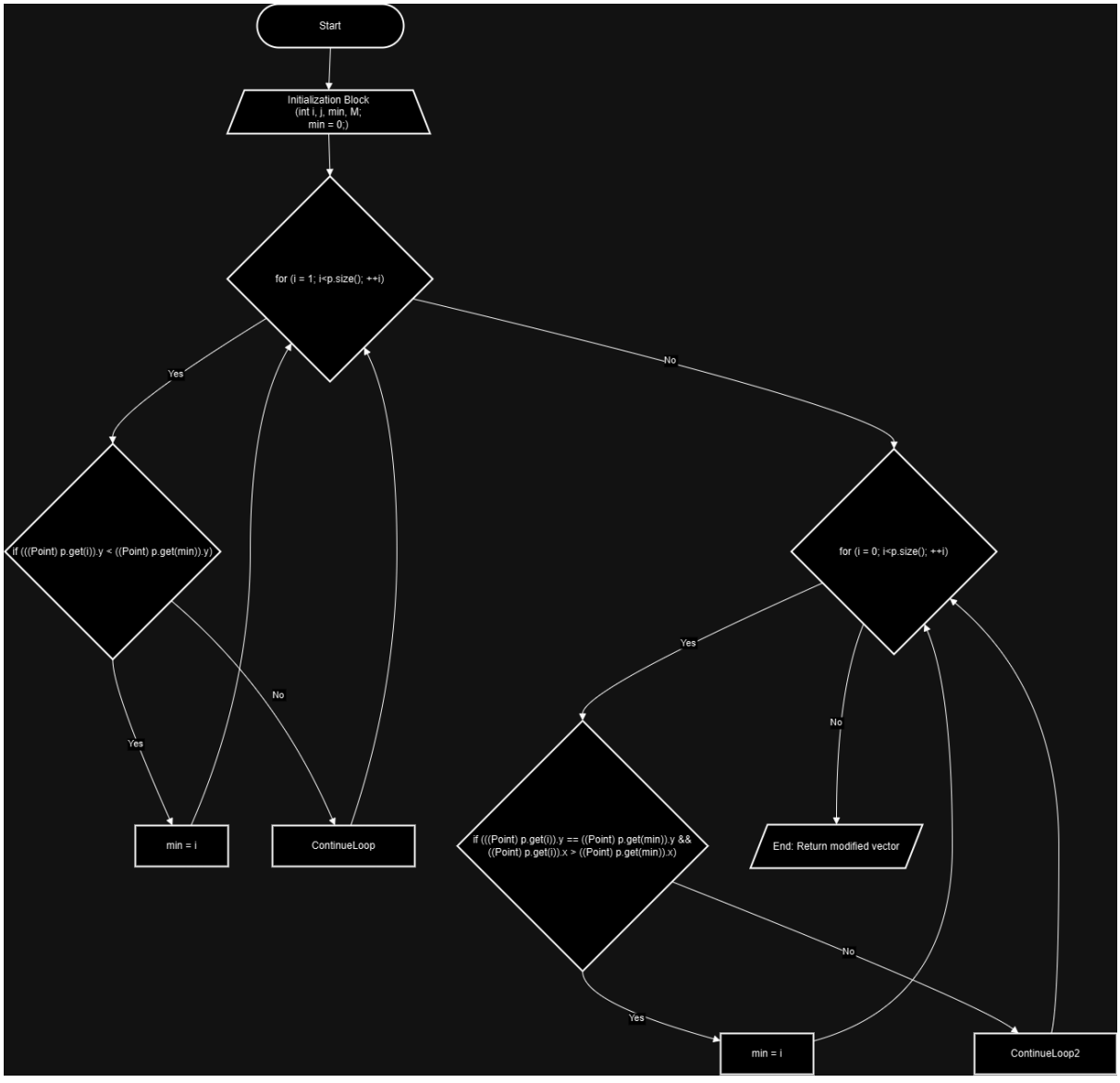
IT314 Lab 9

Mutation Testing

-202201441

Het Panchotiya

1. Convert the code comprising the beginning of the doGraham method into a control flow graph (CFG).



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.

- Statement Coverage requires that every statement in the code is executed at least once.
- Branch Coverage requires that every possible branch taken at least once
- Basic Condition Coverage requires that each basic condition in every decision is evaluated to both true and false.

Minimum Test Cases Which will cover the criteria:

- ✓ p = [(x=2, y=5), (x=1, y=4), (x=3, y=4)]
- ✓ p = [(x=2, y=5), (x=1, y=6), (x=3, y=6)]
- ✓ p = [(x=2, y=5), (x=1, y=6), (x=0, y=6)]

3. For the test set you have just checked can you find a mutation of the code that will result in failure but is not detected by your test set.

Original code:

```
def doGraham(points):
    min_index = 0

    for i in range(1, len(points)):
        if points[i].y < points[min_index].y or (points[i].y == points[min_index].y and points[i].x < points[min_index].x):
            min_index = i

    return points[min_index]
```

Deletion:

```
def doGraham(points):
    min_index = 0

    for i in range(1, len(points)):
        if points[i].y < points[min_index].y or (points[i].y == points[min_index].y and points[i].x < points[min_index].x)

    return points[min_index]
```

This will always give the point which is in first index as output.

Insertion:

```
def doGraham(points):
    min_index = 0

    for i in range(1, len(points)):
        if points[i].y < points[min_index].y or (points[i].y == points[min_index].y and points[i].x < points[min_index].x):
            min_index = i
            i++

    return points[min_index]
```

We are incrementing i by 2 instead of one each time, so if the real correct answer is at an even index, our program will give the wrong output.

Modification:

```
def doGraham(points):
    min_index = 0

    for i in range(1, len(points)):
        if points[i].y > points[min_index].y or (points[i].y == points[min_index].y and points[i].x < points[min_index].x):
            min_index = i

    return points[min_index]
```

We have changed the less than sign to greater than sign, so it gets the maximum y instead of minimum y.

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

- ✓ p = [(x=0, y=0)] (zero times)
- ✓ p = [(x=2, y=5), (x=1, y=1), (x=0, y=2)] (1 time)
- ✓ p = [(x=1, y=1), (x=0, y=1), (x=2, y=3)] (2 times)