

Session 10

Test cases and test report for the matrix transpose web page

This session produces the test cases for the transpose page built in Session 9 and the test report written after running them. Web page testing differs from the C program testing in Session 8: there is no exit code, the input is clicks and keystrokes, and the output is what appears on screen. The cases therefore describe exact screen actions and exact page text. The report at the end is the document a project manager reads; the test case table is what the tester works from.

Objectives

✗ Do not copy. Read for understanding and the viva

- Write test cases for a browser page in terms of user actions and visible results
- Group cases into functional, validation, boundary, and usability categories
- Execute every case in a named browser and record the actual result
- Write a test report with environment, summary counts, defects with severity, and a conclusion

Problem Statement

■ Write in lab record

Develop test cases for the web pages of 9(a). Then, develop test report after testing using the test cases developed.

Concept

✗ Do not copy. Read for understanding and the viva

Four categories

Functional cases check that the page does what the manual asks: two text boxes, Input Elements builds the grid, Compute Transpose shows the transpose. Validation cases feed bad input at each stage and expect a clear message and no wrong output. Boundary cases sit on the edges of the allowed range: 1×1 , 10×10 , 0, 11. Usability cases check things the manual implies but does not say: labels match, errors are readable, a second run does not leave rubbish from the first.

Precondition for a web test

For a page, the precondition is the screen state before the action: "page freshly loaded" or " 2×3 grid shown with all boxes filled". Writing it down stops two testers from getting different results because one had leftovers from a previous case.

Expected output for a page

Say what the user sees, in words a second person can check: the text of the message, the number of boxes, the size and contents of the result table. "Works correctly" is not an expected output.

Severity scale

Use one scale for the whole record. Critical: wrong output or crash on normal input. High: a required feature missing or unusable. Medium: wrong behaviour on unusual but plausible input. Low: cosmetic, or wrong only on input no user would give. Note: an observation with no defect. Every defect in the report carries one of these words, so two readers rate the same defect the same way.

The test report

The report is separate from the case table. It records the environment (browser and version, operating system, date), the counts (total, passed, failed, blocked), the defects with severity, and a one-paragraph conclusion on whether the page is fit to submit. Blocked means the case could not be run, for instance because an earlier failure prevented reaching the state it needs.

Test Cases

■ Write in lab record

Page under test: `transpose.html` from Session 9, opened directly from disk in the browser. Every case starts from a freshly loaded page unless the precondition says otherwise.

TC id	Feature	Precondition	Input	Expected output	Actual output	Status
WT-01	Functional: initial screen	page loaded	none	Two number boxes labelled Rows and Columns and one button labelled Input Elements; no grid; no result	As expected	Pass
WT-02	Functional: grid generation	page loaded	Rows 2, Columns 3, click Input Elements	Heading "Enter 2 × 3 elements", 6 boxes in 2 rows of 3, button Compute Transpose below	Heading, 2 × 3 grid and button shown	Pass
WT-03	Functional: transpose of rectangular matrix	2 × 3 grid shown	fill 1 2 3 / 4 5 6, click Compute Transpose	Heading "Transpose (3 × 2)" and table 1 4 / 2 5 / 3 6	Transpose (3 × 2): 1 4 / 2 5 / 3 6	Pass
WT-04	Functional: transpose of square matrix	2 × 2 grid shown	fill 1 2 / 3 4, click Compute Transpose	Transpose (2 × 2): 1 3 / 2 4	1 3 / 2 4	Pass
WT-05	Functional: column vector	3 × 1 grid shown	fill 7 / 8 / 9, click Compute Transpose	Transpose (1 × 3): single row 7 8 9	7 8 9 in one row	Pass
WT-06	Functional: negative and decimal elements	1 × 2 grid shown	fill -3 and 2.5, click Compute Transpose	Transpose (2 × 1): -3 / 2.5	-3 / 2.5	Pass
WT-07	Functional: repeat with new dimensions	WT-03 completed	change Rows to 1, Columns to 1, click Input Elements	Old grid and old result removed; single box and Compute Transpose shown	Old grid and result cleared, 1 × 1 grid shown	Pass

TC id	Feature	Precondition	Input	Expected output	Actual output	Status
WT-08	Validation: blank dimensions	page loaded	click Input Elements with both boxes empty	Error "Rows and columns must be whole numbers from 1 to 10."; no grid	Error shown, no grid	Pass
WT-09	Validation: zero rows	page loaded	Rows 0, Columns 3, click Input Elements	Same error; no grid	Error shown, no grid	Pass
WT-10	Validation: decimal dimension	page loaded	Rows 2.5, Columns 2, click Input Elements	Same error; no grid	Error shown, no grid	Pass
WT-11	Validation: negative dimension	page loaded	Rows -1, Columns 2, click Input Elements	Same error; no grid	Error shown, no grid	Pass
WT-12	Validation: letters in dimension	page loaded	type abc in Rows, Columns 2, click Input Elements	Same error; no grid	Browser number box does not accept letters; value stays empty; error shown	Pass
WT-13	Validation: blank element	2 × 2 grid shown	leave box (1,2) empty, fill others, click Compute Transpose	Error "Element (1,2) must be a number."; no result table	Error shown, no result	Pass
WT-14	Validation: element error clears on retry	WT-13 state	fill box (1,2) with 9, click Compute Transpose	Error text removed; transpose shown	Error cleared, result shown	Pass
WT-15	Boundary: 1 × 1	page loaded	Rows 1, Columns 1, Input Elements, fill 5, Compute Transpose	Transpose (1 x 1): 5	5	Pass
WT-16	Boundary: 10 × 10	page loaded	Rows 10, Columns 10, click Input Elements	100 boxes in 10 rows; page still usable	100 boxes shown; grid fits within the page width	Pass
WT-17	Boundary: 11 rows	page loaded	Rows 11, Columns 1, click Input Elements	Error; no grid	Error shown, no grid	Pass
WT-18	Boundary: 1 × 10	page loaded	Rows 1, Columns 10, Input Elements, fill 1 to 10, Compute Transpose	Transpose (10 x 1): 1 to 10 in one column	Ten rows, 1 to 10	Pass

TC id	Feature	Precondition	Input	Expected output	Actual output	Status
WT-19	Usability: button labels	page loaded	read the page	Buttons read exactly Input Elements and Compute Transpose	Labels match the manual	Pass
WT-20	Usability: error visibility	WT-08 state	read the page	Error is red text under the button, not an alert box	Red text under the button	Pass
WT-21	Usability: keyboard entry	2 × 2 grid shown	Tab between boxes and type values	Tab moves left to right, top to bottom, then to the button	Tab order follows reading order	Pass
WT-22	Usability: dimension change without clicking Input Elements	2 × 3 grid filled	change Rows to 3 without clicking Input Elements, click Compute Transpose	Transpose still computed for the 2 × 3 grid that is on screen	Transpose (3 × 2) shown, matching the visible grid	Pass
WT-23	Usability: no page reload	2 × 3 grid filled	click Compute Transpose	Page does not reload; typed values remain in the grid	Values remain, no reload	Pass
WT-24	Validation: very large element	1 × 1 grid shown	fill 99999999999999999999, click Compute Transpose	Value displayed as entered	Displayed as 1000000000000000000000	Fail (WD-01)

Test Report

■ Write in lab record

Test environment

Item	Value
Page under test	<code>transpose.html</code> (Session 9), opened from local disk
Browser	Google Chrome 130, default settings, JavaScript enabled
Operating system	macOS 15
Tester	student name and enrolment number
Date	2026-09-26
Test data	as listed in the Input column of each case

Summary

Category	Total	Passed	Failed	Blocked
Functional	7	7	0	0
Validation	8	7	1	0
Boundary	4	4	0	0
Usability	5	5	0	0
All	24	23	1	0

Pass rate: 96 percent. No case was blocked; every case reached the state its precondition needed.

Defects found

Defect id	Found by	Description	Severity	Cause	Suggested fix	Status
WD-01	WT-24	A 20-digit element is displayed as 10000000000000000000, not the digits typed.	Low	<code>Number(v)</code> converts the text to a floating-point value, which cannot hold 20 significant digits exactly.	Keep the element as the trimmed string for display and only use <code>Number</code> for the validity check.	Open
WD-02	WT-12 (observation)	Letters cannot be typed into the dimension boxes because they are <code>type="number"</code> , so the letters case is handled by the browser, not by the page script.	Note	Native number input behaviour.	None needed; record that the validation regular expression still guards paste and non-standard browsers.	Closed

Conclusion

The page meets the problem statement: two text boxes, a button labelled Input Elements that generates the element boxes, and a button labelled Compute Transpose that displays the transpose. All functional, boundary, and usability cases passed. The one failure, WD-01, concerns a 20-digit element outside any realistic use of the page and is rated low; it does not affect the transpose logic. The page is fit for submission with WD-01 recorded as a known limitation.

Viva Questions

✗ Do not copy. Read for understanding and the viva

- **Q:** How does testing a web page differ from testing the C program? **A:** Input is user actions, output is screen text, and there is no exit code; the expected output must describe what the user sees.
- **Q:** What does “blocked” mean in the summary? **A:** The case could not be executed, usually because a prerequisite state could not be reached.
- **Q:** Why is WT-22 a usability case and not a validation case? **A:** The input is valid; the question is whether the page behaves sensibly when the user changes a box without clicking the button.
- **Q:** Why does WT-12 pass even though the page script never saw the letters? **A:** The expected output was “error and no grid”, and that is what happened; the report notes which layer enforced it.
- **Q:** Why is WD-01 low severity? **A:** It needs a 20-digit input, produces a visibly rounded number, and does not affect the transpose of ordinary values.
- **Q:** Why is the test environment recorded? **A:** A different browser or version can behave differently; the report must say where the results hold.

- **Q:** What is the difference between the test case table and the test report? **A:** The table is the tester's working document; the report summarises results, defects, and a fitness decision for the reader.
- **Q:** What would a 100 percent pass rate on the first run suggest? **A:** That the cases were not hard enough; every table should include cases designed to break the page.

Common Mistakes

✕ Do not copy. Read for understanding and the viva

- Writing "works" or "displays correctly" as expected output. Name the text and the numbers.
- Skipping the precondition, so a case that relies on a grid from an earlier case cannot be repeated on its own.
- Testing only 2×2 and 3×3 . The 1×1 , 1×10 , 10×10 , 0, and 11 cases are the ones that find bugs.
- Leaving the browser name and date out of the report.
- Reporting a defect without a severity, or giving every defect the same severity.
- Not retesting after clearing an error (WT-14). A page that shows an error and never recovers is a defect.

Session Summary

■ Write in lab record

- Test case table with 24 cases across functional, validation, boundary, and usability
- Test environment table
- Summary table with total, passed, failed, blocked per category
- Defect table with WD-01 and WD-02, each with severity and suggested fix
- Conclusion stating whether the page is fit to submit

✎ Edit this page

< Previous
Session 9

Next >
Session 11