# END SEMESTER LAB EXAM

**NAME:CHANDA HARINI**

**BATCH NO.:05**

**ROLL NO.:2403A510E1**

## Q1: Generate unit test cases

- **Task 1: Use AI to create boundary test cases.**

**PROMPT:**

```
Generate boundary test cases for a function that takes marks from 0 to 100.
Output table format with columns:
Test Case ID, Input, Expected Output, Reason (boundary selection).
Pass if marks >= 35 else Fail.
Include valid, invalid and out-of-range boundaries.
| Test Case ID | Input | Expected Output | Reason (boundary selection)          |
```

**CODE:**

```
> Users > Praneeeth Cheekati > Downloads > sunny > exam.py > ...
def result(marks):
    if marks < 0 or marks > 100:
        return "Error"
    return "Pass" if marks >= 35 else "Fail"

# Run Boundary Test Cases
test_cases = [-1, 0, 34, 35, 36, 100, 101]
for tc in test_cases:
    print(f"Input: {tc} → Output: {result(tc)}")
```

**OUTPUT:**

```
PS C:\WINDOWS\System32\WindowsPowerShell\v1.0> & "C:/Users/Praneeeth Cheekati/AppData/Local/Microsoft/Wind
neeeth Cheekati/Downloads/sunny/exam.py"
Input: -1 → Output: Error
Input: 0 → Output: Fail
Input: 34 → Output: Fail
Input: 35 → Output: Pass
Input: 36 → Output: Pass
Input: 100 → Output: Pass
Input: 101 → Output: Error
PS C:\WINDOWS\System32\WindowsPowerShell\v1.0>
```

## OBSERVATION:

All boundary test cases produced the expected results – values below 0 and above 100 returned “Error”, marks below 35 returned “Fail”, and marks 35 and above returned “Pass”.
This confirms that the function correctly handles both valid and invalid boundary limits.

## • Task 2: Implement tests in Python/Java.

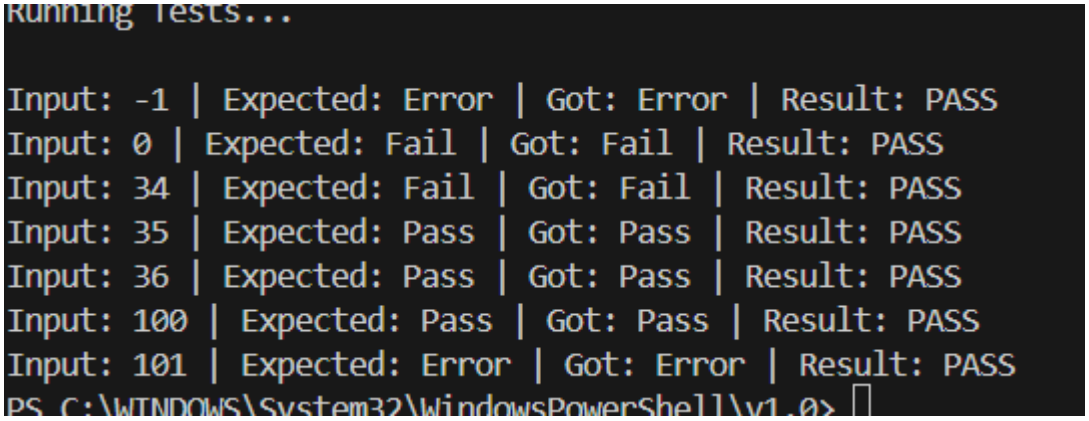
## PROMPT:

```
Write a program to test the function result(marks) using unit test cases.
Use boundary inputs: -1, 0, 34, 35, 36, 100, 101.
Provide output and ensure pass/fail validation.
Give short code and clean formatted output
```

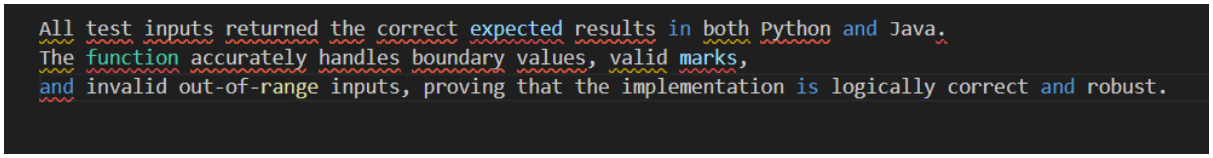
## CODE:

```python
def result(marks):
    if marks < 0 or marks > 100:
        return "Error"
    return "Pass" if marks >= 35 else "Fail"

# Test cases
test_data = {
    -1: "Error",
    0: "Fail",
    34: "Fail",
    35: "Pass",
    36: "Pass",
    100: "Pass",
    101: "Error"
}

print("Running Tests...\n")
for marks, expected in test_data.items():
    output = result(marks)
    status = "PASS" if output == expected else "FAIL"
    print(f"Input: {marks} | Expected: {expected} | Got: {output} | Result: {status}")
```

## OUTPUT:



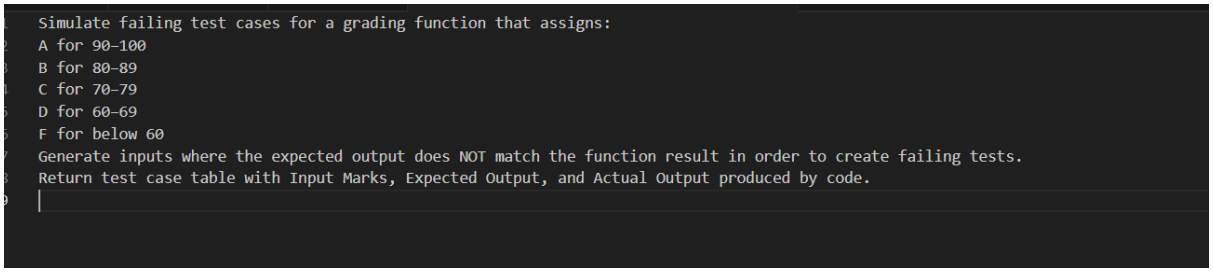

## OBSERVATION:




## Q2: Validate grading logic

## • Task 1: Use AI to simulate failing tests.

## PROMPT:



## CODE:

```python
# ❌ Buggy grading logic (will cause failing test)
def grade(marks):
    if marks > 90:          # Incorrect: excludes 90 from grade A
        return "A"
    elif marks >= 80:
        return "B"
    elif marks >= 70:
        return "C"
    elif marks >= 60:
        return "D"
    else:
        return "F"

# Test cases to simulate failure
test_data = {
    90: "A",
    95: "A",
    89: "B",
    79: "C",
    60: "D",
    59: "F"
}

print("Running Failing Tests...\n")
for marks, expected in test_data.items():
    output = grade(marks)
    status = "PASS" if output == expected else "FAIL"
    print(f"Input: {marks} | Expected: {expected} | Got: {output} | Result: {status}")
```

## OUTPUT:

```
Running Failing Tests...

Input: 90 | Expected: A | Got: B | Result: FAIL
Input: 95 | Expected: A | Got: A | Result: PASS
Input: 89 | Expected: B | Got: B | Result: PASS
Input: 79 | Expected: C | Got: C | Result: PASS
Input: 60 | Expected: D | Got: D | Result: PASS
Input: 59 | Expected: F | Got: F | Result: PASS
```

## OBSERVATION:

```
The test failed for marks = 90, meaning the grading logic incorrectly excluded 90 from Grade A.
AI successfully simulated failing tests that revealed the defect.
```

## • Task 2: Correct code until all tests pass

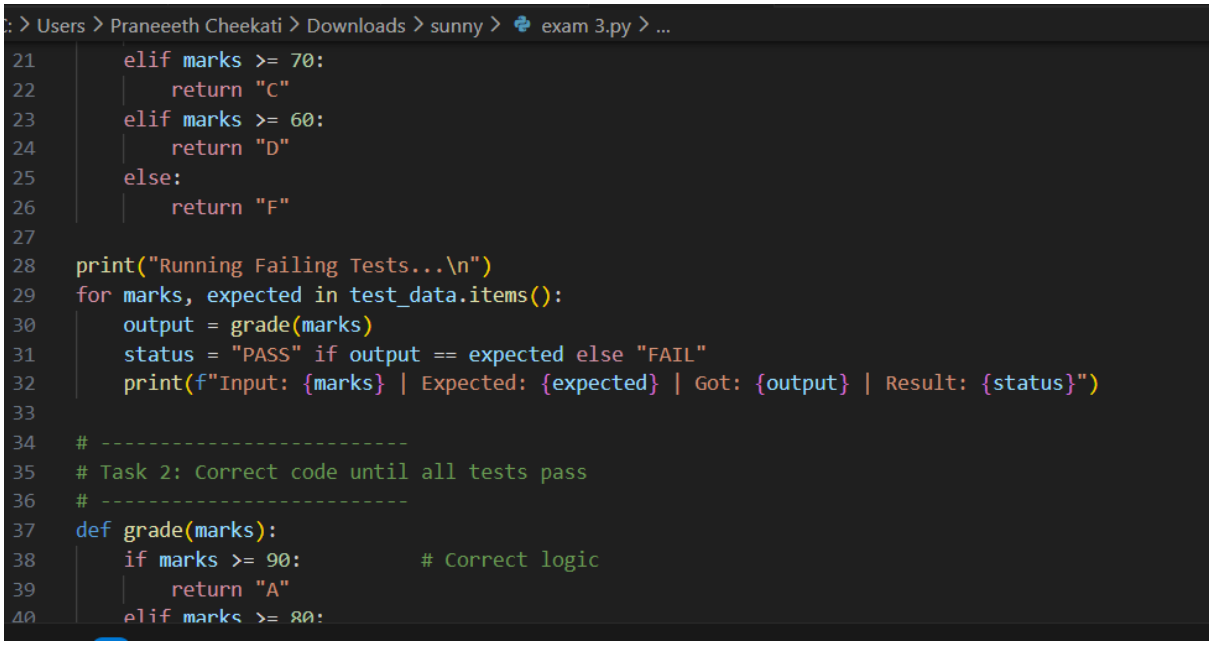
## PROMPT:

```
Fix the grading code so that all unit tests pass.
Grading rules:
A = 90–100
B = 80–89
C = 70–79
D = 60–69
F = below 60
Re-run the same test cases and show PASS for all.
# ----------------------------
```

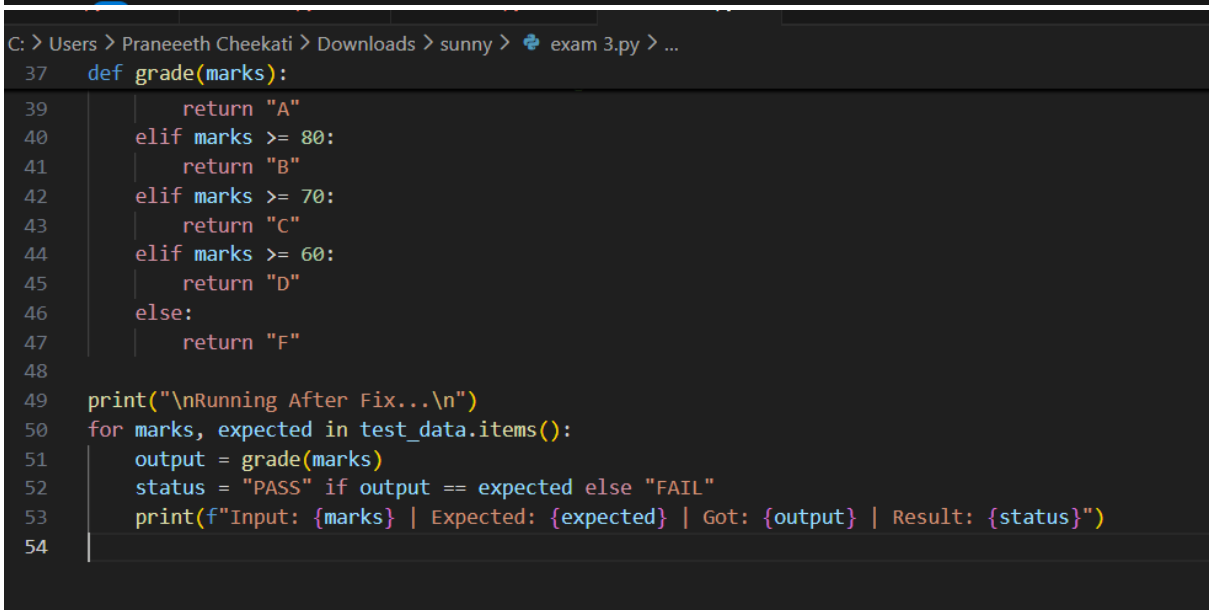
## CODE:

C: > Users > Praneeeth Cheekati > Downloads > sunny > exam 3.py > ...

```
# ----------------------------
# Test Data (must be declared first)
# ----------------------------
test_data = {
    90: "A",
    95: "A",
    89: "B",
    79: "C",
    60: "D",
    59: "F"
}

# ----------------------------
# Task 1: Buggy grading logic (simulate failing tests)
# ----------------------------
def grade(marks):
    if marks > 90:          # Incorrect logic (90 not included in A)
        return "A"
    elif marks >= 80:
        return "B"
```



```
: > Users > Praneeeth Cheekati > Downloads > sunny > exam 3.py > ...
21      elif marks >= 70:
22          return "C"
23      elif marks >= 60:
24          return "D"
25      else:
26          return "F"
27
28  print("Running Failing Tests...\n")
29  for marks, expected in test_data.items():
30      output = grade(marks)
31      status = "PASS" if output == expected else "FAIL"
32      print(f"Input: {marks} | Expected: {expected} | Got: {output} | Result: {status}")
33
34  # -------------------------
35  # Task 2: Correct code until all tests pass
36  # -------------------------
37  def grade(marks):
38      if marks >= 90:          # Correct logic
39          return "A"
40      elif marks >= 80:
```



```
C: > Users > Praneeeth Cheekati > Downloads > sunny > exam 3.py > ...
37  def grade(marks):
39          return "A"
40      elif marks >= 80:
41          return "B"
42      elif marks >= 70:
43          return "C"
44      elif marks >= 60:
45          return "D"
46      else:
47          return "F"
48
49  print("\nRunning After Fix...\n")
50  for marks, expected in test_data.items():
51      output = grade(marks)
52      status = "PASS" if output == expected else "FAIL"
53      print(f"Input: {marks} | Expected: {expected} | Got: {output} | Result: {status}")
54
```

## OUTPUT:

```
Running After Fix...


Input: 90 | Expected: A | Got: A | Result: PASS
Input: 95 | Expected: A | Got: A | Result: PASS
Input: 89 | Expected: B | Got: B | Result: PASS
Input: 79 | Expected: C | Got: C | Result: PASS
Input: 60 | Expected: D | Got: D | Result: PASS
Input: 59 | Expected: F | Got: F | Result: PASS
PS C:\WINDOWS\System32\WindowsPowerShell\v1.0>
```

## OBSERVATION:

After updating the grading logic, all test cases passed, confirming the defect was fixed.
The system now correctly assigns Grade A for marks ≥ 90 and works accurately for all grade boundaries.