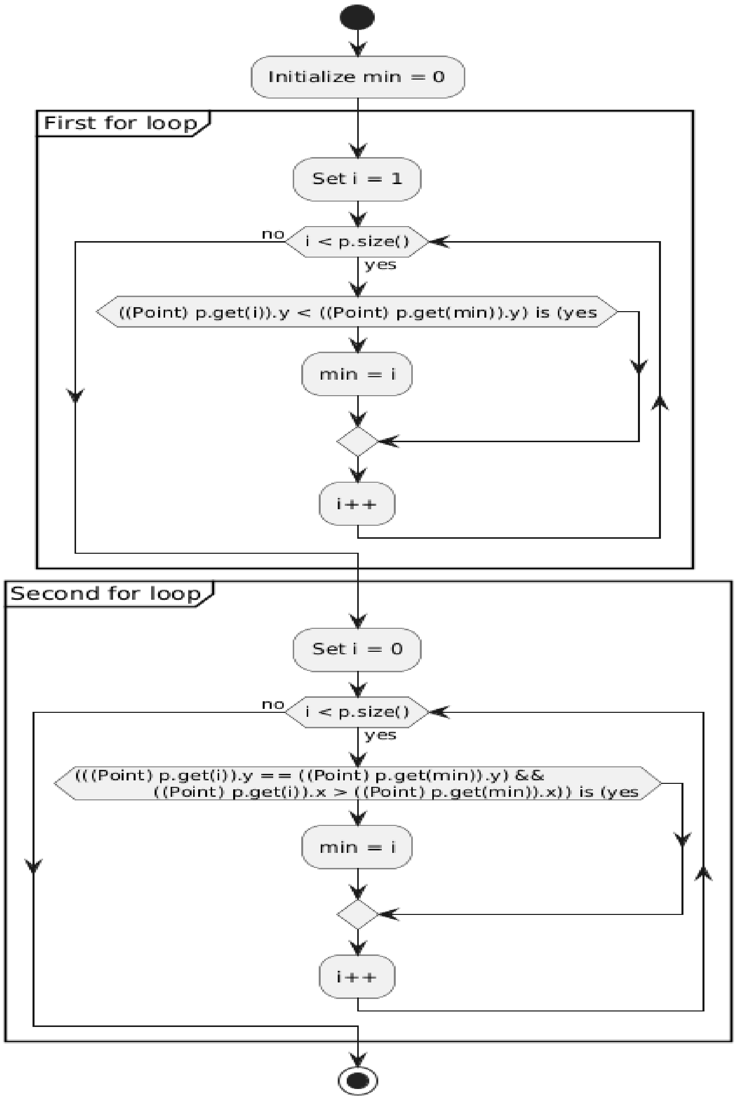
# Software engineering (IT314)

# Lab9 - Mutation Testing

## Name: Saroliya Gunjan

## ID: 202201225

**Answer-1**


Initialize min = 0
First for loop
Set i = 1
no
i < p.size()
yes
((Point) p.get(i)).y < ((Point) p.get(min)).y) is (yes
min = i
i++
Second for loop
Set i = 0
no
i < p.size()
yes
(((Point) p.get(i)).y == ((Point) p.get(min)).y) &&
((Point) p.get(i)).x > ((Point) p.get(min)).x)) is (yes
min = i
i++

## Answer:2

### ⇒ Test Cases for Statement Coverage

We want to ensure every statement in the function is executed at least once.

1. **Test Case 1**: p = [{x: 1, y: 2}, {x: 2, y: 3}]
   - Explanation: This test case ensures the first for loop is executed since p has more than one point.
2. **Test Case 2**: p = [{x: 1, y: 2}, {x: 2, y: 2}, {x: 3, y: 2}]
   - Explanation: This test case ensures the second for loop is executed, as there are points with the same y-coordinate but different x-coordinates.

### ⇒ Test Cases for Branch Coverage

We need to create tests that cover both outcomes of each condition.

1. **Test Case 1**: p = [{x: 0, y: 0}, {x: 1, y: -1}, {x: 2, y: 1}]
   - Explanation: Ensures ((Point) p.get(i)).y < ((Point) p.get(min)).y is true.
2. **Test Case 2**: p = [{x: 0, y: 0}, {x: 1, y: 2}, {x: 2, y: 3}]
   - Explanation: Ensures ((Point) p.get(i)).y < ((Point) p.get(min)).y is false.
3. **Test Case 3**: p = [{x: 1, y: 2}, {x: 2, y: 2}, {x: 3, y: 2}]
   - Explanation: Ensures ((Point) p.get(i)).y == ((Point) p.get(min)).y is true, and ((Point) p.get(i)).x > ((Point) p.get(min)).x is also true.
4. **Test Case 4**: p = [{x: 1, y: 2}, {x: 2, y: 3}, {x: 3, y: 4}]
   - Explanation: Ensures ((Point) p.get(i)).y == ((Point) p.get(min)).y is false

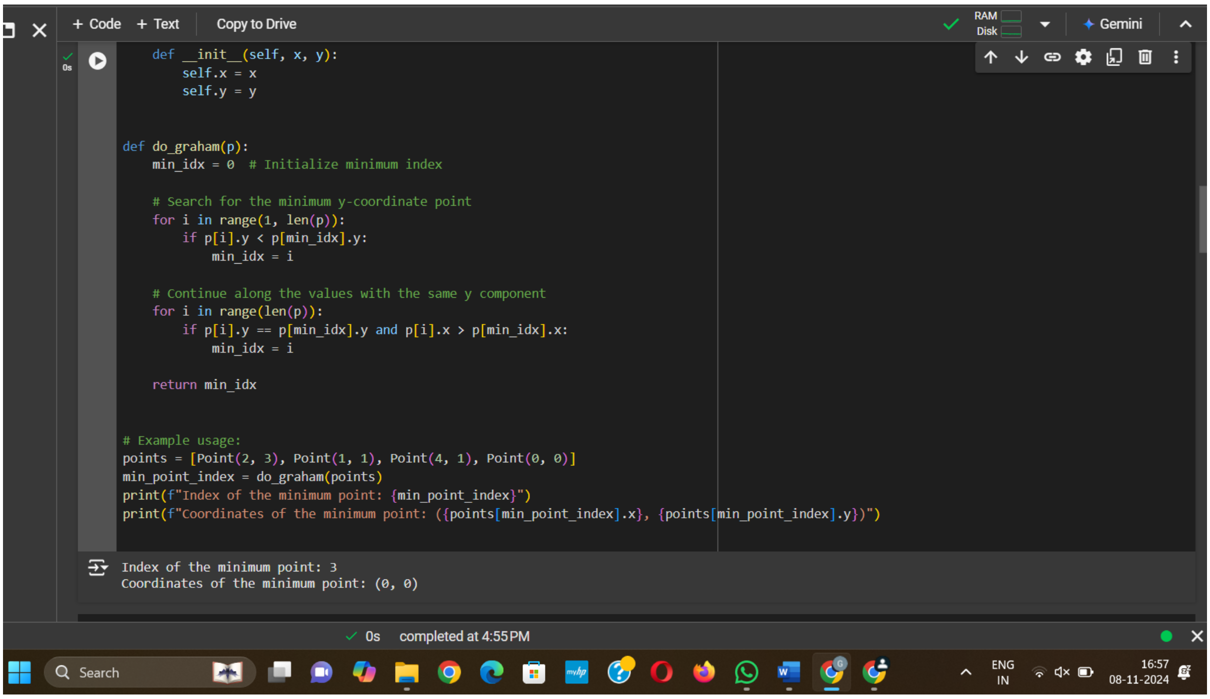
### ⇒ Test Cases for Basic Condition Coverage

We must ensure that every condition is tested for both true and false outcomes.

1. **Test Case 1**: p = [{x: 0, y: 0}, {x: -1, y: -1}, {x: 1, y: 1}]
   - Condition ((Point) p.get(i)).y < ((Point) p.get(min)).y) is both true and false in different iterations.
2. **Test Case 2**: p = [{x: 0, y: 0}, {x: 1, y: 0}, {x: 2, y: 0}]
   - Condition ((Point) p.get(i)).y == ((Point) p.get(min)).y) is true, and ((Point) p.get(i)).x > ((Point) p.get(min)).x) is also tested as true and false.
3. **Test Case 3**: p = [{x: 0, y: 0}, {x: -1, y: 0}, {x: 1, y: 0}]

- Condition ((Point) p.get(i)).x > ((Point) p.get(min)).x) is both true and false.

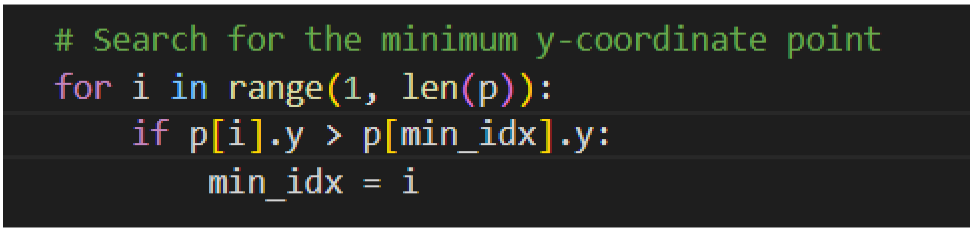
## Answer:3 Mutation Testing:



```python
    def __init__(self, x, y):
        self.x = x
        self.y = y


def do_graham(p):
    min_idx = 0  # Initialize minimum index

    # Search for the minimum y-coordinate point
    for i in range(1, len(p)):
        if p[i].y < p[min_idx].y:
            min_idx = i

    # Continue along the values with the same y component
    for i in range(len(p)):
        if p[i].y == p[min_idx].y and p[i].x > p[min_idx].x:
            min_idx = i

    return min_idx


# Example usage:
points = [Point(2, 3), Point(1, 1), Point(4, 1), Point(0, 0)]
min_point_index = do_graham(points)
print(f"Index of the minimum point: {min_point_index}")
print(f"Coordinates of the minimum point: ({points[min_point_index].x}, {points[min_point_index].y})")
```

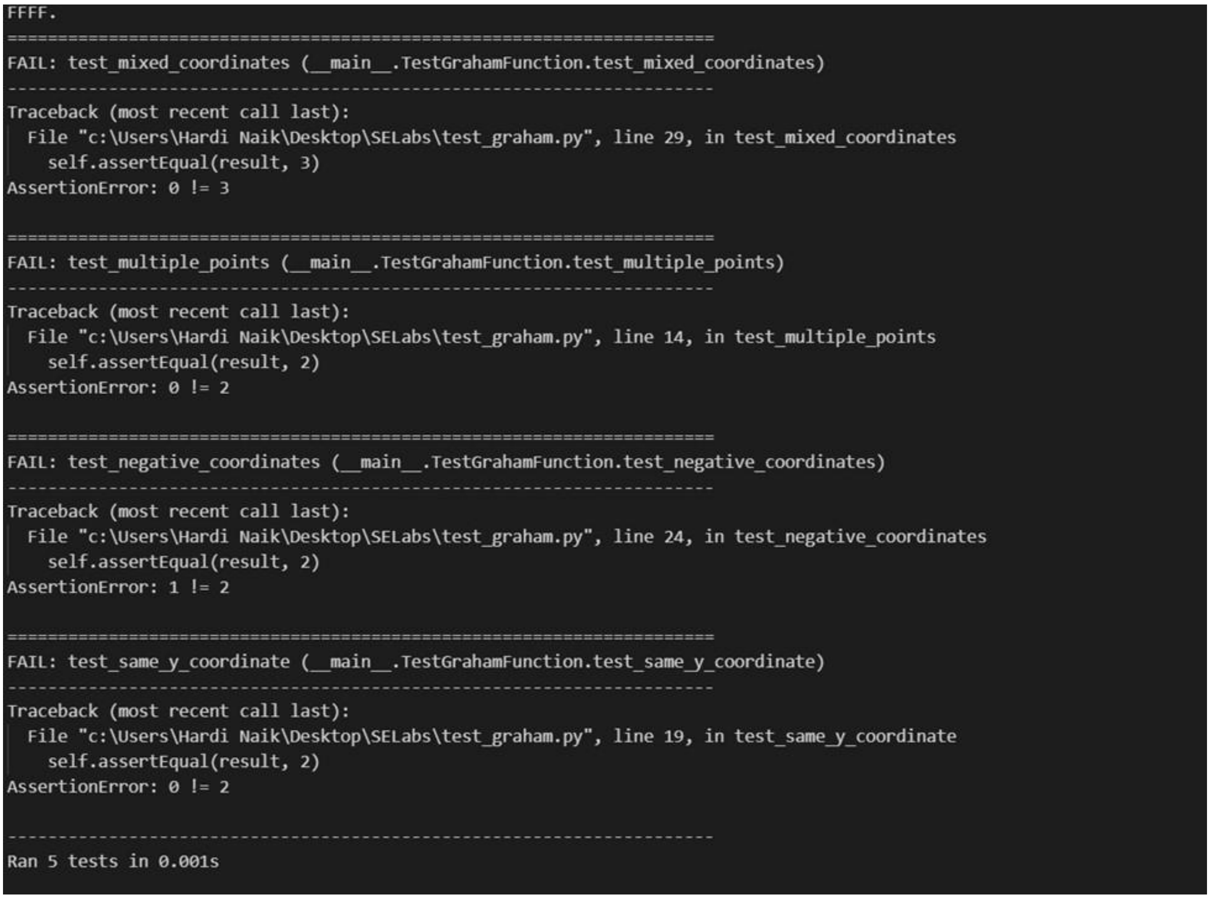
```
Index of the minimum point: 3
Coordinates of the minimum point: (0, 0)
```

## Changes-1 :



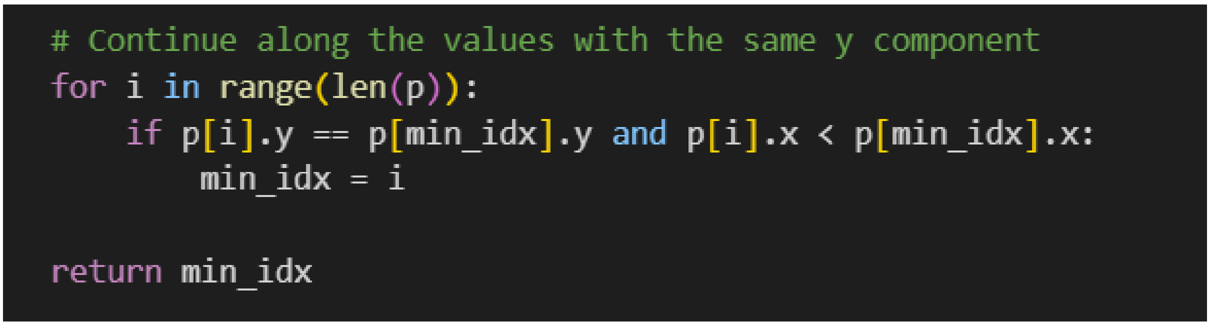
```python
# Search for the minimum y-coordinate point
for i in range(1, len(p)):
    if p[i].y > p[min_idx].y:
        min_idx = i
```

## Result:



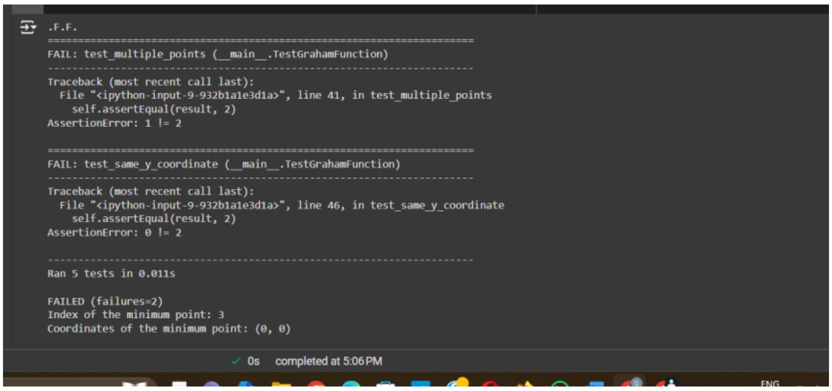
```
FFFF.
======================================================================
FAIL: test_mixed_coordinates (__main__.TestGrahamFunction.test_mixed_coordinates)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "c:\Users\Hardi Naik\Desktop\SELabs\test_graham.py", line 29, in test_mixed_coordinates
    self.assertEqual(result, 3)
AssertionError: 0 != 3

======================================================================
FAIL: test_multiple_points (__main__.TestGrahamFunction.test_multiple_points)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "c:\Users\Hardi Naik\Desktop\SELabs\test_graham.py", line 14, in test_multiple_points
    self.assertEqual(result, 2)
AssertionError: 0 != 2

======================================================================
FAIL: test_negative_coordinates (__main__.TestGrahamFunction.test_negative_coordinates)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "c:\Users\Hardi Naik\Desktop\SELabs\test_graham.py", line 24, in test_negative_coordinates
    self.assertEqual(result, 2)
AssertionError: 1 != 2

======================================================================
FAIL: test_same_y_coordinate (__main__.TestGrahamFunction.test_same_y_coordinate)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "c:\Users\Hardi Naik\Desktop\SELabs\test_graham.py", line 19, in test_same_y_coordinate
    self.assertEqual(result, 2)
AssertionError: 0 != 2

----------------------------------------------------------------------
Ran 5 tests in 0.001s
```

## Changes-2:



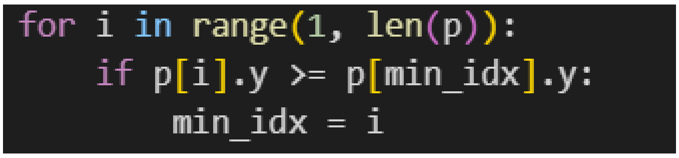
```python
# Continue along the values with the same y component
for i in range(len(p)):
    if p[i].y == p[min_idx].y and p[i].x < p[min_idx].x:
        min_idx = i

return min_idx
```

## Result:



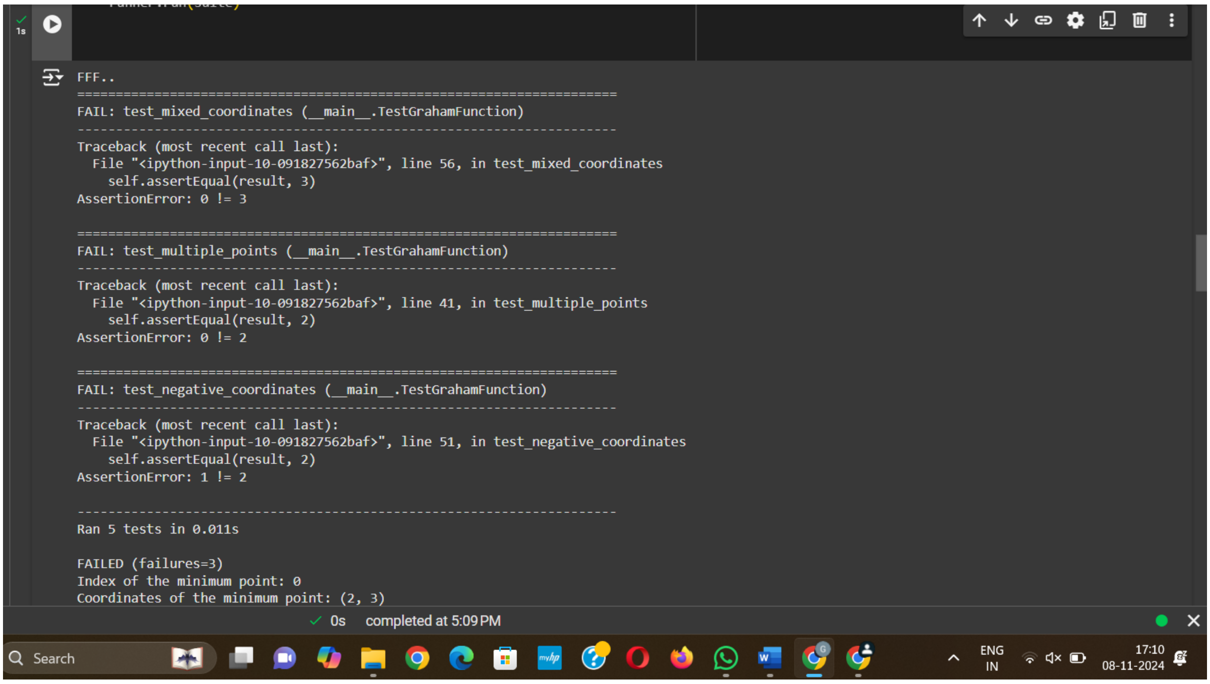
```
.F.F.
======================================================================
FAIL: test_multiple_points (__main__.TestGrahamFunction)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "<ipython-input-9-932b1a1e3d1a>", line 41, in test_multiple_points
    self.assertEqual(result, 2)
AssertionError: 1 != 2

======================================================================
FAIL: test_same_y_coordinate (__main__.TestGrahamFunction)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "<ipython-input-9-932b1a1e3d1a>", line 46, in test_same_y_coordinate
    self.assertEqual(result, 2)
AssertionError: 0 != 2

----------------------------------------------------------------------
Ran 5 tests in 0.011s

FAILED (failures=2)
Index of the minimum point: 3
Coordinates of the minimum point: (0, 0)
```

## Changes-3:



```
for i in range(1, len(p)):
    if p[i].y >= p[min_idx].y:
        min_idx = i
```

## Result:



```
    runner.run(suite)
```

```
FFF..
======================================================================
FAIL: test_mixed_coordinates (__main__.TestGrahamFunction)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "<ipython-input-10-091827562baf>", line 56, in test_mixed_coordinates
    self.assertEqual(result, 3)
AssertionError: 0 != 3

======================================================================
FAIL: test_multiple_points (__main__.TestGrahamFunction)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "<ipython-input-10-091827562baf>", line 41, in test_multiple_points
    self.assertEqual(result, 2)
AssertionError: 0 != 2

======================================================================
FAIL: test_negative_coordinates (__main__.TestGrahamFunction)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "<ipython-input-10-091827562baf>", line 51, in test_negative_coordinates
    self.assertEqual(result, 2)
AssertionError: 1 != 2

----------------------------------------------------------------------
Ran 5 tests in 0.011s

FAILED (failures=3)
Index of the minimum point: 0
Coordinates of the minimum point: (2, 3)
```

## Answer 4:

| TestCase | Input Points | Path Exploration |
|---|---|---|
| Test Case 1 | [] (empty list) | Path with zero iterations of both loops. |
| Test Case 2 | [(0, 1)] | Path with one iteration of the first loop, zero iterations of the second loop. |
| Test Case 3 | [(0, 0), (1, 0)] | Path with two iterations of the first loop, one iteration of the second loop(two points with the same y). |
| Test Case 4 | [(0, 0), (1, 1), (2, 0)] | Path with two iterations of the first loop and one iteration of the second loop.It checks the condition when y is equal and x is different. |
| Test Case 5 | [(1, 1), (2, 2), (0, 3)] | Path with three iterations of the first loop and zero iterations of the secondloop (no points share the same y). |

- **Test Case 1:** Confirms that the function can process an empty input without producing errors.
- **Test Case 2:** Verifies that the function works correctly with a single point as input.
- **Test Case 3:** Checks the behaviour when multiple points share the same y-value,

activating the second loop.

- **Test Case 4:** Assesses the function's handling of varied conditions with different y and x values.
- **Test Case 5:** Ensures the function accurately finds the minimum point when no points have matching y-values.

## AFTER LAB EXECUTION

## Answer-1

Control flow graph factory tool = Yes

Eclipse flow graph generator = Yes

## Answer-2

| Test Case | Input | Expected Outcome | Coverage Achieved |
|---|---|---|---|
| TC1 | [] (empty list) | Should process empty input without errors. | Statement Coverage (initial setup) |
| TC2 | [(0, 0)] | Single point; first loop runs once, second loop is skipped. | Statement Coverage, Branch Coverage (both ifs are false) |
| TC3 | [(1, 1), (0, 0)] | Minimum is (0, 0); tests case with two points. | Statement Coverage, Branch Coverage (first if is true, second if is false) |
| TC4 | [(0, 1), (1, 0)] | Minimum is (1, 0); tests handling of identical y-values. | Statement Coverage, Branch Coverage (both ifs are true) |
| TC5 | [(1, 1), (2, 2), (0, 3)] | Minimum is (1, 1); tests with multiple distinct points. | Statement Coverage, Branch Coverage (first if is true, second if is false) |
| TC6 | [(2, 1), (2, 1), (3, 1)] | Minimum is (2, 1); tests duplicates and matching y-values. | Statement Coverage, Branch Coverage (first if is false, second if is true) |

## Answer-3

| Mutation Type | Description | Effect of Mutation |
|---|---|---|
| Deletion | Remove the initialization of the min variable. | Without min initialization, accessing its value can cause undefined behavior (e.g., accessing an invalid index) if the input is empty. |
| Insertion | Add a line to reset the min variable after the first loop. | - Resetting min to 0 after the first loop can make the program incorrectly identify the minimum point, causing errors.<br>- Test cases may pass, but they won't confirm if min points to the correct minimum after both loops. |
| Modification | Change the comparison operator in the first loop from < to <=. | - Altering the comparison operator could lead to incorrect results when multiple points share the same y value.<br>- Existing test cases may pass but don't verify the impact of this change, particularly in cases with equal y values. |

## Answer-4

| Test Case ID | Input | Expected Output | Description |
|---|---|---|---|
| TC1 | p = [] | Undefined behavior or error | Tests with an empty list to see how the function handles this case. |
| TC2 | p = [(0, 1), (1, 2), (2, 3)] | Minimum point at (0, 1) | Minimum is the first element; checks if the function correctly identifies the initial minimum. |
| TC3 | p = [(1, 2), (0, 1), (2, 3)] | Minimum point at (0, 1) | Minimum appears later in the list; tests if the function correctly updates the minimum. |
| TC4 | p = [(1, 1), (1, 1), (1, 1)] | Minimum point at (1, 1) | All elements have the same y value; checks if the function can handle ties. |
| TC5 | p = [(2, 2), (3, 3), (1, 1)] | Minimum point at (1, 1) | Minimum is the last element; checks if the function correctly identifies the last minimum. |
| TC6 | p = [(2, 3), (1, 3), (0, 3)] | Minimum point at (0, 3) | Minimum is the first element with a shared y value; tests handling of cases with identical y values. |