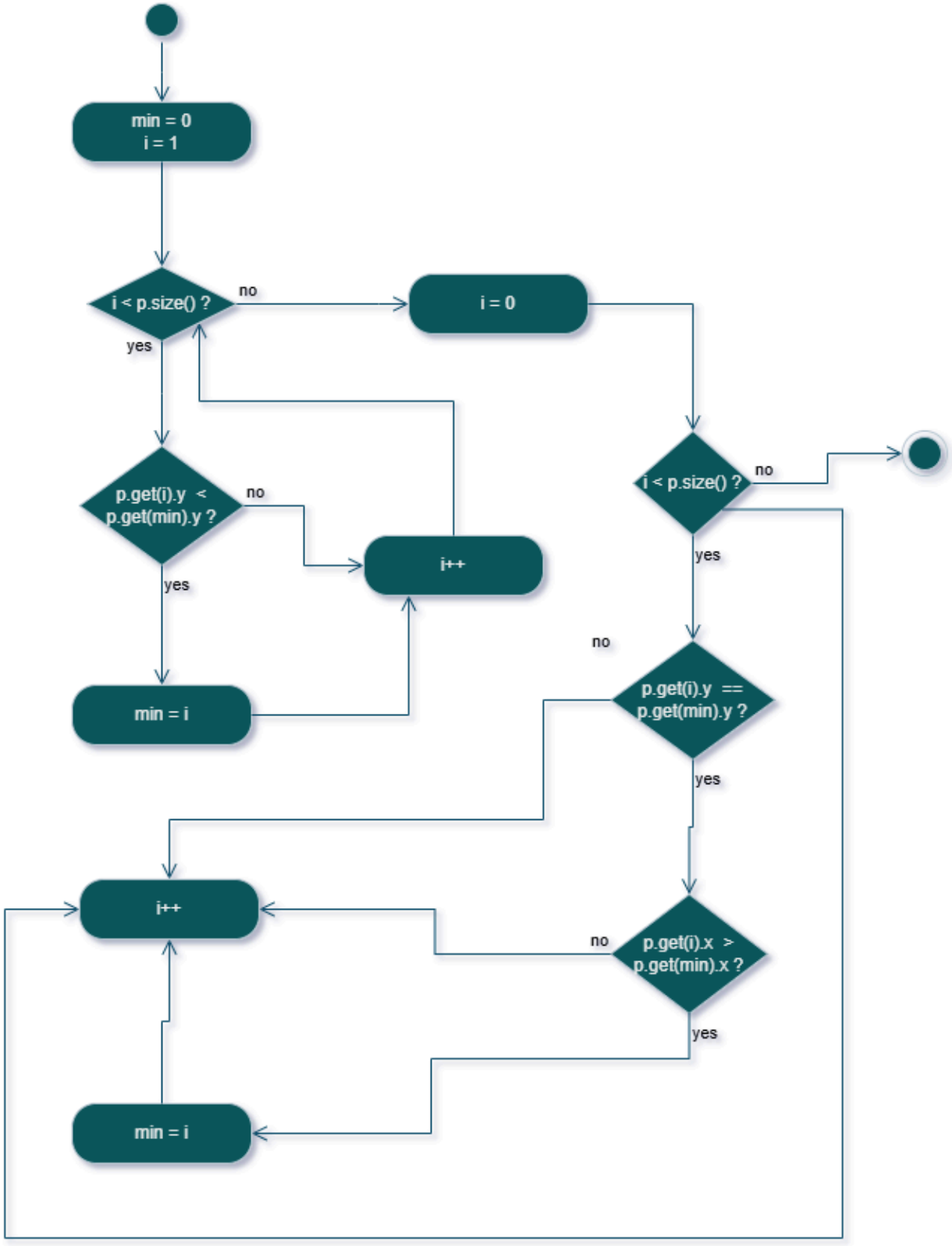
IT314 :- Software Engineering
Lab 9
Mutation Testing

Name :- Aryan Solanki
ID :- 202201239

Q.1.
Code Snippet :

```
Vector doGraham(Vector p) {
        int i,j,min,M;

        Point t;
        min = 0;

        // search for minimum:
        for(i=1; i < p.size(); ++i) {
            if( ((Point) p.get(i)).y <
                        ((Point) p.get(min)).y )
            {
                min = i;
            }
        }

        // continue along the values with same y component
        for(i=0; i < p.size(); ++i) {
            if(( ((Point) p.get(i)).y ==
                        ((Point) p.get(min)).y ) &&
                 (((Point) p.get(i)).x >
                        ((Point) p.get(min)).x ))
            {
                 min = i;
            }
        }
```

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.


min = 0
i = 1
i < p.size() ?
no
yes
i = 0
p.get(i).y < p.get(min).y ?
no
yes
i++
min = i
i < p.size() ?
no
yes
no
p.get(i).y == p.get(min).y ?
yes
i++
no
p.get(i).x > p.get(min).x ?
yes
min = i

2. Construct test sets for your flow graph that are adequate for the following criteria:

a. Statement Coverage:

TC1: p = [(1, 7), (2, 3), (8, 1)]

● Covers cases where p.get(i).y < p.get(min).y is true.

TC2: p = [(1, 1), (2, 1), (3,1), (4,1)]

● covers cases where p.get(i).y == p.get(min).y and p.get(i).x > p.get(min).x

b. Branch Coverage:

TC3: p = [(1, 2), (3, 3), (5, 9), (10, 15)]

● covers the false branch for condition p.get(i).y < p.get(min).y

TC4: p = [(10, 15), (5, 9), (3, 3), (1, 2)]

● covers the true branch for condition p.get(i).y < p.get(min).y

TC5: p = [(1, 1), (2, 1), (3, 1)]

● cover both true and false branches of p.get(i).y == p.get(min).y and p.get(i).x > p.get(min).x.

c. Basic Condition Coverage:

TC6: p = [(1, 2), (3, 3)]

● Covers p.get(i).y < p.get(min).y as false.

TC7: p = [(1, 5), (1, 3)]

● Covers p.get(i).y < p.get(min).y as true.

TC8: p = [(1, 1), (2, 1)]

● Covers p.get(i).y == p.get(min).y as true and p.get(i).x > p.get(min).x as true.

TC9: p = [(2, 2), (1, 1)]

● Covers p.get(i).y == p.get(min).y as false and p.get(i).x > p.get(min).x as false.

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.

☐ **Deletion Mutation:**
Remove min = i in the first if condition.

Mutation Code:

```
// Remove this line:
min = i;
if (((Point) p.get(i)).y < ((Point) p.get(min)).y)
{
        // min = i; <- this line is removed
}
```

☐ **Insertion Mutation:**
Add min = 0 at the beginning of the second loop.

Mutation Code:

```
for(i = 0; i < p.size(); ++i)
{
        min = 0;
        // Inserted line
        if (((Point) p.get(i)).y == ((Point) p.get(min)).y && ((Point)
p.get(i)).x < ((Point) p.get(min)).x)
        {
                min = i;
        }
}
```

☐ **Modification Mutation:**

Change p[i].y < p[min].y to p[i].y > p[min].y in the first if condition.

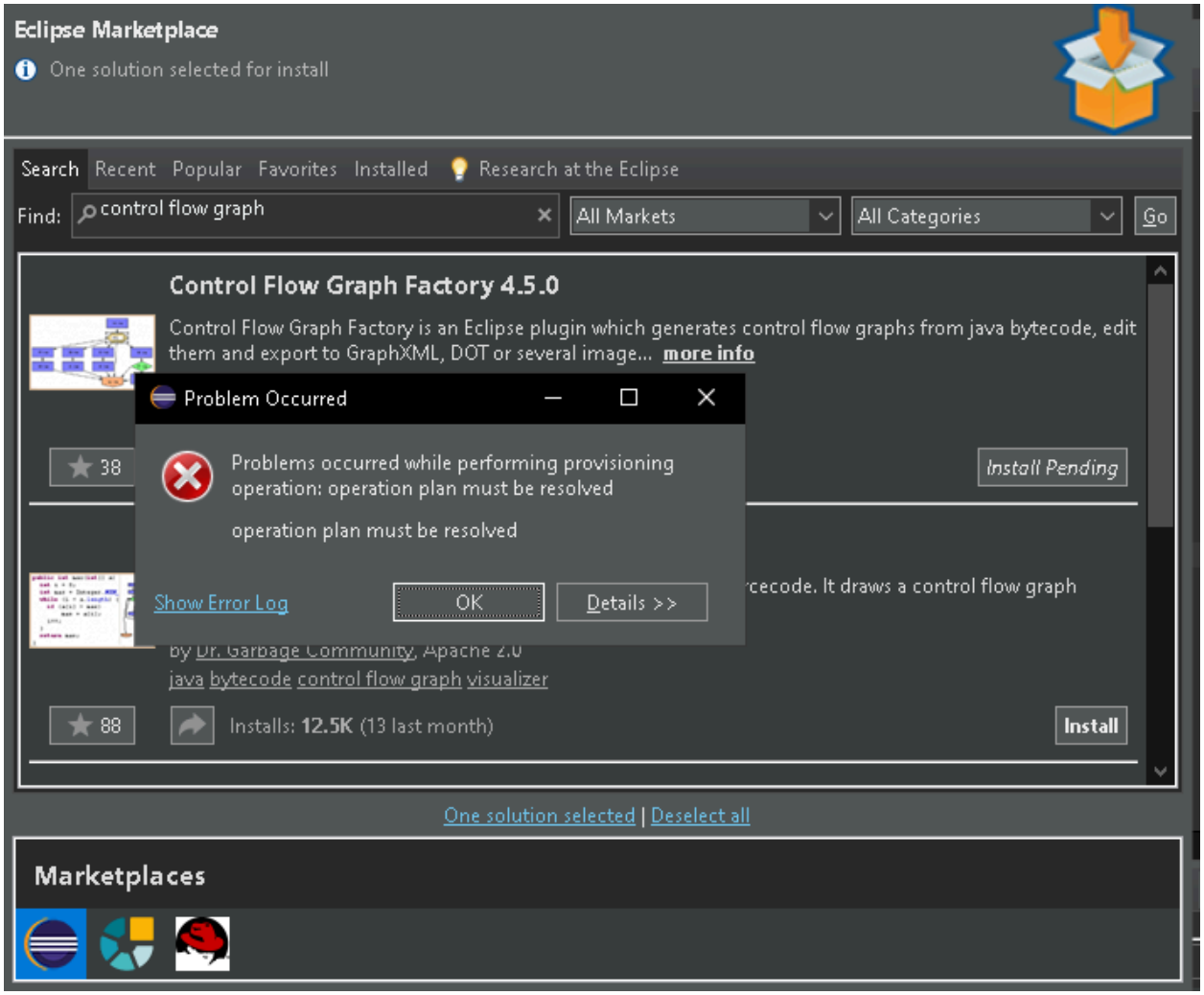
Mutation Code:

```
if (((Point) p.get(i)).y > ((Point) p.get(min)).y)
{
        // Modified from < to >
        min = i;
}
```

4. Create a test set that satisfies the path coverage criterion where every loop is explored at least zero, one or two times.

```
import unittest
from point import Point, find_min_point

class TestFindMinPointPathCoverage(unittest.TestCase):

    def test_no_points(self):
        points = []
        with self.assertRaises(IndexError):  # Expect an IndexError due to
empty list
            find_min_point(points)

    def test_single_point(self):
        points = [Point(0, 0)]
        result = find_min_point(points)
        self.assertEqual(result, points[0])  # Expect the point (0, 0)

    def test_two_points_unique_min(self):
        points = [Point(1, 2), Point(2, 3)]
        result = find_min_point(points)
        self.assertEqual(result, points[0])  # Expect the point (1, 2)

    def test_multiple_points_unique_min(self):
        points = [Point(1, 4), Point(2, 3), Point(0, 1)]
```

```
        result = find_min_point(points)
        self.assertEqual(result, points[2])  # Expect the point (0, 1)

    def test_multiple_points_same_y(self):
        points = [Point(1, 2), Point(3, 2), Point(2, 2)]
        result = find_min_point(points)
        self.assertEqual(result, points[1])  # Expect the point (3, 2)

    def test_multiple_points_minimum_y_ties(self):
        points = [Point(1, 2), Point(2, 2), Point(3, 1), Point(4, 1)]
        result = find_min_point(points)
        self.assertEqual(result, points[3])  # Expect the point (4, 1)

if __name__ == "__main__":
    unittest.main()
```



There was some error in the eclipse such that I am not able to install the Control flow graph factory.