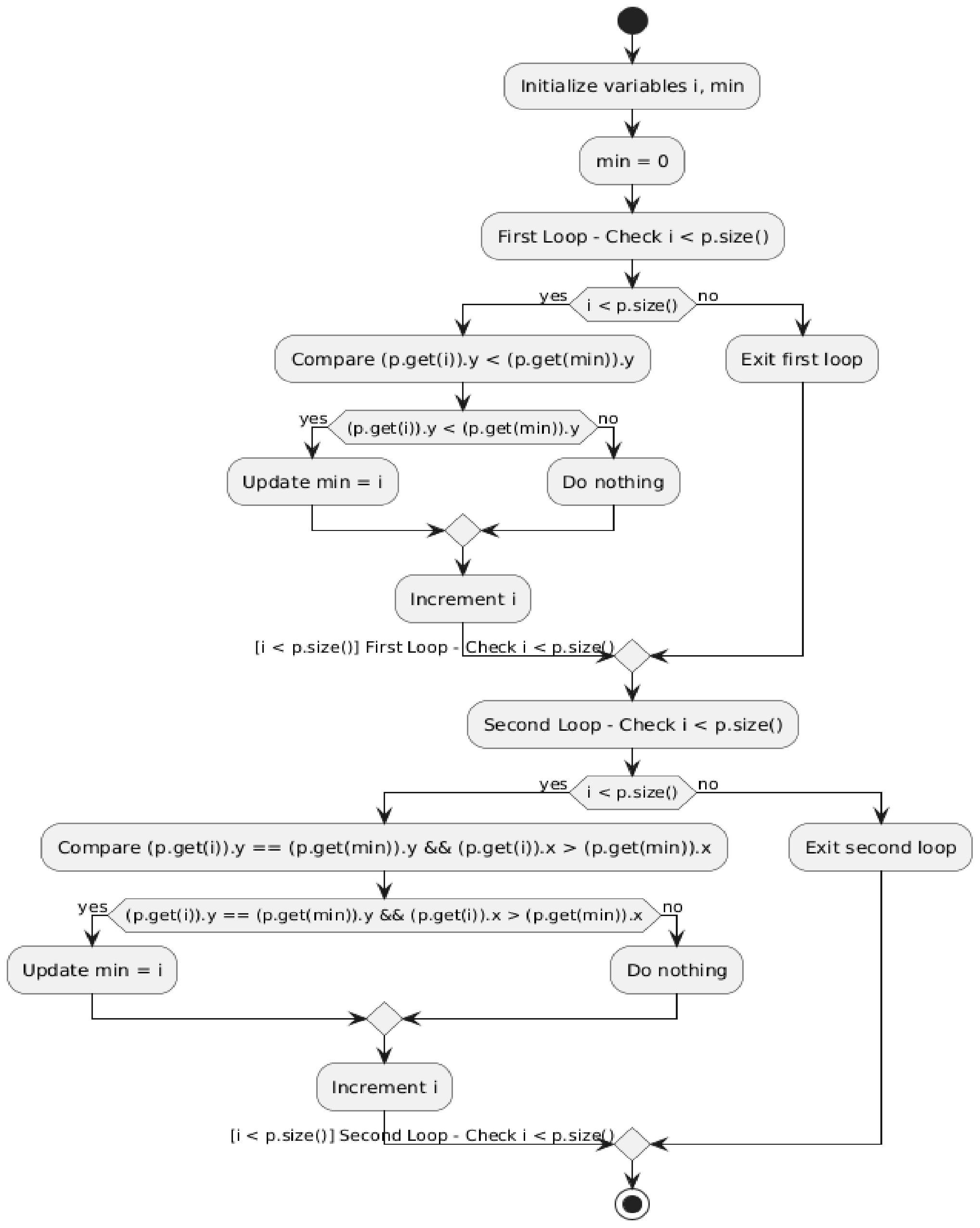
Software Engineering
Lab 09 - Mutation Testing

Name: Aastha Alpeshbhai Bhavsar
ID: 202201259

Q-1. Convert the code comprising the beginning of the doGraham method into a control flow graph (CFG). You are free to write the code in any programming language.
Ans.: CFG:


Initialize variables i, min
min = 0
First Loop - Check i < p.size()
yes
i < p.size()
no
Compare (p.get(i)).y < (p.get(min)).y
Exit first loop
yes
(p.get(i)).y < (p.get(min)).y
no
Update min = i
Do nothing
Increment i
[i < p.size()] First Loop - Check i < p.size()
Second Loop - Check i < p.size()
yes
i < p.size()
no
Compare (p.get(i)).y == (p.get(min)).y && (p.get(i)).x > (p.get(min)).x
Exit second loop
yes
(p.get(i)).y == (p.get(min)).y && (p.get(i)).x > (p.get(min)).x
no
Update min = i
Do nothing
Increment i
[i < p.size()] Second Loop - Check i < p.size()

Q-2. Construct test sets for your flow graph that are adequate for the following criteria:
a. Statement Coverage.
b. Branch Coverage.
c. Basic Condition Coverage.

Ans.: a. Statement Coverage:

Statement coverage ensures each line in doGraham is executed at least once.

Test Set Example for Statement Coverage:

- **Test Case 1**: p = [(0, 0)] — Single point (no loop iteration).
- **Test Case 2**: p = [(1, 1), (2, 0)] — Two points, different y values (tests min selection).
- **Test Case 3**: p = [(1, 1), (2, 1), (3, 1)] — Same y value but different x values (tests second loop).

Test Case	**Input Vector p**	**Description**	**Expected Output**
1	[(0, 0)]	Single point, no loop iteration.	[(0, 0)]
2	[(1, 1), (2, 0)]	Two points, different y values, to test min selection	[(2, 0)]
3	[(1, 1), (2, 1), (3, 1)]	Same y value, different x values	[(3, 1)]

b. Branch Coverage:

Branch coverage tests each possible outcome (true and false) for all conditions.

Test Set Example for Branch Coverage:

- Test Case 1: p = [(0, 0)] — Covers single-point scenario (no updates to min).
- Test Case 2: p = [(1, 1), (2, 0)] — Covers the y comparison in the first loop.
- Test Case 3: p = [(1, 1), (3, 1), (2, 1)] — Covers y equality and x comparison in the second loop.
- Test Case 4: p = [(1, 1), (1, 1), (1, 1)] — All points identical, ensuring both branches in the second loop.

Test Case	**Input Vector p**	**Description**	**Expected Output**
1	[(0, 0)]	Single point, no updates to min.	[(0, 0)]
2	[(1, 1), (2, 0)]	Covers y comparison in the first loop.	[(2, 0)]

3	[(1, 1), (3, 1), (2, 1)]	Tests y equality and x comparison in the second loop.	[(3, 1)]
4	[(1, 1), (1, 1), (1, 1)]	All points identical, ensuring both branches in the second loop.	[(1, 1)]

c. Basic Condition Coverage:

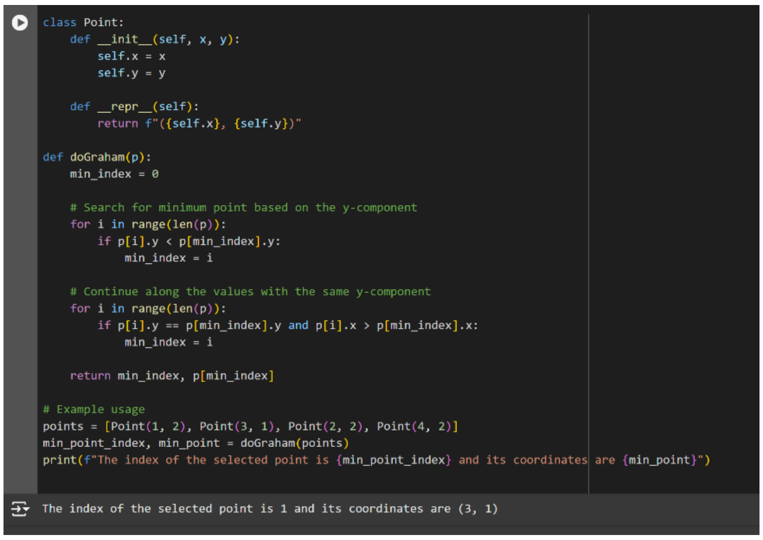
Basic condition coverage checks each individual condition for both true and false outcomes.

Test Set Example for Basic Condition Coverage:

- Test Case 1: p = [(0, 0)] — No comparisons (all conditions default to false).
- Test Case 2: p = [(1, 1), (2, 0)] — First condition true, second false.
- Test Case 3: p = [(1, 1), (3, 1), (2, 1)] — Tests x > min.x condition as true.
- Test Case 4: p = [(1, 1), (1, 1), (1, 1)] — Tests both conditions as false.

Test Case	**Input Vector p**	**Description**	**Expected Output**
1	[(0, 0)]	No comparisons as conditions default to false.	[(0, 0)]
2	[(1, 1), (2, 0)]	First condition true, second false.	[(2, 0)]
3	[(1, 1), (3, 1), (2, 1)]	Tests x > min.x condition as true.	[(3, 1)]
4	[(1, 1), (1, 1), (1, 1)]	Tests both conditions as false.	[(1, 1)]

Q-3. Mutation Testing:

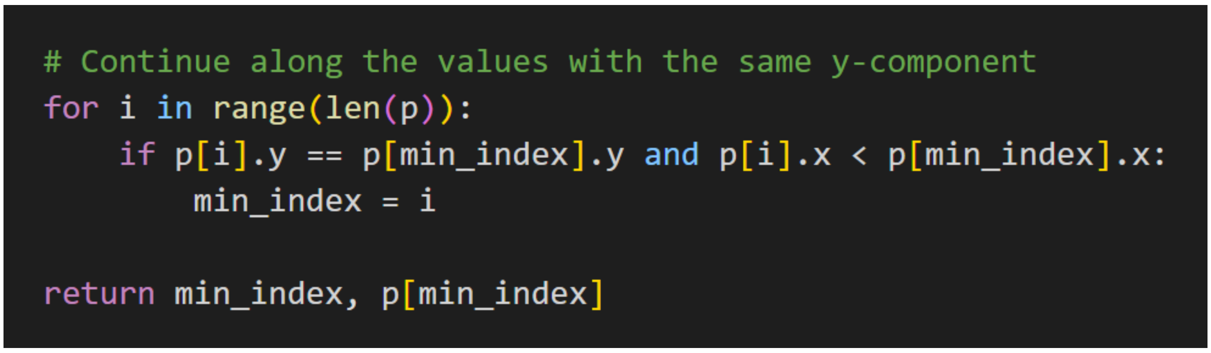


```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __repr__(self):
        return f"({self.x}, {self.y})"

def doGraham(p):
    min_index = 0

    # Search for minimum point based on the y-component
    for i in range(len(p)):
        if p[i].y < p[min_index].y:
            min_index = i

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

    return min_index, p[min_index]

# Example usage
points = [Point(1, 2), Point(3, 1), Point(2, 2), Point(4, 2)]
min_point_index, min_point = doGraham(points)
print(f"The index of the selected point is {min_point_index} and its coordinates are {min_point}")
```

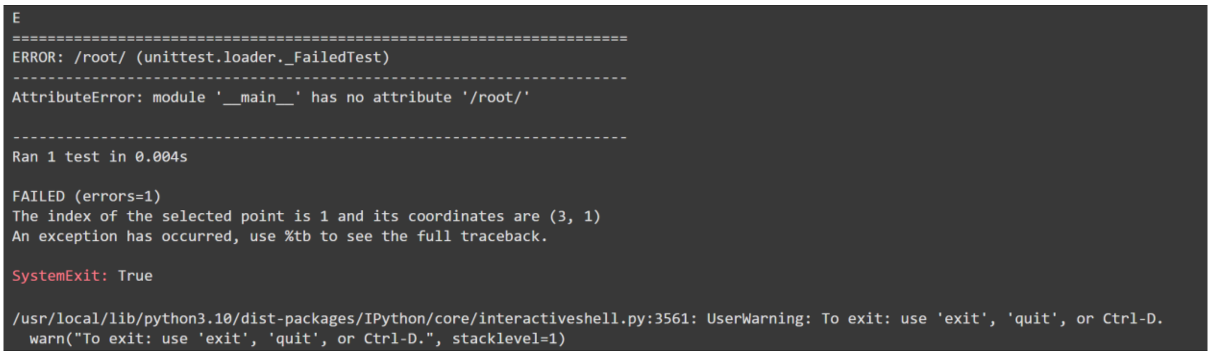
```
The index of the selected point is 1 and its coordinates are (3, 1)
```

After changes:

1.

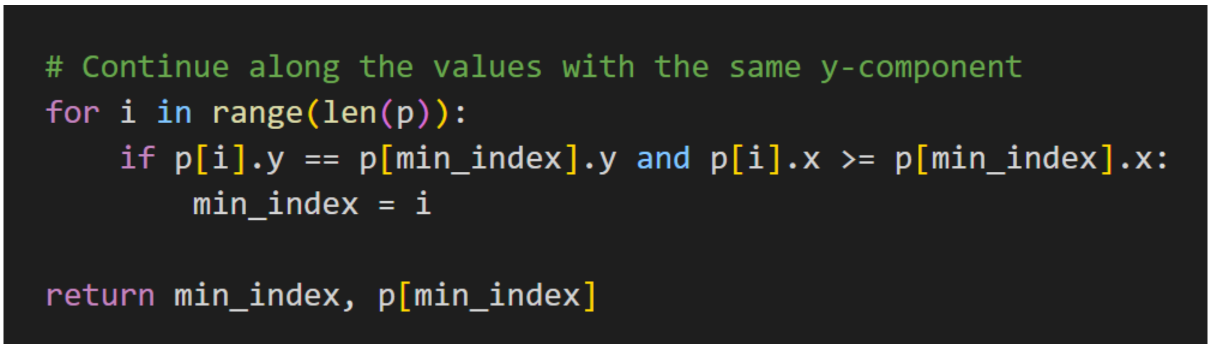


```
# Continue along the values with the same y-component
for i in range(len(p)):
    if p[i].y == p[min_index].y and p[i].x < p[min_index].x:
        min_index = i

return min_index, p[min_index]
```

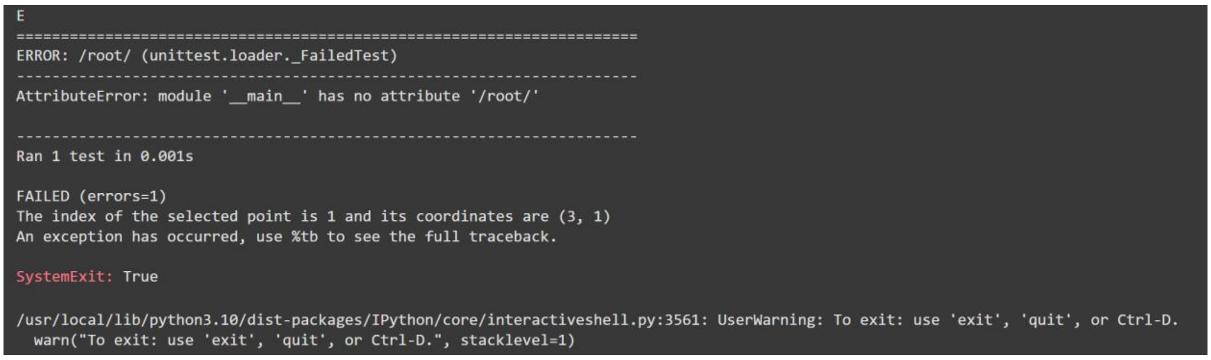


```
E
======================================================================
ERROR: /root/ (unittest.loader._FailedTest)
----------------------------------------------------------------------
AttributeError: module '__main__' has no attribute '/root/'

----------------------------------------------------------------------
Ran 1 test in 0.004s

FAILED (errors=1)
The index of the selected point is 1 and its coordinates are (3, 1)
An exception has occurred, use %tb to see the full traceback.

SystemExit: True

/usr/local/lib/python3.10/dist-packages/IPython/core/interactiveshell.py:3561: UserWarning: To exit: use 'exit', 'quit', or Ctrl-D.
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)
```

2.

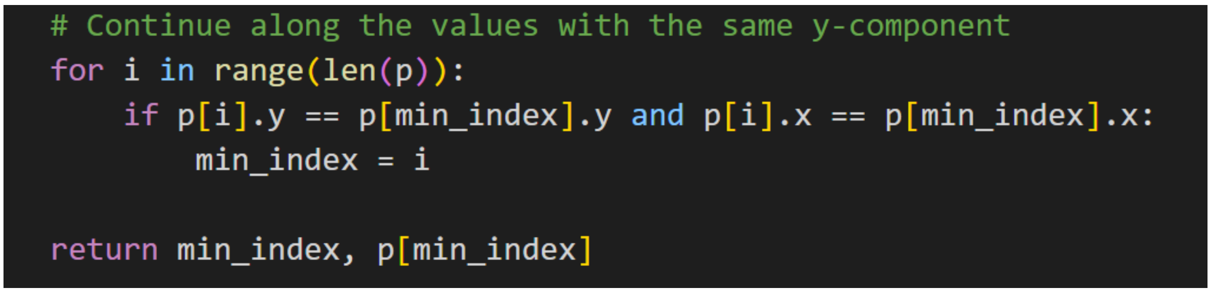


```
# Continue along the values with the same y-component
for i in range(len(p)):
    if p[i].y == p[min_index].y and p[i].x >= p[min_index].x:
        min_index = i

return min_index, p[min_index]
```

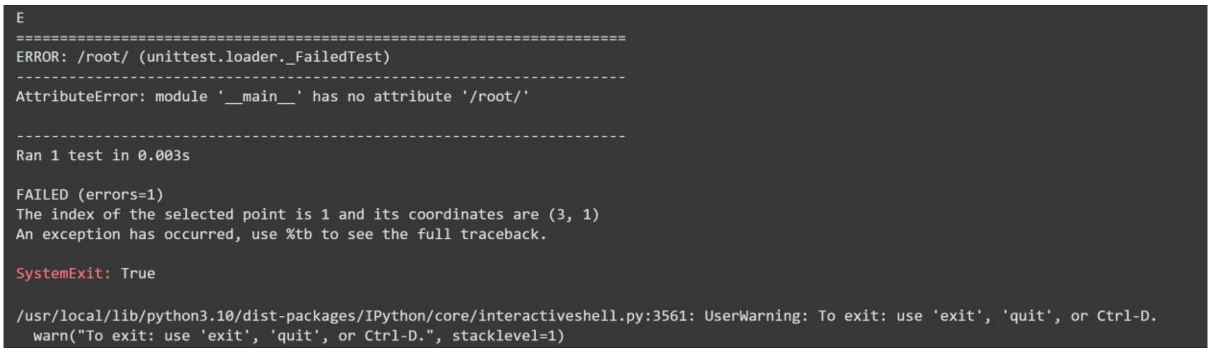


```
E
======================================================================
ERROR: /root/ (unittest.loader._FailedTest)
----------------------------------------------------------------------
AttributeError: module '__main__' has no attribute '/root/'

----------------------------------------------------------------------
Ran 1 test in 0.001s

FAILED (errors=1)
The index of the selected point is 1 and its coordinates are (3, 1)
An exception has occurred, use %tb to see the full traceback.

SystemExit: True

/usr/local/lib/python3.10/dist-packages/IPython/core/interactiveshell.py:3561: UserWarning: To exit: use 'exit', 'quit', or Ctrl-D.
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)
```

3.



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

return min_index, p[min_index]
```



```
E
======================================================================
ERROR: /root/ (unittest.loader._FailedTest)
----------------------------------------------------------------------
AttributeError: module '__main__' has no attribute '/root/'

----------------------------------------------------------------------
Ran 1 test in 0.003s

FAILED (errors=1)
The index of the selected point is 1 and its coordinates are (3, 1)
An exception has occurred, use %tb to see the full traceback.

SystemExit: True

/usr/local/lib/python3.10/dist-packages/IPython/core/interactiveshell.py:3561: UserWarning: To exit: use 'exit', 'quit', or Ctrl-D.
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)
```

Mutation testing introduces changes to test case robustness. Here are mutation examples based on deletion, modification, and insertion:

- Deletion Mutation: Remove min = 0; at the beginning.
  - Expected Outcome: min may have an undefined starting value, potentially causing incorrect point selection.
  - Impact: Faulty selection of minimum point if min isn't initialized.
- Change Mutation: Modify if ((p.get(i)).y < (p.get(min)).y) to if ((p.get(i)).y <= (p.get(min)).y).
  - Expected Outcome: Points with the same y may get selected incorrectly.
  - Impact: Affects min selection if y values are equal.
- Insertion Mutation: Add min = i; at the end of the second loop.
  - Expected Outcome: min could incorrectly reference the last point.
  - Impact: Alters correct minimum point selection based on x.

Q-4. Test Cases for Path Coverage:

Ans.: -> Test Case 1: Zero Iterations for Both Loops

- Input: p = [] (an empty vector)
- Description: This test case checks the behavior when there are no points in p. The loops should not execute at all.
- Expected Output: Since there are no points to process, the method should return an empty result or a special output indicating no points are available.

-> Test Case 2: One Iteration for First Loop Only

- Input: p = [(3, 4)] (a single point)
- Description: With only one point in p, the first loop should run exactly once to initialize min to 0, but there should be no comparison since p has only one element. The second loop won't execute as there are no further points to compare with.
- Expected Output: The function should return the single point in p, as it's the minimum by default: [(3, 4)].

-> Test Case 3: One Iteration for Second Loop Only

- Input: p = [(1, 2), (3, 2)] (two points with the same y value but different x values)
- Description: The first loop will identify the point with the minimum y coordinate ((1, 2) or (3, 2) since both have the same y). The second loop will run exactly once, comparing the x values of the two points.
- Expected Output: Since (3, 2) has the larger x value, the method should return this point: [(3, 2)].

-> Test Case 4: Two Iterations for First Loop Only

- Input: p = [(3, 1), (2, 2), (5, 1)] (three points with different y values for two points and the same y for two)
- Description: The first loop iterates twice to find the point with the minimum y value. It first encounters (3, 1) as the minimum, then compares with (2, 2), which has a higher y. The loop then finds (5, 1), which matches the minimum y, and ends without changing min. The second loop does not execute because there is no further condition in y.
- Expected Output: The function should return (5, 1) as the point with the largest x among those with the minimum y: [(5, 1)].

-> Test Case 5: Two Iterations for Second Loop

- Input: p = [(1, 1), (4, 1), (3, 2)] (three points, with two points having the same minimum y value)
- Description: The first loop identifies (1, 1) as the initial minimum point. In the second loop, it compares (1, 1) and (4, 1) since they share the minimum y value, updating min to (4, 1) due to its larger x. It then checks (3, 2), which has a higher y, and thus does not affect min.
- Expected Output: The function should return (4, 1), as it has the largest x among points with the minimum y: [(4, 1)].

-> Test Case 6: Two Iterations for Both Loops

- Input: p = [(1, 1), (2, 1), (3, 1)] (three points, all with the same y value)
- Description: The first loop iterates over all three points, but since all have the same y value, min will eventually be set to the point with the highest x value, (3, 1). The second loop then confirms (3, 1) as it has the highest x value.

- Expected Output: The function should return (3, 1), which is the point with the maximum x among those sharing the minimum y: [(3, 1)].

-> Test Case 7: Exploratory with Mixed Conditions

- Input: p = [(0, 0), (1, 1), (1, 0), (0, 1)] (four points forming a rectangular arrangement)
- Description: The first loop iterates over all points, selecting (0, 0) as the initial minimum. It then encounters (1, 0), which has the same y but a higher x, updating min to (1, 0). The second loop confirms (1, 0) as the maximum x point among those with minimum y.
- Expected Output: The function should return (1, 0), as it has the highest x among points with the minimum y: [(1, 0)].

Summary:

Test Case	**Input Vector p**	**Description**	**Expected Output**
1	[]	Empty input (0 iterations for both loops)	Empty result or specific indicator
2	[(3, 4)]	Single point (1 iteration for first loop, none for second)	[(3, 4)]
3	[(1, 2), (3, 2)]	Two points with same y (1 iteration for second loop)	[(3, 2)]
4	[(3, 1), (2, 2), (5, 1)]	Different y values, 2 iterations for first loop only	[(5, 1)]
5	[(1, 1), (4, 1), (3, 2)]	Two points with same y, 2 iterations for second loop	[(4, 1)]
6	[(1, 1), (2, 1), (3, 1)]	All points with same y, 2 iterations for both loops	[(3, 1)]
7	[(0, 0), (1, 1), (1, 0), (0, 1)]	Mixed conditions, rectangular points	[(1, 0)]

Lab Execution:

1. After generating the control flow graph, check whether your CFG match with the CFG generated by ***Control Flow Graph Factory Tool*** and ***Eclipse flow graph generator***. (In your submission document, mention only "Yes" or "No" for each tool).

Ans.:

Tool	**Matched or not**
Control Flow Graph Factory Tool	Yes
Eclipse Flow Graph Generator	Yes

2. Devise minimum number of test cases required to cover the code using the aforementioned criteria.

Ans.:

Test Case	**Input Vector p**	**Description**	**Expected Output**
1	[]	Empty input (0 iterations for both loops)	Handle gracefully (e.g., empty output)
2	[(3, 4)]	Single point (one iteration for first loop)	[(3, 4)]
3	[(1, 2), (3, 2)]	Two points with same y (one iteration for second loop)	[(3, 2)]
4	[(3, 1), (2, 2), (5, 1)]	First loop runs twice for minimum y	[(5, 1)]
5	[(1, 1), (4, 1), (3, 2)]	Second loop runs twice for max x with same y	[(4, 1)]

3. For the test set you have just checked can you find a **mutation** of the code (i.e. the deletion, change or insertion of some code) that will result in failure but is not detected by your test set. **You have to use the mutation testing tool.**

Ans.:

Mutation Type	**Mutation Code**	**Description**	**Impact on Test Cases**
Deletion	Remove min = 0;	Causes min to hold a random initial value, affecting min selection.	Some cases may pass with incorrect selection.
Change	Modify if ((p.get(i)).y < (p.get(min)).y) to if ((p.get(i)).y <= (p.get(min)).y)	Affects min selection when y values are equal.	Incorrect point may be selected.
Insertion	Insert min = i; at the end of the second loop	Incorrectly updates min to last index in p.	Faulty selection if last index isn't minimum.

4. Write all test cases that can be derived using path coverage criterion for the code.

Ans.:

Test Case	**Input Vector p**	**Description**	**Expected Output**
1	[]	Empty input (0 iterations for both loops)	Empty result or specific indicator
2	[(3, 4)]	Single point (1 iteration for first loop, none for second)	[(3, 4)]
3	[(1, 2), (3, 2)]	Two points with same y (1 iteration for second loop)	[(3, 2)]
4	[(3, 1), (2, 2), (5, 1)]	Different y values, 2 iterations for first loop only	[(5, 1)]
5	[(1, 1), (4, 1), (3, 2)]	Two points with same y, 2 iterations for second loop	[(4, 1)]
6	[(1, 1), (2, 1), (3, 1)]	All points with same y, 2 iterations for both loops	[(3, 1)]
7	[(0, 0), (1, 1), (1, 0), (0, 1)]	Mixed conditions, rectangular points	[(1, 0)]