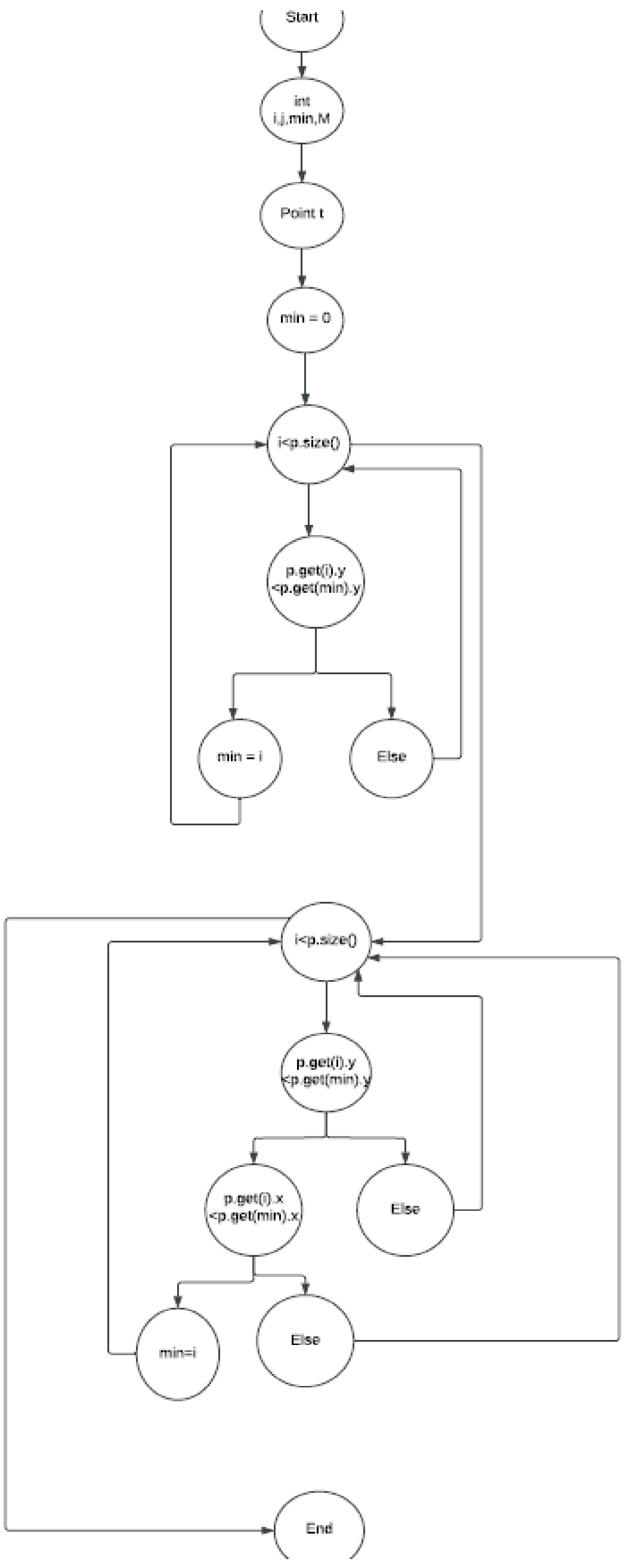
SE Lab 9

Student Name: Shrey Bavishi

Student ID : 202201478

```
Q1)

class Point:

    def __init__(self, x, y):

        self.x = x

        self.y = y

def findMinPoint(points):

    min_index = 0

    for i in range(1, len(points)):

        if points[i].y < points[min_index].y:

            min_index = i

    for i in range(len(points)):

        if points[i].y == points[min_index].y and points[i].x >
points[min_index].x:

            min_index = i

    return points[min_index]
```


Start
int i,j,min,M
Point t
min = 0
i<p.size()
p.get(i).y <p.get(min).y
min = i
Else
i<p.size()
p.get(i).y <p.get(min).y
p.get(i).x <p.get(min).x
Else
min=i
Else
End

Test Cases for ConvexHull Algorithm

1. Statement Coverage Test Cases

These test cases ensure that each statement in the code is executed at least once.

Input: p = [(2,3), (1,2), (3,4)]
Input: p = [(1,1), (2,1), (3,1)]

2. Branch Coverage Test Cases

These test cases ensure that each possible branch (True/False) from decision points is taken at least once.

- First Loop - True Branch
  Input: p = [(2,3), (1,1), (3,4)]
- First Loop - False Branch
  Input: p = [(1,3), (2,4), (3,5)]
- Second Loop - True Branch
  Input: p = [(1,2), (3,2), (2,2)]
- Second Loop - False Branch
  Input: p = [(1,1), (2,2), (3,3)]

3. Basic Condition Coverage Test Cases

These test cases ensure each basic condition in compound conditions is evaluated to both True and False.

- Testing y equality
  Input: p = [(1,2), (2,2), (3,3)]
- Testing x comparison

Input: p = [(1,2), (3,2), (2,2)]

- Testing both conditions

  Input: p = [(1,1), (2,1), (0,1)]

Q3)

```
[*] Start mutation process:
   - targets: point
   - tests: test_points
[*] 3 tests passed:
   - test_points [0.24341 s]
[*] Start mutants generation and execution:
   - [#   1] COI point:
--------------------------------------------------------------------------------
   6:
   7: def find_min_point(points):
   8:     min_index = 0
   9:     for i in range(1, len(points)):
- 10:         if points[i].y < points[min_index].y:
+ 10:         if not (points[i].y < points[min_index].y):
  11:             min_index = i
  12:     for i in range(len(points)):
  13:         if (points[i].y == points[min_index].y and points[i].x >
points[min_index].x):
  14:             min_index = i
--------------------------------------------------------------------------------
[0.15408 s] killed by
test_points.py::TestFindMinPointPathCoverage::testMultiplePoints
   - [#   2] COI point:
--------------------------------------------------------------------------------
   9:     for i in range(1, len(points)):
  10:         if points[i].y < points[min_index].y:
  11:             min_index = i
  12:     for i in range(len(points)):
- 13:         if (points[i].y == points[min_index].y and points[i].x >
points[min_index].x):
```

```
+ 13:        if not ((points[i].y == points[min_index].y and points[i].x >
points[min_index].x)):
  14:            min_index = i
  15:     return points[min_index]
--------------------------------------------------------------------------------
[0.14159 s] killed by
test_points.py::TestFindMinPointPathCoverage::testMultiplePointSamyY
   - [#   3] LCR point:
--------------------------------------------------------------------------------
   9:    for i in range(1, len(points)):
  10:        if points[i].y < points[min_index].y:
  11:            min_index = i
  12:     for i in range(len(points)):
- 13:        if (points[i].y == points[min_index].y and points[i].x >
points[min_index].x):
+ 13:        if (points[i].y == points[min_index].y or points[i].x > points[min_index].x):
  14:            min_index = i
  15:     return points[min_index]
--------------------------------------------------------------------------------
[0.15599 s] killed by
test_points.py::TestFindMinPointPathCoverage::testMultiplePointSamyY
   - [#   4] ROR point:
--------------------------------------------------------------------------------
   6:
   7: def find_min_point(points):
   8:    min_index = 0
   9:    for i in range(1, len(points)):
- 10:        if points[i].y < points[min_index].y:
+ 10:        if points[i].y > points[min_index].y:
  11:            min_index = i
  12:     for i in range(len(points)):
  13:        if (points[i].y == points[min_index].y and points[i].x >
points[min_index].x):
  14:            min_index = i
--------------------------------------------------------------------------------
[0.14234 s] killed by
test_points.py::TestFindMinPointPathCoverage::testMultiplePoints
   - [#   5] ROR point:
--------------------------------------------------------------------------------
```

```
   6:
   7: def find_min_point(points):
   8:     min_index = 0
   9:     for i in range(1, len(points)):
- 10:         if points[i].y < points[min_index].y:
+ 10:         if points[i].y <= points[min_index].y:
  11:             min_index = i
  12:     for i in range(len(points)):
  13:         if (points[i].y == points[min_index].y and points[i].x >
points[min_index].x):
  14:             min_index = i
--------------------------------------------------------------------------------
[0.11556 s] survived
   - [#   6] ROR point:
--------------------------------------------------------------------------------
   9:     for i in range(1, len(points)):
  10:         if points[i].y < points[min_index].y:
  11:             min_index = i
  12:     for i in range(len(points)):
- 13:         if (points[i].y == points[min_index].y and points[i].x >
points[min_index].x):
+ 13:         if (points[i].y != points[min_index].y and points[i].x >
points[min_index].x):
  14:             min_index = i
  15:     return points[min_index]
--------------------------------------------------------------------------------
[0.14255 s] killed by
test_points.py::TestFindMinPointPathCoverage::testMultiplePointSamyY
   - [#   7] ROR point:
--------------------------------------------------------------------------------
   9:     for i in range(1, len(points)):
  10:         if points[i].y < points[min_index].y:
  11:             min_index = i
  12:     for i in range(len(points)):
- 13:         if (points[i].y == points[min_index].y and points[i].x >
points[min_index].x):
+ 13:         if (points[i].y == points[min_index].y and points[i].x <
points[min_index].x):
  14:             min_index = i
  15:     return points[min_index]
--------------------------------------------------------------------------------
[0.14933 s] killed by
test_points.py::TestFindMinPointPathCoverage::testMultiplePointSamyY
   - [#   8] ROR point:
--------------------------------------------------------------------------------
   9:     for i in range(1, len(points)):
  10:         if points[i].y < points[min_index].y:
  11:             min_index = i
  12:     for i in range(len(points)):
- 13:         if (points[i].y == points[min_index].y and points[i].x >
points[min_index].x):
+ 13:         if (points[i].y == points[min_index].y and points[i].x >=
points[min_index].x):
  14:             min_index = i
  15:     return points[min_index]
--------------------------------------------------------------------------------
[0.11332 s] survived
[*] Mutation score [1.52260 s]: 75.0%
   - all: 8
   - killed: 6 (75.0%)
   - survived: 2 (25.0%)
   - incompetent: 0 (0.0%)
   - timeout: 0 (0.0%)
```

Q4)

```python
import unittest
from point import Point, findMinPoint

class TestFindMinPointPathCoverage(unittest.TestCase):

    def TestEmptyList(self):
        points = []
        with self.assertRaises(IndexError):
findMinPoint(points)

    def TestSinglePoint(self):
        points = [Point(2, 2)]
        result = findMinPoint(points)
        self.assertEqual(result, points[0])
    def testTwoUniquePoint(self):
        points = [Point(2, 1), Point(3, 2)]
        result = findMinPoint(points)
        self.assertEqual(result, points[0])
    def TestMultipleuniquePoint(self):
        points = [Point(1, 3), Point(2, 4), Point(3, 5)]
        result = findMinPoint(points)
        self.assertEqual(result, points[0])

    def testMultiplePointSamyY(self):
        points = [Point(1, 2), Point(3, 2), Point(2, 2)]
        result = findMinPoint(points)
        self.assertEqual(result, points[1])

    def testMultiplePoints(self):
        points = [Point(1, 2), Point(2, 2), Point(3, 1), Point(4, 1)]
        result = findMinPoint(points)
        self.assertEqual(result, points[3])

# Run the tests if this file is executed
if __name__ == "__main__":
    unittest.main()
```

```
Test Result with mut.py
Mutation score [1.52260 s]: 75.0%
   - all: 8
   - killed: 6 (75.0%)
   - survived: 2 (25.0%)
   - incompetent: 0 (0.0%)
   - timeout: 0 (0.0%)
```