Internal assessment: the computational solution

IB DP Computer Science internal assessment · SL and HL · criteria and deadlines come from the teacher

A project for a real user

> Problem, approach, tested evidence, documentation, each stage done well.

- Your teacher supplies the criteria, word limit and deadlines.
- One fictional project runs through every stage.

What makes a good problem

Question	Why it matters
Can I talk to the user several times?	you need their requirements and feedback
Does it need more than a few lines of code?	you must show a range of techniques
Can I finish it in the time given?	an unfinished product cannot be tested

Use the evidence

Running example: the chess coach spends two hours a week pairing players by hand.

Weak and strong success criteria

Weak	**Strong**
The program is easy to use.	The coach enters a result in under 30 seconds, with at most three clicks.
It stores the players.	It stores each player's name, year group and rating, kept after closing.
Pairings are fair.	No two players meet twice in a row, and rating gaps are as small as possible.
It is secure.	Only the coach, with a password, can change results.

Decompose before you code

Part	Design product
data	fields, data types and an example; a file or database layout
algorithms	a flowchart or pseudocode for each non-trivial process, such as pairing
screens	a sketch of each screen, checked with the user
testing	a test plan, written before coding

Use the evidence

Chess program: players and results, rating updates, weekly pairings, the coach's screens.

A test plan with three kinds of data

Test	Data	Type	Expected
result entry	win for player A	normal	A's rating rises; B's falls
year group	7, the lowest allowed	boundary	accepted
year group	14	invalid	rejected with a message
pairing	9 players	boundary	one bye; no repeat pair

Explaining a technique

- Name it, show an extract, explain it here, justify it.
- A list of Python features is not an explanation.

```
def make_pairings(players):
    """Pair (name, rating) tuples; return the pairs and the bye."""
    ranked = sorted(players, key=lambda p: p[1], reverse=True)
    bye = None
    if len(ranked) % 2 == 1:
        bye = ranked.pop()      # lowest rating sits out
    pairs = []
    for i in range(0, len(ranked), 2):
        pairs.append((ranked[i][0], ranked[i + 1][0]))
    return pairs, bye
```

Other people's work

> Say where borrowed code came from, what you changed, and how it works, in your own words.

- Acknowledge libraries, websites and AI tools.
- Your own explanation shows understanding.

Testing with evidence

Test	Expected	Actual	Met?
3	year group 14 rejected	rejected: "Year group must be 7 to 13"	yes
4	9 players: one bye, no repeat	bye given; one pair repeated	no; fixed and retested

Evaluate against each criterion

Weak

The program works well and the coach likes it.

Add more features.

Strong

Criterion 1 is met: the coach entered results in 12 seconds on average (test 1).

Add printing of the pairings, as the coach asked in feedback.

Practice

Your turn

Try these.

A. Rewrite "The program should be fast" so it can be tested.

B. Give normal, boundary and invalid ages for "accepts ages 11 to 18".

C. Why is the test plan written before the code?

D. Pair ratings 1500, 1420, 1610, 1380, 1550, 1490 with make_pairings.

Check yourself

Your turn

Before you hand in, check each.

A. Is there a real user you consulted, with testable criteria agreed early?
B. Does every criterion have a test, evidence and a judgement?
C. Did you explain a few techniques well, with extract, role and reason?
D. Did you acknowledge every library, code source and AI tool?