Lesson 26

technology

Thinking like a computer: precise instructions (unplugged)

Lesson at a glance

Open as a literal robot: pupils call directions to a chair and watch vague wording fail. Name algorithm, bug and debugging, then groups build short sequences with forward / turn-left / turn-right sticky notes on a taped 4-by-4 floor grid, dry-run at tables, and debug one public run with you as robot. Move the same thinking onto the algorithm-sequencer interactive on the IWB (avoid the pond), link to everyday ordered jobs, and record a working sequence plus a named bug-and-fix on the Investigation Journal page.

Learning objectives

- Write a precise ordered sequence of instructions for a simple task
- Spot where a missing or wrong-order step causes a bug and fix it
- Explain that computers and robots only do exactly what they are told

Before the bell — prep

Before pupils arrive, clear floor space and tape a 4-by-4 grid of ~40 cm squares; mark one Start and one nearby Target (3–5 steps including turns). Per table: sticky notes — at least 6 forward, 3 turn left, 3 turn right. Open the algorithm-sequencer interactive on the IWB. Bags clear of the walking lane.

Materials

Item	Qty	Per	Source	Low-cost substitute
masking tape for floor grid	1	class	classroom	chalk lines on a hard playground surface, or large sheets of paper taped into a grid
start and target markers (cones, books or star cards)	2	class	classroom	two sheets of coloured paper labelled Start and Target
instruction sticky notes or scrap card (forward, turn left, turn right)	1	group	classroom	scrap paper squares labelled with simple arrows
Investigation Journal page	1	pupil	provided printable	a blank page ruled into three parts: steps, bug, fix

Lesson 26

Safety watch-point

Calm walking only on the grid — no running. Masking-tape edges can trip; keep bags and chairs clear of the floor lane.

Teaching moves

- **Getting Started:** Stand off-grid facing a chair. Obey every call literally — one tiny step for "go over there," one step only for "walk forward," spin until stopped for "turn." After 60–90 seconds freeze and ask why you missed the chair; draw out that unclear words left you guessing.
- **Exact words, exact order:** Keep board work to about five minutes. Link once to the chair walk: Was that an algorithm yet? Where was the bug? Name debugging live later on the floor rather than as a third word to memorise. Head off "the robot is silly" — the algorithm needs the fix.
- **Direct the robot teacher:** Demo forward = one square and turn = quarter-turn on the spot; class copies both. Model a short path with a planted missing turn, name the bug, insert the fix, re-run to target. Groups build the same Start/facing/Target with sticky notes (~10 min); call time even if imperfect. Mandate every group dry-run (finger-walk or quiet floor step) before one public run where you robot and the class names the buggy step and celebrates the fix.
- **Order the steps on the board:** Open algorithm-sequencer in explore mode: robot bottom-left facing up, target top-right, pond in the middle. Pupils call; you or a volunteer drag forward / turn left / turn right and press run. Keep at least one failed run so debugging stays visible. Fallback: sticky notes on the board and re-walk the floor grid.
- **Instructions older than computers:** Before journal writing, draw everyday algorithms: soda-bread recipe order, packing a bag so the lunchbox is not crushed, gate-to-shop directions. Finish on the sentence: computers and robots only do exactly what they are told, in order.
- **Record our algorithm:** Hand the Investigation Journal page. Every pupil records three things: ordered steps (group dry-run or shared front-run), the buggy step by number, and the fix. Pair-share aloud first; allow numbered arrows instead of full sentences. Check that "bug" names a specific step, not "it broke."

What it should show

A correct floor or screen path reaches Target without leaving squares or crossing the pond — typically forwards along a row/column, a quarter-turn, then forwards again (screen shortest safe route often forward ×4, turn right, forward ×4). Odd landings usually mean a missing/extra turn, a forward that stepped off-grid, or a turn done while moving; re-run from Start facing the agreed way after one fix.

Misconceptions & interventions

- **"The robot is being silly" or "I'm bad at this" when the path fails.** — Freeze on the wrong square and say the robot did its job exactly. Ask which step number sent it wrong; change only that sticky note and re-run so they see the algorithm, not the person, needed the fix.
- **Pupils treat "turn left/right" as walking while turning, or "forward" as several steps.** — Re-demo on one square: forward = one step onto the next square only; turn = pivot a quarter-turn and stay put. Have the group copy one left and one right on the spot before rewriting the sequence.
- **A bug is described vaguely as "it broke" rather than a named step.** — Point at the sticky-note line and ask "Which number?" Require "step 4 should be turn left" before they may change anything; celebrate that precise name as the start of debugging.

Differentiation

Emerging	Developing	Proficient
• Pair at the teacher table with a partly ordered sticky-note path (forwards already placed); pupil adds only the turns, then finger-walks once before any floor run. • Journal: number steps and draw arrows to Target; dictate the buggy step orally for an adult or peer to scribe.	• After a successful dry-run, change one turn and predict the new landing square before testing. • On the algorithm-sequencer, spot the pond-hit step and propose a single-block fix for the class to try.	• Critique the shared class algorithm: is there a shorter safe route, and does it still stay on squares? • Write a second everyday algorithm (toast or bag-pack) and mark where a wrong order would plant a bug.

Cross-curricular hook

Link to English procedural writing and Maths position: ordered steps, quarter-turns and grid paths use the same precise sequence language.