 + 3 = 5 lb.
The blend price of equal amounts of $4 and $10 coffee is:	**$7 (the simple average, since amounts are equal)** — Equal amounts make the weighted average the simple average: (4 + 10)/2 = $7.
Adding PURE WATER (0% salt) to a solution:	**lowers the concentration** — Water adds volume but no salt, so the same salt spread over more liquid lowers the concentration (dilution).

Adding pure salt (100%) to a solution:	**raises the concentration** — Adding pure salt increases the salt without adding water, raising the concentration.
For q14 with x liters of 30% added: the salt equation is:	**0.10(4) + 0.30x = 0.20(4 + x)** — Conserved salt: 0.10(4) + 0.30x = 0.20(4 + x) — the mixture equation.
If a mixture answer gives a concentration ABOVE both ingredients, you know:	**there's an error — re-check the set-up** — A blend can't exceed its strongest ingredient, so a too-high concentration flags an error (Vetta's reasonableness).
Doodle's diagram for a mixture stacks two bars (each ingredient) whose heights show:	**the amount of pure ingredient in each** — Doodle draws each ingredient's pure-part as a bar; stacking them shows the blend's total pure amount.
The weighted average of 40% (in 1 part) and 40% (in 4 parts) is:	**40% (same value, so the blend equals it)** — If both parts are 40%, any blend of them is 40% — a weighted average of equal values is that value.
A mixture problem combines two things and asks about the:	**combined amount and its overall price or concentration** — Mixture problems combine two quantities and ask about the blend's total and its overall price or concentration.
"How much 30% solution to add to 4 L of 10% to make 20%?" Kindra recognizes this as:	**a mixture problem — set up conserved salt** — Combining concentrations to a target is a mixture problem; Kindra sets up the conserved-salt equation.
Big idea: a mixture is a WEIGHTED AVERAGE, so its value:	**lands between the parts, pulled toward whichever there's more of** — A mixture's value is a weighted average — always between the inputs, pulled toward the larger amount — set up via the conserved total.
Solve 0.10(4) + 0.30x = 0.20(4 + x). x = ?	**4 liters** — 0.4 + 0.3x = 0.8 + 0.2x -> 0.1x = 0.4 -> x = 4 liters of 30% solution.

9. start from the goal and reverse the steps

✂ Question (fold →)	Answer
Big idea: work backwards means:	**reverse each operation, in reverse order, from the goal to the start** — Work backwards reverses each operation in reverse order from the goal to the start — Retta's whole method.
To UNDO 'add 5', you:	**subtract 5** — Subtraction undoes addition, so to reverse 'add 5' you subtract 5.
Continuing q15: 6 L was left after HALF drained, so the full tank was:	**12 L** — 6 L is the half that remained, so full = 6 x 2 = 12 L.
Working backwards is really the idea of applying:	**inverse operations in reverse order** — It's applying each step's inverse operation, in the reverse order — undo the last thing first.
To UNDO 'subtract 7', you:	**add 7** — Addition undoes subtraction, so reverse 'subtract 7' by adding 7.
Working backwards is equivalent to solving an equation by:	**'undoing' operations one at a time to isolate the unknown** — Working backwards is exactly how you solve an equation: apply inverse operations to isolate the unknown.
Working backwards is a good match for which real situation?	**figuring out what time to leave, given when you must arrive** — Planning a departure from a required arrival is work-backwards: start at the goal (arrival) and subtract each duration.
"Double a number, then add 3, get 17." The number is:	**7** — Undo 'add 3' (17 - 3 = 14), then undo 'double' (14 / 2 = 7).
"After spending half her money, then $3 more, Mia has $5." Working backwards, before the $3 she had:	**$8 (5 + 3)** — Undo the last step ('spend $3 more') by adding it back: 5 + 3 = $8 before that step.
A great CHECK after working backwards is to:	**run your start value FORWARD and see if it gives the ending** — Running your answer forward through the original steps should reproduce the ending — a clean check (Vetta).
Why is working backwards often EASIER than forwards for these problems?	**the ending is known, so each undo has a definite value** — Because the ending is a known number, each reverse step produces a concrete value, avoiding an unknown-heavy forward set-up.
"A tank was filled, then half drained, then 4 L added, ending at 10 L." Before adding 4 L:	**6 L (10 - 4)** — Undo the last step (add 4 L) by subtracting: 10 - 4 = 6 L.
"A number divided by 4, then plus 1, equals 6." The number is:	**20** — Undo 'plus 1' (6 - 1 = 5), then undo 'divide by 4' (5 x 4 = 20).
When working backwards, you also reverse the ORDER of steps. If forward was 'add 2 then double', backward is:	**halve, then subtract 2** — Reverse the order and each operation: undo the last step (double -> halve) first, then the first step (add 2 -> subtract 2).
Working backwards connects to peg solitaire / Hanoi thinking because you can:	**reason from the goal state back to the start** — Reasoning from a goal state back to the start is a shared idea — reversing steps/moves toward the beginning.

"I think of a number, add 4, get 11." Working backwards, the number is:	**7 (11 - 4)** — Undo 'add 4' by subtracting: 11 - 4 = 7.
To UNDO 'multiply by 3', you:	**divide by 3** — Division undoes multiplication, so to reverse 'multiply by 3' you divide by 3.
The most common mistake in working backwards is:	**forgetting to reverse the ORDER of the steps** — The classic error is undoing steps in the original order; you must reverse the order (last step first).
The work-backwards strategy is best when a problem gives you:	**the ENDING and a chain of steps, and asks for the start** — Work backwards fits when you know the end and the steps taken, and must find the starting value.
Continuing q08: $8 was what remained after spending HALF. So she started with:	**$16 (8 is half, so double it)** — $8 is the half that remained, so the start was double: 8 x 2 = $16.
Retta's habit is to point herself at:	**the goal, then walk the steps in reverse** — Retta starts at the goal and reverses the steps back to the start — the work-backwards move.
"Add 6, then divide by 2, get 8." The number is:	**10** — Undo 'divide by 2' (8 x 2 = 16), then undo 'add 6' (16 - 6 = 10).
"Triple a number, subtract 2, get 19." The number is:	**7** — Undo 'subtract 2' (19 + 2 = 21), then undo 'triple' (21 / 3 = 7).
The mindset Retta models is:	**'I know where this ends — let me undo my way to the start'** — Retta anchors on the known ending and undoes each step back to the start — calm, reverse reasoning.
If forward operations were add-then-multiply, the SINGLE thing you must remember backwards is:	**undo the multiply first, then the add** — The last forward step (multiply) is undone first; then the add — last-done-first-undone.

10. solve a tiny version first to see the structure

✂ Question (fold →)	Answer
"A knockout tournament with 16 teams — how many games to find a winner?" Small case (2 teams):	**1 game; 4 teams -> 3 games; pattern is teams minus 1** — Small cases (2->1, 4->3) show each game eliminates one team, so n teams need n - 1 games (16 -> 15).
Trying MULTIPLE small cases (2, 3, 4) is better than one because it:	**lets you see how the answer CHANGES, revealing the rule** — Several small cases show how the result changes with size, which is exactly what reveals the underlying rule.
The 16-team tournament needs how many games?	**15** — Each game eliminates one team; to reduce 16 to 1 champion you eliminate 15, so 15 games.
From q19, 4 cuts give a maximum of:	**11 pieces** — Following the pattern 1, 2, 4, 7, then +4: the 4th cut gives 11 pieces.
Try-a-smaller-case and Genny's 'generalize' fit together because:	**small cases reveal the rule, and generalizing writes it for any n** — Tad's small cases expose the pattern; Genny's generalizing writes it as a formula for any n — a natural pipeline.
Tad's motto is:	**'if it's too big, make it small enough to understand'** — Tad shrinks a big problem until it's understandable, then grows the insight back up — his motto.
The DANGER of scaling up from small cases is:	**assuming a pattern holds without checking it truly continues** — A pattern from a few cases might not continue; you should justify WHY the rule holds, not just assume it — careful generalizing.
Small cases 2x2=4 and 3x3=9 suggest an nxn grid has:	**n squared small squares (8x8 = 64)** — The small cases (2^2, 3^2) reveal n^2, so an 8x8 grid has 64 small squares.
"How many small squares on an 8x8 checkerboard?" A smaller case is:	**count a 2x2 and 3x3 first (4 and 9)** — A 2x2 has 4 and a 3x3 has 9 small squares; the small cases show it's the side squared (8x8 = 64).
Trying a smaller case is closely related to Nestor's recursion because both:	**use a smaller version of the same problem** — Both lean on a smaller version of the same problem — recursion is 'try a smaller case' made systematic.
When is trying a smaller case NOT helpful?	**when the small case is so tiny it hides the pattern** — A single overly-tiny case can hide the pattern; trying several small sizes is safer to reveal the structure.
"How many handshakes if 10 people all shake hands once?" A smart smaller case is:	**try 2, then 3, then 4 people and look for a pattern** — Trying 2 (1 handshake), 3 (3), 4 (6) reveals the pattern before tackling 10.
"Cut a pizza with 4 straight cuts — max pieces?" Small cases (0,1,2 cuts):	**1, 2, 4 pieces; each cut adds one more piece than the last** — Pieces go 1, 2, 4, 7, 11 — each cut adds one more than the previous cut. Small cases reveal this growth.
The pattern found from small cases should always be:	**verified on one more case before trusting it** — Before scaling, test the discovered rule on another small case to confirm it holds — a Vetta-style check.
	understand the structure on something manageable, then scale up —

Big idea: try a smaller case to:	Shrink to a manageable case, understand the structure, verify the pattern, then scale up — Tad's method.
The small case reveals a pairing: 1+2+3+4 pairs as (1+4)+(2+3) = 5+5. For 1..100 the sum is:	**5050 (100 x 101 / 2)** — Pairing gives 50 pairs of 101: 50 x 101 = 5050 (i.e., 100 x 101 / 2) — the shortcut the small case revealed.
Why solve a SMALLER version first?	**the small case reveals the pattern or method for the big one** — A small case is easy to solve and see clearly; the pattern or method it reveals then transfers to the full problem.
After solving small cases, the KEY step is to:	**find the pattern/rule and check it predicts the next small case** — You extract a rule from the small cases and confirm it predicts the next one before scaling to the big problem.
In chickens-and-cows, Tad first solves it with just:	**a small number of heads (like 3) to see the method** — Tad tries a tiny version (few heads) to see the guess-and-check method, then applies it to the full problem.
"Sum of 1 + 2 + 3 + ... + 100?" A smaller case to try is:	**1 + 2 + 3 + 4 (= 10) to find a shortcut** — Summing 1..4 first (=10) helps you spot the pairing shortcut before summing 1..100.
Tad's strategy when a problem feels too big is to:	**solve a tiny version first** — Tad shrinks a hard problem to a small version he can solve, then uses what he learns to scale up.
Handshakes for n people equals n times (n-1) divided by 2. For 10 people:	**45** — 10 x 9 / 2 = 45 handshakes — the formula the small cases led to.
Which problem is a GOOD candidate for 'try a smaller case'?	**anything with a big number that has a repeating structure** — Problems with a large size and a repeating structure are ideal — the small-case pattern scales up to the big number.
Trying a smaller case builds CONFIDENCE because:	**solving a piece you understand makes the big problem less scary** — Succeeding on a small version builds understanding and confidence, making the full problem approachable (anti-shame framing).
Handshakes for 2, 3, 4 people are 1, 3, 6. The pattern is:	**add the next-smaller count each time (triangular numbers)** — 1, 3, 6, 10... are triangular numbers; each adds one more than the previous gap — a pattern the small cases exposed.

11. constraint + logical elimination in classic brain-teasers

✂ Question (fold →)	Answer
With 27 coins (one heavy), the minimum weighings to find it is:	**3 (27 -> 9 -> 3 -> 1)** — Thirding each time: 27 -> 9 -> 3 -> 1 takes 3 weighings (3 to the 3rd power = 27).
A jug riddle asks for 4 L with 3 L and 5 L jugs. Before pouring, you should note:	**gcd(3,5)=1, so 4 L is possible** — gcd(3,5)=1, so every amount 1-5 (including 4) is measurable — check feasibility before pouring (Evan/Gill style).
Retta approaches a logic riddle by:	**tracking what each clue rules OUT until one possibility remains** — Retta eliminates possibilities each clue forbids until only one remains — systematic logical elimination.
Weighing puzzles connect to number reasoning because each weighing gives you:	**information that cuts the possibilities (like dividing by 3)** — Each weighing (3 outcomes) roughly thirds the candidates — information as a divisor, connecting logic to number reasoning.
You have 9 coins; one is heavier. Split into three groups of 3 and weigh two groups. If they balance, the heavy coin is:	**in the third (unweighed) group** — If the two weighed groups balance, the heavy coin must be in the third group — elimination narrows it down.
From q14, Carol's hat is:	**red** — With Ben green and Ann blue, the only hat left for Carol is red.
"3 friends, 3 hats (red, blue, green); Ann isn't red, Ben is green." Ann's hat is:	**blue (Ben has green, Ann not red, so Ann is blue)** — Ben takes green; Ann can't be red, so Ann is blue (and Carol is red) — elimination.
If a riddle's clues lead to a CONTRADICTION, it means:	**an earlier assumption was wrong — back up and try another** — A contradiction means a guess/assumption was wrong; you backtrack and try a different option — proof by elimination.
Retta stays organized in a long riddle by:	**writing down what's confirmed and what's eliminated as she goes** — Recording confirmed facts and eliminated options keeps a multi-step deduction organized and error-free.
If the scale BALANCES, the two sides are:	**equal in weight** — A level (balanced) scale means both sides weigh the same.
These riddles teach that many problems are solved not by calculating but by:	**reasoning carefully about what MUST be true** — Crossing/pouring/weighing riddles reward careful reasoning about necessary truths, not just arithmetic.
The clever river-crossing insight is that sometimes you must:	**bring something back to keep every state legal** — To keep all states legal, you sometimes ferry an item back — a temporary backward move that enables progress.
A grid/table is a great tool for a logic riddle because it:	**lets you mark what's ruled in or out for each possibility** — A logic grid records which combinations each clue rules in or out, organizing the elimination.
The heart of these riddles is:	**using each result to eliminate possibilities systematically** — These riddles are solved by systematic elimination — each observation removes possibilities until one remains.
Why is splitting into THREE groups (not	**the balance has three outcomes (left, right, equal), each pointing to a third** — A balance gives three outcomes, so dividing into thirds lets

two) smart for weighing puzzles?	one weighing pinpoint which third holds the coin.
With 9 coins (one heavy), how many weighings are needed to find it (best strategy)?	**2** — Each weighing narrows 9 to 3, then 3 to 1 — two weighings suffice.
In a balance-scale puzzle, one pan going DOWN means that side is:	**heavier** — On a balance, the pan that goes down holds the heavier weight.
Big idea: crossing, pouring, and weighing riddles are solved by:	**tracking constraints and eliminating what can't be true** — These brain-teasers yield to tracking constraints and systematically eliminating impossibilities — Retta's reasoning.
You have 8 coins, one FAKE (lighter). Weigh 3 vs 3. If the left is lighter, the fake is:	**among the 3 on the lighter (left) side** — The fake is lighter, so it's on the lighter (left) pan — among those 3 coins.
In a river-crossing, a move is only allowed if AFTER it:	**no forbidden pair is left unattended** — A crossing move is legal only if the resulting state leaves no forbidden pair unattended — a constraint check.
In a jug-pouring riddle, the amounts you can measure are limited to:	**multiples of the GCD of the jug sizes** — Pouring only produces multiples of the jug sizes' GCD — a number fact that bounds the riddle.
These riddles overlap with PuzzleForge's puzzles because both involve:	**constraints, reachable states, and careful step-by-step reasoning** — Both apps share constraint-tracking, reachable-state reasoning, and careful stepwise deduction — sibling skills.
A good FIRST move in any logic riddle is to:	**use the most definite clue to lock something in** — Starting with the most definite clue locks in a fact, which then cascades to eliminate others (most-constrained-first).
The best mindset for these brain-teasers is:	**patient, step-by-step elimination — and it's fine to backtrack** — Patient elimination with willingness to backtrack is the right mindset — being wrong and undoing is part of the method (anti-shame).
Continuing q11 (fake among 3, lighter): weigh 1 vs 1 of them. If they balance, the fake is:	**the third coin (the one not weighed)** — If the two weighed coins balance, neither is the light fake, so it's the unweighed third coin.

12. substitute back + reasonableness (the look-back step)

✂ Question (fold →)	Answer
Vetta pairs naturally with EVERY other cast member because checking:	**confirms whatever method they used actually worked** — Checking verifies any method's result, so Vetta's look-back is the universal final step for the whole cast.
Checking is the step that turns a solver from 'I think it's right' into:	**'I know it's right because it fits every condition'** — Verification via substitution and reasonableness converts hope into justified confidence — the value of look-back.
A negative answer for a COUNT of people means:	**an error — counts of people can't be negative** — You can't have a negative number of people, so a negative result flags a set-up or arithmetic error.
If substituting back does NOT satisfy the original, you should:	**find and fix the error, not accept the answer** — A failed substitution reveals an error; the right response is to find and fix it — the check did its job.
For 'together they finish in 2 hours' when one alone takes 3, Vetta confirms it's reasonable because:	**together (2 h) is faster than the faster worker alone (3 h)** — Working together must be faster than either alone; 2 hours < 3 hours passes the reasonableness check.
"A shirt is 20% off $40." A student answers $60. Vetta's reasonableness check flags it because:	**a discount should make it CHEAPER, not more expensive** — A discount lowers the price, so $60 (more than $40) is unreasonable — the correct sale price is $32.
Estimating BEFORE solving (not just after) helps because:	**you have a target in mind and notice if your work drifts** — Estimating first sets an expectation, so you notice immediately if your computed answer drifts far from it.
The 'look-back' step in problem solving means:	**check your answer against the original problem** — Look-back checks whether the answer actually satisfies the original problem — the crucial final phase.
Big idea: never trust an answer until you've:	**substituted it back and confirmed it's reasonable** — Trust an answer only after substituting it back and confirming it's reasonable — Vetta's look-back discipline.
When an answer is a FRACTION but the question needs whole items (like buses), you should:	**reconsider — often round UP so everyone fits** — A fractional count of whole items is unreasonable; for capacity problems you usually round UP so no one is left behind.
A speed of 500 miles per hour for a BICYCLE is:	**unreasonable — bikes don't go that fast, re-check** — 500 mph is impossible for a bike, so even 'correct-looking' math must be re-checked — reasonableness catches set-up errors.
'Estimation' helps checking by:	**giving a ballpark to compare your exact answer against** — A quick estimate gives a ballpark; if the exact answer is wildly different, something's wrong.
A word-problem answer should ALWAYS be reported as:	**a number WITH its unit, answering the exact question** — Report the number with its unit, matching what the question asked (12 chickens) — a complete, checkable answer.

Estimate 19 x 21 quickly to check an exact answer:	**about 400 (near 20 x 20)** — 19 x 21 is near 20 x 20 = 400 (exact 399); an exact answer far from 400 would be suspect.
You solve 3x + 4 = 19 and get x = 5. Substituting back: 3(5) + 4 =	**19 (checks out)** — 3(5) + 4 = 15 + 4 = 19, matching the equation — the answer checks out.
A good habit is to check an answer using a DIFFERENT method than you solved with because:	**if both agree, you're much more confident it's right** — Independent methods agreeing gives strong confidence; e.g., solve algebraically then verify by guess-and-check.
The look-back step also asks:	**'is there a simpler way I could have solved this?'** — Look-back also reflects on the method — spotting simpler approaches strengthens future problem-solving (Polya's phase 4).
A chickens-and-cows answer of 7 chickens and 5 cows should be checked against BOTH:	**the head count AND the leg count** — A valid answer must satisfy every condition — both the head total and the leg total — so check both.
An answer with the WRONG UNIT (legs when the question asked for chickens) is:	**wrong, even if the number is right** — If the question asks for chickens, an answer in legs is wrong regardless of the number — units are part of the answer (Tagg + Vetta).
Getting an answer WRONG and catching it with a check is:	**a success of the process — the check protected you** — Catching a wrong answer with a check is the process working — being wrong then correcting is exactly the point (anti-shame).
Checking is MOST valuable because it catches:	**set-up mistakes that arithmetic alone won't reveal** — Perfect arithmetic on a wrong set-up gives a wrong answer; substituting back into the story exposes the set-up error.
The most reliable check is to:	**substitute the answer back into the original equation/story** — Substituting the answer back and confirming every condition holds is the strongest check.
For 'x + (x + 4) = 20' you find x = 8. Checking: 8 + 12 =	**20 (correct)** — 8 + (8 + 4) = 8 + 12 = 20 — matches, so x = 8 checks out.
A mixture concentration answer of 25% from ingredients of 10% and 20% is:	**unreasonable — a blend must be BETWEEN 10% and 20%** — A blend's concentration must lie between the inputs; 25% exceeds both, flagging an error (mixture reasonableness).
Vetta's core question is:	**'does this answer make sense in the real situation?'** — Vetta asks whether the answer makes sense in the real situation — reasonableness, not just correct computation.

13. write the rule that works for any numbers

✂ Question (fold →)	Answer
The general rule for 'coin problems' (count c, value v, coin worth w each) for the OTHER coin type is powerful because it:	**solves any coin problem of that shape by substitution** — A generalized coin formula solves every coin problem of that structure by plugging in the specific counts and values.
Genny's approach starts AFTER a specific problem is solved because:	**the worked example shows the structure to generalize** — Solving a specific case reveals the structure; Genny then replaces the numbers with variables to generalize it.
The rule 8n + 5 for ticket cost is a LINEAR rule because:	**cost changes by a constant amount ($8) per extra ticket** — Each extra ticket adds a constant $8, so 8n + 5 is linear — a steady rate of change plus a starting fee.
A generalized rule uses VARIABLES so that:	**the same rule adapts to different numbers** — Variables let one rule adapt to any input — the essence of a general formula.
After writing a general rule, you should:	**test it on a known case to confirm it's correct** — Testing the general rule on a known answer confirms it's correct before relying on it (Vetta-style check).
The BEST test of whether you truly understand a problem is if you can:	**write the general rule and explain why it works** — Being able to state the general rule AND explain why it holds demonstrates true understanding of the structure.
Proving a rule works for ALL numbers (not just tested ones) requires:	**an argument based on the structure, not just examples** — Examples build confidence, but a general proof needs a structural argument for why the rule must always hold.
Using cows = (L - 2H)/2 with H = 10 heads, L = 28 legs:	**4 cows** — cows = (28 - 2x10)/2 = (28 - 20)/2 = 4 — the general rule plugged in.
Big idea: generalizing turns a solved problem into:	**a reusable rule (formula) for every problem of that type** — Generalizing converts a single solution into a reusable formula for all problems of that type — Genny's method.
The sum of two even numbers, 2a + 2b, generalizes to 2(a+b), showing it's:	**always even** — 2a + 2b = 2(a+b), which is 2 times a whole number — proving the sum of evens is always even (a general proof).
Chickens-and-cows with H heads and L legs: the number of cows is:	**(L - 2H) / 2** — If all H animals were chickens there'd be 2H legs; each cow adds 2 extra, so cows = (L - 2H)/2 — a general rule.
"Cost of n tickets at $8 each" is:	**8n** — n tickets at $8 each cost 8n — a general expression for any n.
A general rule that FAILS a known case means:	**the rule is wrong and needs fixing** — If a rule contradicts a known-correct case, the rule is flawed and must be revised.
The general rule for the sum 1 + 2 + ... + n is:	**n(n+1)/2** — Pairing terms gives n/2 pairs each summing n+1, so the total is n(n+1)/2 — the generalized sum.
A general rule is more powerful than a single answer because it:	**solves every future problem of that type instantly** — A general rule handles all future instances by plugging in numbers — far more powerful than one specific answer.
	reason with letters that stand for any number — Working with

Generalizing is a bridge into algebra because it teaches you to:	variables that stand for any number is the heart of algebra — generalizing is the on-ramp.
Genny turns Tad's small-case pattern (1, 3, 6, 10 handshakes) into the rule:	**n(n-1)/2 for n people** — Each of n people shakes n-1 hands, each handshake counted twice, giving n(n-1)/2 — the generalized handshake rule.
Genny's motto is best stated as:	**'once I've solved one, I write the rule for all'** — Genny generalizes from one solved case to a rule for all cases — her defining habit.
"Any even number" can be generalized as:	**2n (for a whole number n)** — Every even number is 2 times a whole number, so 2n generalizes 'any even number'.
Using n(n+1)/2, the sum 1 + 2 + ... + 20 is:	**210** — 20 x 21 / 2 = 420 / 2 = 210.
"Any odd number" can be generalized as:	**2n + 1** — An odd number is one more than an even number: 2n + 1.
"Total cost with a $5 booking fee plus $8 per ticket" generalizes to:	**8n + 5** — Per-ticket cost 8n plus a one-time $5 fee gives 8n + 5 — a general linear rule.
Generalizing connects to Tad because:	**Tad finds the pattern in small cases; Genny writes it as a formula** — Tad's small cases reveal the pattern; Genny formalizes it into a rule — a natural pipeline.
To 'generalize' a solution means to:	**write a rule that works for ANY numbers, not just the ones given** — Generalizing replaces the specific numbers with variables to make a rule that solves every version of the problem.
"A rectangle's perimeter" generalizes to:	**P = 2(length + width)** — Perimeter is two lengths plus two widths: P = 2(l + w) — a formula for ANY rectangle.

14. two equations, two unknowns — substitution and elimination

✂ Question (fold →)	Answer
Substitution is usually easiest when:	**one variable already has a coefficient of 1** — A coefficient of 1 makes isolating that variable clean (no fractions), so substitution is easiest then.
The reason systems matter is they solve problems with:	**two unknowns linked by two conditions** — Systems handle two unknowns pinned down by two conditions — a hugely common real-world shape.
Big idea: a system is solved by turning TWO unknowns into:	**ONE unknown (via substitution or elimination), then solving** — Both methods reduce two unknowns to one solvable equation — the core strategy for systems.
The SOLUTION to a two-variable system is:	**the pair (x, y) that makes both equations true** — A system's solution is the (x, y) pair satisfying every equation at once.
If a system has INFINITELY many solutions, its two equations are:	**really the same line** — If both equations describe the same line, every point on it solves both — infinitely many solutions.
Solve 2(10 - w) + 4w = 28 for w (cows):	**4** — 20 - 2w + 4w = 28 -> 20 + 2w = 28 -> 2w = 8 -> w = 4 cows (and 6 chickens).
If a system has NO solution, its two lines are:	**parallel (never cross)** — Parallel lines never intersect, so the system has no (x, y) satisfying both — no solution.
Sometimes elimination needs you to MULTIPLY an equation first so that:	**a variable's coefficients match (to cancel)** — Scaling an equation makes a variable's coefficients match so adding/subtracting cancels it.
Elimination is usually easiest when:	**a variable has matching or opposite coefficients** — Matching/opposite coefficients let a variable cancel directly, so elimination is easiest then.
Solve 2a + 3k = 34 and a + 2k = 19 by substituting a = 19 - 2k. The prices are:	**a = 11 (adult), k = 4 (kid)** — Substituting a = 19 - 2k: 2(19 - 2k) + 3k = 38 - 4k + 3k = 38 - k = 34, so k = 4 and a = 19 - 8 = 11. Check: 2(11) + 3(4) = 34 and 11 + 2(4) = 19.
The 'substitution' method solves a system by:	**solving one equation for a variable and plugging it into the other** — Substitution isolates one variable from one equation and substitutes it into the other, leaving a single-unknown equation.
A 'system of equations' is:	**two or more equations that must ALL be true at once** — A system is a set of equations that share unknowns and must all hold simultaneously.
For x + y = 12 and x - y = 4, ADDING the equations gives:	**2x = 16, so x = 8** — Adding cancels y: (x + y) + (x - y) = 2x = 16, so x = 8 (and y = 4).
After solving a system, Vetta reminds you to:	**substitute the pair into BOTH original equations** — A valid solution satisfies BOTH equations, so substitute the pair into each to confirm.
The chickens-and-cows system is: c + w = heads and 2c + 4w = legs, where:	**c = chickens, w = cows (count and leg equations)** — c + w = heads (count) and 2c + 4w = legs (value) — two equations in the two unknowns c and w.
From q12, subtracting 2x + 2y = 10 from 2x + 3y = 12 gives:	**y = 2** — (2x + 3y) - (2x + 2y) = y = 12 - 10 = 2, so y = 2 (and x = 3).

For c + w = 10 and 2c + 4w = 28, substitute c = 10 - w into the leg equation to get:	**2(10 - w) + 4w = 28** — Substituting c = 10 - w gives 2(10 - w) + 4w = 28 — one equation in w.
From q09, 2y = 6 gives y = 3; then x =	**2 (from 2x + 3 = 7)** — y = 3, so 2x + 3 = 7 -> 2x = 4 -> x = 2. Solution (2, 3).
"2 adults and 3 kids cost $34; 1 adult and 2 kids cost $19." This is:	**a 2x2 system (adult price a, kid price k)** — Two unknown prices and two total equations form a 2x2 system: 2a + 3k = 34 and a + 2k = 19.
The 'elimination' method solves a system by:	**adding or subtracting the equations to cancel a variable** — Elimination combines the equations (add/subtract, possibly after scaling) so one variable cancels out.
To eliminate x from x + y = 5 and 2x + 3y = 12, multiply the first by 2 to get:	**2x + 2y = 10** — Multiplying x + y = 5 by 2 gives 2x + 2y = 10, matching the 2x in the other equation.
To use elimination on 2x + 3y = 13 and 2x + y = 7, you can:	**subtract the second from the first to cancel 2x** — Both equations have 2x, so subtracting eliminates x: (2x + 3y) - (2x + y) = 2y = 6.
Reck and Lettie together teach that a system is:	**systematic guess-and-check (Reck) made exact by set-up + algebra (Lettie)** — Reck's systematic guess-and-check becomes Lettie's exact algebraic method — the two together are the bridge to systems.
Reck's systematic guess-and-check is the INTUITION behind systems because it:	**narrows guesses using both conditions until they're satisfied** — Systematic guess-and-check adjusts using both conditions — systems formalize that into an exact algebraic method.
A coin system 'n + d = 15, 5n + 10d = 110' is solved the SAME way as chickens-and-cows because:	**both are two linear equations in two unknowns** — Both are 2x2 linear systems (count + value), so substitution or elimination solves them identically.

15. which solving route fits which problem

✂ Question (fold →)	Answer
A problem with confusing relationships between amounts. A great FIRST step:	**draw a diagram (Doodle)** — When relationships are tangled, a bar diagram (Doodle) makes the structure visible.
Cass's most important message to a stuck learner is:	**'being stuck is normal — try a different plan, and it's okay to not know the way in yet'** — Cass normalizes being stuck and encourages trying another heuristic — the anti-shame, flexible-solver frame.
"How many diagonals in a 12-sided polygon?" Which heuristic fits BEST?	**try a smaller case, then generalize (Tad + Genny)** — Try small polygons (Tad) to find the pattern, then generalize the diagonal formula (Genny).
Cass (the mentor) frames every problem with the four phases:	**understand -> plan -> execute -> look back** — Cass runs Polya's understand -> plan -> execute -> look-back; choosing a heuristic is the plan phase.
"Mix 30% and 10% solutions to get 20%." Which heuristic fits BEST?	**recognize the mixture type, track the conserved amount (Kindra + Doodle)** — Recognize the mixture type (Kindra) and track the conserved ingredient, often with a diagram (Doodle).
The single most COMMON reason to get a word problem wrong is:	**jumping to compute before choosing a fitting plan** — Rushing to compute before understanding and planning is the top error — choose a fitting heuristic first.
"12 coins, dimes and quarters, worth $2.10." Which heuristic fits BEST?	**recognize the coin type, set up count + value (Kindra + Lettie)** — Recognize the coin schema (Kindra) and set up count + value equations (Lettie) — a system.
A problem gives the ENDING and a chain of steps, asking for the start. Best heuristic:	**work backwards (Retta)** — Known ending + a chain of operations to reverse = work backwards (Retta).
Big idea: strong problem-solving is choosing the RIGHT tool by:	**reading the problem's structure, then matching a heuristic to it** — Skilled solving reads the problem's structure and matches the fitting heuristic — the cast's shared meta-lesson.
You can't tell what a problem is asking. Best FIRST move:	**recognize the type and label the units (Kindra + Tagg)** — When lost, identify the problem type and label every number's unit (Kindra + Tagg) to reveal the structure.
A problem that's really a LOGIC deduction (who owns what) fits:	**systematic elimination with a grid (Retta)** — A deduction problem fits systematic elimination, often with a grid (Retta's reasoning).
You've solved one instance and expect many more like it. Best move:	**generalize into a formula (Genny)** — Expecting many similar problems, generalize into a reusable formula (Genny).
Before picking a heuristic, the very first thing to do is:	**understand the problem — what's known, unknown, and asked** — Understanding the problem (Polya's first phase) precedes choosing a plan — you must know the shape to pick the tool.
"I think of a number, triple it, subtract 5, get 16." Which	**work backwards (Retta)** — A known result with reversible steps is a work-backwards problem (Retta): 16 + 5 = 21, 21 / 3 = 7.

heuristic fits BEST?	
If your first heuristic isn't working, the right response is to:	**try a different heuristic — it's fine to switch** — Switching heuristics when one stalls is smart, not failure — flexibility is a core solving skill (anti-shame).
"Find the pattern: 2, 6, 12, 20, 30, ..." Which heuristic fits BEST?	**examine small cases and generalize the rule (Tad + Genny)** — Examine the given small terms (Tad) to spot the rule n(n+1), then generalize (Genny).
Recognizing you can APPROACH a problem two different ways is valuable because:	**solving it two ways lets each check the other** — Two independent approaches agreeing cross-checks the answer, building confidence (a Vetta-style verification).
Which two heuristics form a great TEAM for an unfamiliar big problem?	**try a smaller case (Tad), then generalize (Genny)** — Tad's small cases reveal the pattern and Genny's generalizing captures it as a rule — a powerful pair for the unfamiliar.
"A recipe for 4 needs scaling to 10 people." Which heuristic fits BEST?	**set up a proportion / ratio (rate thinking)** — Scaling a recipe is proportional reasoning — set up a ratio (multiply each amount by 10/4).
A problem with two unknowns linked by two facts. Best heuristic:	**set up a system of equations (Lettie + Reck)** — Two unknowns with two conditions is a system — set it up (Lettie) and solve (Reck).
Several heuristics can often be COMBINED. For example:	**draw a diagram, set up an equation from it, then check reasonableness** — Heuristics chain: diagram (Doodle) into an equation (Lettie), solved, then checked (Vetta).
A problem has a huge number with a repeating structure. Best first move:	**try a smaller case (Tad)** — A big, repeating-structure problem calls for trying a smaller case to reveal the pattern (Tad).
Having MANY heuristics available is like having:	**a full toolbox instead of a single tool** — A range of heuristics is a toolbox — you pick the right tool for each problem's shape.
The BIG lesson of choosing a heuristic is:	**match the strategy to the problem's shape, don't force one method on all** — Skilled solvers pick the heuristic that fits the problem's structure rather than forcing one method everywhere.
You just got an answer. The universal LAST step is:	**check it's reasonable and substitute back (Vetta)** — Every solution ends with the look-back: substitute and sense-check (Vetta).

16. mixed-family review across the whole solving scaffold

✂ Question (fold →)	Answer
"x + y = 9 and x - y = 3." Using elimination (add), x =	**6** — Adding the equations cancels y: 2x = 12 -> x = 6 (y = 3).
The single habit that improves EVERY answer is:	**substituting back and checking reasonableness (Vetta)** — Substituting back and sense-checking (Vetta) verifies any method's answer — the universal improving habit.
A discount answer of $50 for '20% off $40' should be rejected because:	**a discount can't raise the price above $40** — A discount lowers the price; $50 > $40 fails Vetta's reasonableness check (correct answer is $32).
A negative or fractional answer to a 'how many people' question means:	**re-check — people counts are whole and non-negative** — People counts can't be negative or fractional; such a result signals an error to find (Vetta's reasonableness).
"2 adults + 3 kids = $34; 1 adult + 2 kids = $19." Kid price?	**$4** — Substituting a = 19 - 2k: 38 - k = 34 -> k = 4 (adult $11) — check: 2(11)+3(4)=34, 11+2(4)=19.
From q16, Sam has:	**12 marbles** — 3 units = 18 -> unit = 6; Sam is 2 units = 12 (Kim 6).
"Hose fills in 4 h, another in 4 h. Together?"	**2 hours** — Rates 1/4 + 1/4 = 1/2 tank/hour, so together 2 hours — faster than either alone.
Cass's whole-app message is:	**understand it, plan a route, solve, then look back — and being stuck is okay** — Cass frames every problem as understand -> plan -> execute -> look-back, and normalizes being stuck — the ReckonQuest thesis.
"Ann is 3 times Ben's age; together 24." Ben's age?	**6** — Ben (x) + Ann (3x) = 24 -> 4x = 24 -> x = 6 (Ben 6, Ann 18).
"After giving away half, then 2 more books, Sam has 6." How many did he start with (backwards)?	**16** — Undo 'give 2 more' (6 + 2 = 8); 8 is the half that remained, so start = 16 (Retta).
"A tank fills in 3 h, drains in 6 h. Net time to fill?"	**6 hours (net rate 1/3 - 1/6 = 1/6)** — Net rate = 1/3 - 1/6 = 1/6 tank/hour, so it fills in 6 hours (opposing rates subtract).
"How many handshakes among 6 people?" using n(n-1)/2:	**15** — 6 x 5 / 2 = 15 handshakes — the generalized handshake rule (Tad's pattern, Genny's formula).
A bar diagram is MOST helpful for which of these?	**'Sam has twice as many marbles as Kim; together 18'** — A comparison-with-total is ideal for Doodle's unit bars: Kim 1 unit + Sam 2 units = 3 units = 18 (Kim 6, Sam 12).
Chickens and cows: 8 heads, 22 legs. How many cows?	**3 cows** — If all 8 were chickens there'd be 16 legs; the 6 extra come from cows at 2 each: (22 - 16)/2 = 3 cows (5 chickens).
To find the LAST digit of a huge product, the best heuristic is:	**try small cases to find the repeating last-digit pattern (Tad)** — Last digits repeat in cycles; small cases (Tad) reveal the cycle without computing the whole product.
Before solving ANY of these, Kindra	**name the problem type and label every number's unit** — Kindra names

+ Tagg would first:	the type and Tagg labels the units first — the structural foundation for choosing a method.
The reason ReckonQuest teaches the SCAFFOLD (not just answers) is:	**the method transfers to problems you've never seen** — The scaffold (understand-plan-execute-check + a toolbox of heuristics) transfers to brand-new problems — the whole point.
"12 coins, nickels and dimes, worth 95 cents." Number of dimes?	**7 dimes** — n + d = 12 and 5n + 10d = 95; substituting n = 12 - d: 5(12 - d) + 10d = 95 -> 60 + 5d = 95 -> d = 7 dimes (5 nickels).
"Mix 2 L of 10% with 2 L of 30%." Blend concentration?	**20%** — Equal volumes give the simple average: (10% + 30%)/2 = 20% (salt 0.2 + 0.6 = 0.8 L in 4 L = 20%).
The cows-formula cows = (L - 2H)/2 came from Genny GENERALIZING which family?	**chickens-and-cows (count + value)** — Genny generalized the chickens-and-cows (count + value) solution into cows = (L - 2H)/2.
"Double a number, add 7, get 19." The number is (work backwards):	**6** — Undo 'add 7' (19 - 7 = 12), then 'double' (12 / 2 = 6) — Retta's method.
"9 coins, one heavier — fewest weighings to find it?"	**2** — Thirds + a 3-outcome balance narrows 9 to 3 to 1 in 2 weighings (Retta's elimination).
A problem says 'total' but describes a start you must recover after items were removed. You should:	**read the structure — this likely means ADD back, not the word 'total'** — Kindra reads the structure over the key word: recovering a start after removal often means adding back, despite 'total'.
"Sum 1 + 2 + ... + 50" using n(n+1)/2:	**1275** — 50 x 51 / 2 = 2550 / 2 = 1275 — Genny's generalized sum.
Big idea of ReckonQuest: a word problem is beaten by:	**recognizing its type, choosing a fitting method, solving, and checking** — ReckonQuest's thesis: recognize the type, choose the fitting heuristic, solve, and check — the reliable scaffold over key-word tricks.

Keep going

You practiced **400 cards** from ReckonQuest. Come back to the ones you missed — spaced-out practice makes them stick.

Keep exploring online

Play it now	Meet the cast	More free books
`https://spark-and-anvil.org/play/reckonquest`	`https://spark-and-anvil.org/cast/reckonquest`	`https://spark-and-anvil.org/books`

Scan a code with any phone camera, or type the address. Every app, story, and book is free, forever — no account, no tracking.