LESSON 7

Pseudo-code and flow charts: plan it, then build it

40 minutes | Strand 1: Introducing computer science | Outcome 1.5 | Outcome 1.6

WHAT THIS LESSON DOES

Write a quiz algorithm as pseudo-code and as a flow chart, build and match-check it in Scratch, then extend the plan, swap with a classmate, and flag where the wording needs to be more precise.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
1.5	develop appropriate algorithms using pseudo-code and/or flow charts
1.6	construct code to implement algorithms

WHAT STUDENTS SHOULD BE ABLE TO DO

- Write a plan as pseudo-code and draw the same plan as a flow chart (LO 1.5)
- Build the plan in Scratch and check the program matches it (LO 1.6)

BEFORE THE BELL

- Have space on the board ready for demonstrating the sample pseudo-code and drawing the matching shapes plan live.
- Pre-assign pairs for the swap step so logistics do not eat minutes.
- Be ready to show the class Scratch save routine live near the end of the extend, swap and flag step (for example File, Save to your computer as QuizPlan_Name in the class folder), so that the saving step that follows is quick.
- Student devices with Scratch; plans from the previous step beside each device
- Student devices with Scratch; paper or a notes app for plans; pairs pre-assigned

Pseudo-code and flow charts: plan it, then build it

HOW THE LESSON RUNS

4 min	introduction	Start
2 min	content	Two ways to write a plan
3 min	content	The quiz you will build, as pseudo-code
5 min	activity	Write the pseudo-code and draw the flow chart
2 min	content	Get something running early
6 min	activity	Build the decision path and run both paths
3 min	content	Check the plan against the blocks
8 min	activity	Extend your plan, swap it, and flag one unclear step
1 min	content	Save your project
3 min	content	Make sense
2 min	content	Reflect
1 min	summary	What you covered

TEACHING MOVES

- Keep the sample pseudo-code on the board and draw the shapes plan live while students write their own; teach shapes only as each matching line appears.
- Get plans started within the first ten minutes; push anyone with two solid pseudo-code lines into Scratch early so something runs before the full build.
- Circulate during planning to check step precision; insist on testable lines such as "if answer equals 32".
- Open the build step with a short whole-class rebuild of the decision diamond so that every student has seen an if-else with both branches working, even if their own build is unfinished; name what you are doing as testing and debugging.
- In the peer step, require the extend, the swap and the precision note from everyone; put a 4-minute timer on the extend rewrite, stop the whole class once the flag is written, and treat the peer build as early-finisher work only.
- Encourage students to flag vague steps in peer plans rather than correcting them silently.
- Show the class save routine live before the final two minutes.

WHAT TO WATCH FOR

- Students write vague steps like 'greet them'. Solution: Insist on specific actions such as 'ask for a name then say hello'.
- Flow charts miss decision diamonds for questions. Solution: Remind them to use diamond shapes for yes/no branches.
- Built program does not match the plan. Solution: Have students cross-check each step after coding; model two lines as a class.
- Score climbs on re-runs because it was never reset. Solution: Require SET score to 0 at the start of the script so plan and program still match.
- Extension plan is only a fragment. Solution: Require a full rewritten short plan a classmate can build from scratch, using the before then after lives example as a model of complete length.

DIFFERENTIATION

- Support: Offer the sample pseudo-code as sentence starters; core finish line is SET score plus the counties decision path only.
- Support: Pair students needing help with those who are confident planners.
- Extension: Early finishers complete the peer build to the decision diamond, or add a costume change on correct still matching the plan.