Lab 7: Code & Branch Coverage – Submission

# Student Information

**Name:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**SRN:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**Course/Section:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Submission Checklist

☑ Coverage report screenshots (index.html views): Files, Functions, Classes

☑ Final test file: test\_processor\_<YOUR\_SRN>.py

☑ Answers to reflection questions

# Quick Start (Commands)

1. **Install dependencies:**   
   pip install coverage pytest
2. **Run tests with coverage (line + branch), focusing only on lab files:**

python -m coverage run --branch --source=order\_processor.py,test\_processor\_<YOUR\_SRN>.py -m pytest test\_processor\_<YOUR\_SRN>.py

1. **Console report:**

python -m coverage report -m

1. **HTML report:**

python -m coverage html  
(Open htmlcov/index.html in your browser.)

# Screenshots of Coverage Report (after ≥90%)

Insert the following three screenshots from htmlcov/index.html:

1) Files tab (top table) – showing file-wise coverage

2) Functions tab – function coverage view

3) Classes tab – class coverage view (if not present, include the default index overview)

## Screenshot 1: Files View

[Paste screenshot here]

## Screenshot 2: Functions View

[Paste screenshot here]

## Screenshot 3: Classes View / Index Overview

[Paste screenshot here]

# Final Test File (Paste or Attach)

Rename the provided template file to match your SRN (e.g., test\_processor\_PES1UG23AM122.py).

import pytest  
from order\_processor import calculate\_discount, update\_order\_status  
  
# --- calculate\_discount tests ---  
  
def test\_regular\_low\_amount\_zero\_discount():  
 assert calculate\_discount("regular", 500) == 0  
  
def test\_regular\_high\_amount\_five\_percent():  
 assert calculate\_discount("regular", 1500) == 75  
  
def test\_premium\_ten\_percent():  
 assert calculate\_discount("premium", 2000) == 200  
  
@pytest.mark.parametrize("amount,expected", [(6000, 1200), (3000, 300)])  
def test\_vip\_discounts(amount, expected):  
 assert calculate\_discount("vip", amount) == expected  
  
def test\_unknown\_customer\_type\_raises():  
 with pytest.raises(ValueError):  
 calculate\_discount("student", 1000)  
  
# --- update\_order\_status tests ---  
  
def test\_pending\_paid\_true\_moves\_to\_processing():  
 order = {"status": "pending", "paid": True}  
 assert update\_order\_status(order) == "processing"  
 assert order["status"] == "processing"  
  
def test\_pending\_paid\_false\_moves\_to\_awaiting\_payment():  
 order = {"status": "pending", "paid": False}  
 assert update\_order\_status(order) == "awaiting\_payment"  
 assert order["status"] == "awaiting\_payment"  
  
def test\_processing\_items\_available\_true\_moves\_to\_shipped():  
 order = {"status": "processing", "items\_available": True}  
 assert update\_order\_status(order) == "shipped"  
 assert order["status"] == "shipped"  
  
def test\_processing\_items\_available\_false\_moves\_to\_backorder():  
 order = {"status": "processing", "items\_available": False}  
 assert update\_order\_status(order) == "backorder"  
 assert order["status"] == "backorder"

# Reflection (Answer briefly)

1) If the logic in order\_processor.py changes, what’s your strategy to ensure tests and coverage stay updated?

Suggested points to cover:

* • Treat tests as living documentation; update tests alongside code changes.
* • Re-run coverage; inspect missed branches and add tests for new decision paths.
* • Use pull-request gates (e.g., min 90% coverage) and run CI locally.
* • Refactor for testability (pure functions, smaller branches).

2) What are the trade-offs between writing more tests for coverage vs fewer high‑quality tests?

Suggested points to cover:

* • High coverage can miss bugs if tests are superficial; prefer meaningful assertions.
* • Aim for risk-based testing: critical paths first; avoid chasing 100% if it adds little value.
* • Balance: set a floor (e.g., 90%) but prioritize scenario realism and boundary cases.