Part 1: Baseline coverage:

I used the 40 experiments from homework 1 (hw spreadsheet here) and made an automated test suite for them using pytest. Full results with line coverage, branch coverage and summary, can be found here:
https://docs.google.com/spreadsheets/d/1W3YKAGkcWBoeDpk2z8B-Qn5OLWnmVbPB9BRapQ-9iWE/edit?gid=0#gid=0
There is an html also attached on the github along with the code used:
https://github.com/CharlesLiu-Umass/CS520_HW2

Problems were taken from the humaneval dataset and APPs dataset (see more details in homework 1 spreadsheet)

Pytest did not have a convenient way of exporting the data to excel so the %branch and %line could not be calculated but the values are there.

- %line = Stms / Stms + Miss
- %branch = (Branch - BrPart) / Branch

Stms = Statements executed
Miss = Statement not executed
Branch = Total number of branches
BrPart = Number of branches not executed

Overall across all 40 experiments we have:

Stms	Miss	Branch	BrPart
674	316	372	10

Where:

%line	%branch
68.1%	97.3%

Several problems such as problem 2,4 and 10 ran into critical errors during testing which resulted in a bias toward more missed lines and less BrPart branches.

Part 2: LLM-assisted test generation

Here I decided to select an experiment where the generated function was actually able to run, but also had room to improve. The first experiment was using code generated by Gemini using the CoT prompting strategy on problem 1, the close elements problem from the humaneval dataset. This experiment had 83.3% line coverage and 50% branch coverage. Full details on line 1 of the spreadsheet.

The following is my prompt to ChatGPT-5:

You are a python programmer who want to increase code coverage of the following code, you want to target the edge case regarding less than 2 numbers. Respond only with the unit tests in the format ((input_parms), (expected_output)).

Additionally, the generated code was given ChatGPT.

The original code accounts for an edge case that was not present in the tests so I asked ChatGPT to make tests with that edge case in mind which it was able to do. As a result the line and branch coverage become 100%.

The other experiment was code generated by ChatGPT using the SCoT prompting method on problem 8, the s-palendrom problem from APPS. This experiment had 84.6% line coverage and 66% branch coverage. Full details on line 45 on the spreadsheet.

I prompted:

You are a python programmer who want to increase code coverage of the following code, you want to target the last two return statements. Respond only with the unit tests in the format ((input_parms), (expected_output)).

GPT responded with two additional tests:

(("ab"), "NIE")
(("bd"), "TAK")

When added, it caused the line coverage and branch coverage to reach 100%.

Part 3 Buggy Code:

With the first experiment we selected in part 2, I added a bug where instead of testing if the difference between values < threshold, I changed the less than to <=. As a result one of the tests flipped from true to false and was detected. Since our test always runs this inequality, the test case that was made to check that it was a strict less than was able to detect the bug.

With the second experiment we selected in part 2, the algorithm depends on knowing that ‘A’ is symmetrical with itself. This information is stored in a dictionary. When removing this value from the dictionary, the unit test that checks if ‘AA’ is a symmetrical string fails when it should have passed.