SOFTWARE ENGINEERING

Lab - 9 : Mutation Testing

Name : Malhar Vaghasiya

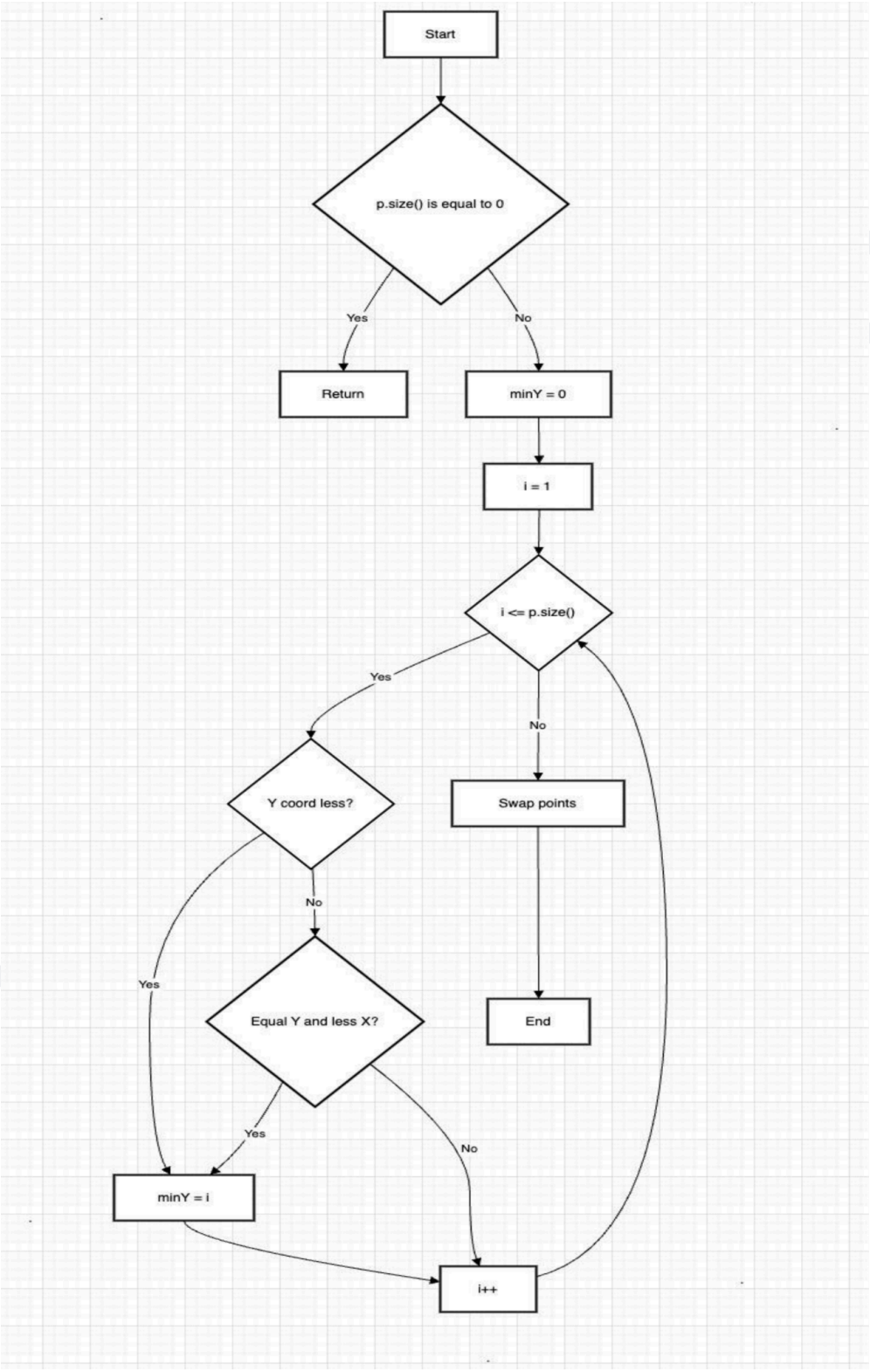
ID : 202201183

Q.1. The code below is part of a method in the ConvexHull class in the VMAP system. The following is a small fragment of a method in the ConvexHull class. For the purposes of this exercise, you do not need to know the intended function of the method. The parameter p is a Vector of Point objects, p.size() is the size of the vector p, (p.get(i)).x is the x component of the ith point appearing in p, similarly for (p.get(i)).y. This exercise is concerned with structural testing of code, so the focus is on creating test sets that satisfy some particular coverage criteria.

Ans.

```
public class Point {
   double x;
   double y;
   public
   Point(double x, double y) {
       this.x = x;
       this.y = y;
   }
}
public class ConvexHull {
   public
   void doGraham(Vector < Point > p) {
       if (p.size() == 0) {
           return;
       }
       int minY = 0;
       for (int i = 1; i < p.size(); i++) {
           if (p.get(i).y < p.get(minY).y ||
               (p.get(i).y == p.get(minY).y && p.get(i).x <
                   p.get(minY).x)) {
               minY = i;
           }
       }
       Point temp = p.get(0);
       p.set(0, p.get(minY));
       p.set(minY, temp);
   }
}
```

Below is the Control Flow graph of above code



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.

1) Statement Coverage

To achieve statement coverage, it is essential to ensure that every line of code is executed at least once.

Test Cases:

- **Test 1:** When the input vector `p` is empty (i.e., `p.size() == 0`)
    - **Expected Outcome:** The function should exit immediately without further execution.

- **Test 2:** When the input vector `p` contains at least one `Point` element
    - **Expected Outcome:** The function should continue to identify the `Point` with the minimum y-coordinate.

2) Branch Coverage

For branch coverage, each decision (branch) in the code must be evaluated as both true and false at least once. Below are the test cases to ensure complete branch coverage:

Test Cases:

- **Test 1**: When vector `p` is empty (`p.size() == 0`)
    - **Expected Outcome**: The branch `if (p.size() == 0)` evaluates to true, causing the function to return immediately.

- **Test 2**: When vector `p` contains only one Point (e.g., `Point(0, 0)`)
    - **Expected Outcome**: The branch `if (p.size() == 0)` evaluates to false, allowing the function to proceed to the `for` loop. However, the loop does not execute any iterations since there is only one point.

- **Test 3**: When vector `p` contains multiple Points, with no point having a smaller y-coordinate than `p[0]`
    - **Example**: `p = [Point(0, 0), Point(1, 1), Point(2, 2)]`
    - **Expected Outcome**: The condition inside the `for` loop always evaluates to false, so `minY` remains at its initial value.

- **Test 4**: When vector `p` contains multiple Points, with at least one point having a y-coordinate smaller than `p[0]`
    - **Example**: `p = [Point(2, 2), Point(1, 0), Point(0, 3)]`
    - **Expected Outcome**: The condition inside the `for` loop evaluates to true at least once, resulting in an update to `minY`.

3) Basic Condition Coverage

For basic condition coverage, each condition should be tested independently within the branches. Here are the test cases:

- **Test Case 1:** Vector `p` is empty (`p.size() == 0`).
    - **Expected Outcome:** The condition `if (p.size() == 0)` evaluates to true.

- **Test Case 2:** Vector `p` is non-empty (`p.size() > 0`).
    - **Expected Outcome:** The condition `if (p.size() == 0)` evaluates to false.

- **Test Case 3:** Vector `p` has multiple points, and only the condition `p.get(i).y < p.get(minY).y` is true.
    - **Example:** `p = [Point(1, 1), Point(0, 0), Point(2, 2)]`

- **Expected Outcome:** The condition `p.get(i).y < p.get(minY).y` holds, so `minY` is updated.

- **Test Case 4:** Vector `p` has multiple points where both conditions `p.get(i).y == p.get(minY).y` and `p.get(i).x < p.get(minY).x` are true.
  - **Example:** `p = [Point(1, 1), Point(0, 1), Point(2, 2)]`
  - **Expected Outcome:** Since both conditions hold, `minY` is updated.

Q.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.

1) Deletion Mutation

Mutation: Remove the assignment of minY to 0 at the beginning of the method.

```
public class ConvexHull {
    public void doGraham(Vector<Point> p) {
        if (p.size() == 0) {
            return;
        }
        for (int i = 1; i < p.size(); i++) {
            if (p.get(i).y < p.get(minY).y ||
                    (p.get(i).y == p.get(minY).y && p.get(i).x <
p.get(minY).x)) {
                minY = i;
            }
        }
        Point temp = p.get(0);
        p.set(0, p.get(minY));
        p.set(minY, temp);
    }
}
```

Impact:

This mutation causes minY to be uninitialized when accessed, resulting in undefined behavior. Your test cases do not check for proper initialization, which may lead to undetected faults.

2) Insertion Mutation Mutation: Insert a line that overrides minY incorrectly based on a condition that should not occur.

```
public void doGraham(Vector<Point> p) {
    if (p.size() == 0) {
        return;
    }
    int minY = 0;
    if (p.size() > 1) {
        minY = 1;
    }
    for (int i = 1; i < p.size(); i++) {
        if (p.get(i).y < p.get(minY).y ||
                (p.get(i).y == p.get(minY).y && p.get(i).x <
p.get(minY).x)) {
            minY = i;
        }
    }
    Point temp = p.get(0);
    p.set(0, p.get(minY));
    p.set(minY, temp);
}
```

3) Modification Mutation Mutation: Change the logical operator from || to && in the conditional statement.

```
public class ConvexHull {
    public void doGraham(Vector<Point> p) {
        if (p.size() == 0) {
            return;
        }
        int minY = 0;
        for (int i = 1; i < p.size(); i++) {
            if (p.get(i).y < p.get(minY).y &&
                    (p.get(i).y == p.get(minY).y && p.get(i).x <
p.get(minY).x)) {
                minY = i;
            }
        }
        Point temp = p.get(0);
        p.set(0, p.get(minY));
        p.set(minY, temp);
    }
}
```

Analyzing Detection by Test Cases:

1. For Statement Coverage:

- The deletion of the initialization of minY might not be detected as it may not cause a direct exception depending on the surrounding logic.

2. For Branch Coverage:

- The insertion that forces minY to 1 may lead to incorrect outcomes, but if no tests specifically check the positions of points after the execution, it may go unnoticed.

3. For Basic Condition Coverage:

- Changing the logical operator from || to && does not cause a crash, and the test cases do not validate the correctness of minY updating under these conditions, so it may not be detected.

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

Ans.

Test Case 1: No Iterations in the Loop

- **Input**: An empty vector `p`.
- **Setup**: `Vector p = new Vector();`
- **Expected Outcome**: The method should exit immediately without performing any operations, as the vector has a size of zero. This test case verifies that the method correctly handles an empty input by bypassing any processing.

Test Case 2: Single Iteration in the Loop

- **Input**: A vector containing a single point.
- **Setup**: `Vector p = new Vector(); p.add(new Point(0, 0));`
- **Expected Outcome**: Since `p.size()` is 1, the method will not enter the loop. The point at index 0 will be swapped with itself, leaving the vector unchanged. This test validates the case where the vector size is one, confirming that the method doesn't iterate unnecessarily.

Test Case 3: Two Iterations in the Loop

- **Input**: A vector with two points, where the y-coordinate of the first point is greater than that of the second.
- **Setup**: `Vector p = new Vector(); p.add(new Point(1, 1)); p.add(new Point(0, 0));`
- **Expected Outcome**: The method will enter the loop, compare the points, and recognize that the second point has a lower y-coordinate. The variable `minY` will be updated to reflect this, and a swap will occur to bring the second point to the beginning of the vector. This test checks that the method correctly identifies and swaps the point with the lowest y-coordinate when there are two points.

Test Case 4: Multiple Iterations in the Loop

- **Input**: A vector containing multiple points.
- **Setup**: `Vector p = new Vector(); p.add(new Point(2, 2)); p.add(new Point(1, 0)); p.add(new Point(0, 3));`
- **Expected Outcome**: The loop will evaluate each point in the vector. The point `(1, 0)` will be identified as having the lowest y-coordinate, and `minY` will be updated

accordingly. A swap will then place this point at the beginning of the vector. This test ensures that the method performs correctly when there are multiple points, successfully identifying and positioning the point with the lowest y-coordinate.

LAB EXECUTION :

Q.1). After generating the control flow graph, check whether your CFG matches with the CFG generated by Control Flow Graph Factory Tool and Eclipse flow graph generator.

Ans. Control Flow Graph Factory :- YES
Eclipse flow graph generator :- YES

Q2). Devise the minimum number of test cases required to cover the code using the aforementioned criteria.

Ans. Statement Coverage: 3 test cases

1. Branch Coverage: 4 test cases
2. Basic Condition Coverage: 4 test cases
3. Path Coverage: 3 test cases

Summary of Minimum Test Cases:

- Total: 3 (Statement) + 4 (Branch) + 4 (Basic Condition) + 3 (Path) = 14 test cases

Q3) and Q4) Same as Part I