LESSON 107

Testing, Debugging, and Milestones

80 minutes (2 periods) | Coursework Project Preparation | Outcome 2.19 | Outcome 2.20 | Outcome 2.21 | Outcome 2.22 | Outcome 1.22

WHAT THIS LESSON DOES

Please note: this module follows the 2026 project brief, whose theme was forests. The State Examinations Commission publishes a new brief each year, with its own theme and its own dates. These lessons will be updated once this year's brief is published. Check the current brief on the SEC website before you begin. In this lesson, you will log key milestones from your project using a timeline table, document comprehensive testing of your embedded system and model with results, identify and resolve at least one encountered problem, and detail your Python model or embedded algorithm. Finally, compile these into an 800-word Create section to meet the brief's requirements.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
2.19	test solutions and decisions to determine their short-term and long-term outcomes
2.20	identify and fix/debug warnings and errors in computer code and modify as required
2.21	critically reflect on and identify limitations in completed code and suggest possible improvements
2.22	explain the different stages in software testing
1.22	read, write, test, and modify computer programs

WHAT STUDENTS SHOULD BE ABLE TO DO

- Document key milestones in the development process to demonstrate project progression.
- Conduct and record comprehensive testing throughout the development lifecycle.
- Identify, analyse, and resolve a significant problem encountered during implementation.
- Provide a detailed explanation of the programmed model or embedded algorithm.
- Compile a complete Create section that integrates all required elements coherently.

BEFORE THE LESSON

- Have the official brief open at page 9 so you can display the exact Create wording; students must document to the brief, not to memory.
- Ensure each student can reach their own project files, test results and code from earlier work, since everything they write here must reference what they actually built.
- Confirm where students are saving the Create section, and that it is somewhere backed up, not just on a local machine.

Testing, Debugging, and Milestones

HOW THE LESSON RUNS

TEACHING MOVES

- 2. What the Brief Requires for the Create Section. Read the four brief requirements aloud and write them as a visible checklist. Stress that the 20 marks map directly onto these four elements, so a missing element is lost marks regardless of effort elsewhere.
- 3. Key Milestones You Must Describe. Have students list milestones from their real project history, not an idealised one. Five to eight is plenty. Each row needs a date and a genuine one-line account of what happened, including any dead ends.
- 4. Testing Throughout Development. Insist on real expected versus actual results, including failures. A table of all passes reads as fabricated. Failed rows are where the marks live, because they connect to the problem section.
- 5. Explain One Problem and How You Overcame It. Push students to pick a problem they can actually describe the fix for. Vague problems get vague marks. The 150 to 200 words must show the change made and its measurable effect.
- 6. Describe Your Model or Embedded Algorithm in Detail. Have model-builders walk through their own code line by line and embedded-only students describe their sense-decide-act loop. Check they are explaining what their code does, not pasting it without commentary.
- 7. Writing the Full Create Section. Remind them to connect the parts: a failed test should lead into the problem, a milestone should lead into the model. Coherence between elements is what lifts marks above a list.

WHAT TO WATCH FOR

- Students record only passing tests because they think failures make them look bad. Explain that documented failures with fixes demonstrate a real development process and score well; an all-pass table looks invented and undercuts the problem section.
- Students paste code screenshots and assume that counts as describing the model or algorithm. Require them to explain in words what each key section does and why. Ask them to read a snippet aloud and say what it computes before it goes in.
- Students write generic milestones that could apply to any project. Ask for one specific detail per milestone that only their project would have, such as the exact theme chosen or the sensor wired.

DIFFERENTIATION

- Emerging: Give the student the milestones table first as a low-stakes entry point, then build outward from what they can already recall about their own project. Sit with them and have them talk through one problem before writing, then transcribe what they said into the template.
- Developing: Prompt them to link two elements explicitly, such as which test revealed the problem, rather than writing four disconnected blocks.
- Proficient: Have them describe an alternative fix they considered and rejected, which shows the evaluative reasoning Higher Level rewards. Ask them to check their finished section line by line against the four brief requirements and mark where each is evidenced.