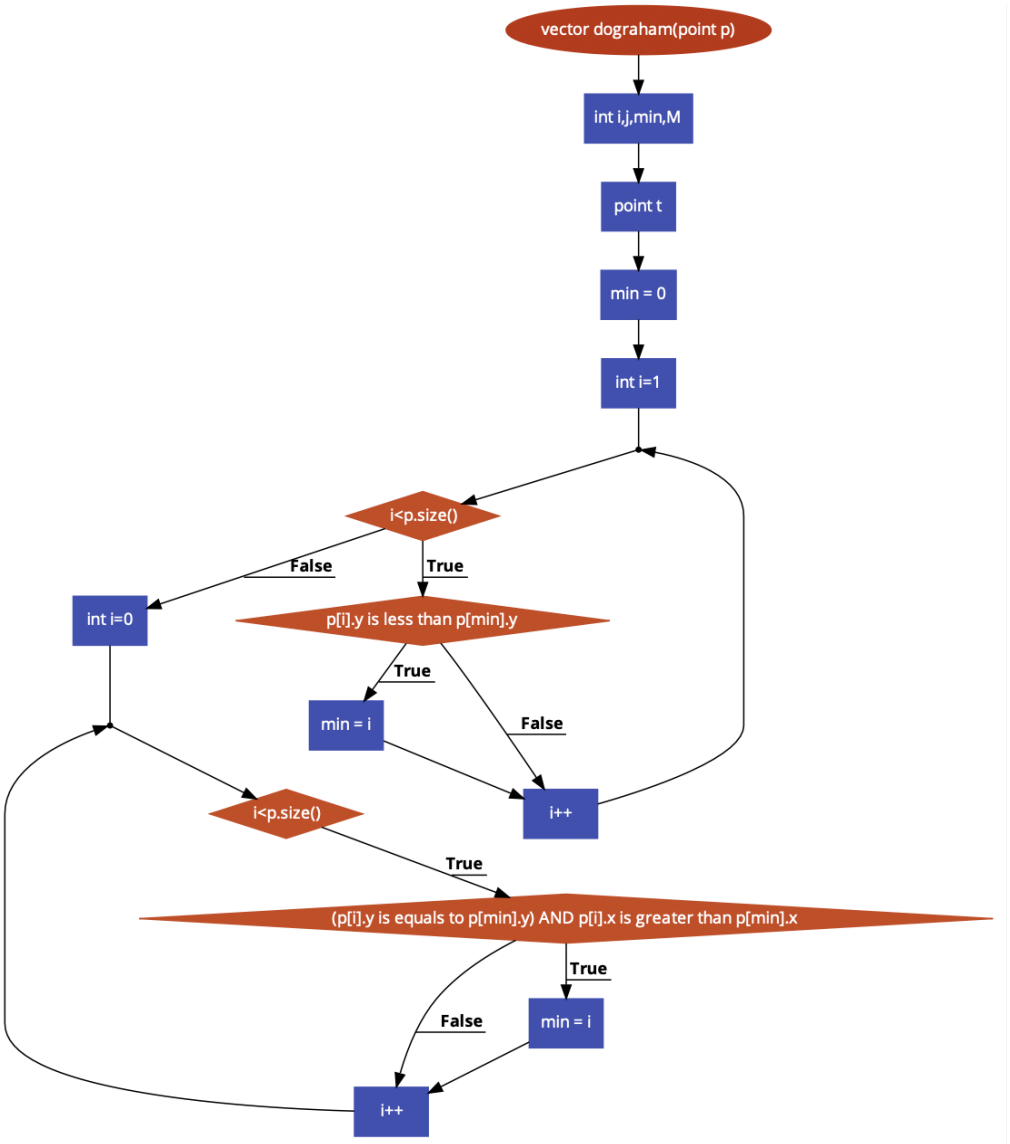
Software Engineering
(IT-314)

Name - Mihir Moolchandani
ID - 202201088

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:



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

Test Cases for Statement Coverage:

Statement 1: Declaration of variables
Statement 2: Start of the first for loop
Statement 3: Condition in the first loop
Statement 4: Update min in the first loop
Statement 5: Start of the second for loop
Statement 6: Condition in the second loop
Statement 7: Update min in the second loop

Test Case Number	Test Case	Statements Covered

Test Case 1 (Single Point)	{point(1, 0)}	1, 5, 6
Test Case 2 (Different y values)	{point(1, 4), point(3, 1), point(5, 5)}	1, 2, 3, 4, 5, 6
Test Case 3 (Same y values but different x values)	{point(3, 5), point(2, 5), point(7, 5)}	1, 2, 3, 5, 6, 7,

Test Cases for Branch Coverage:

Branch 1: First if condition inside the first for loop:

True: The current point has a smaller y than the current minimum y.

False: The current point's y is not smaller than the current minimum y.

Branch 2: Second if condition inside the second for loop:

True: The current point has the same y as the current minimum and a larger x.

False: The current point's y is different from the current minimum or the x is not larger.

Test Case Number	**Test Case**	**Branch Covered**
Test Case 1 (Single Point)	{point(0, 1)}	1 (False), 2(False)
Test Case 2 (Different y values)	{point(1, 4), point(3, 1), point(5, 5)}	1 (True), 1(False), 2(False)
Test Case 3 (Same y values but different x values)	{point(3, 5), point(2, 5), point(7, 5)}	1 (False) , 2 (True), 2(False)

Test Cases for Branch Condition Coverage:

Condition 1: y < min.y
Condition 2: y == min.y
Condition 3: x > min.x

Test Case Number	**Test Case**	**Condition Covered**
Test Case 1 (Single Point)	{point(0, 1)}	2 (True), 3 (False)
Test Case 2 (Different y values)	{point(1, 4), point(3, 1), point(5, 5)}	1 (True), 1(False), 2(False)
Test Case 3 (Same y values but different x values)	{point(3, 5), point(2, 5), point(7, 5)}	1 (False) , 2 (True), 2(False), 3 (True), 3(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.
Ans:
Possible Mutations:
Mutation 1: Change ((Point) p.get(i)).y < ((Point) p.get(min)).y to ((Point) p.get(i)).y <= ((Point) p.get(min)).y.
Mutation 2: Change ((Point) p.get(i)).y == ((Point) p.get(min)).y to ((Point) p.get(i)).y != ((Point) p.get(min)).y.
Mutation 3: Change ((Point) p.get(i)).x > ((Point) p.get(min)).x to ((Point) p.get(i)).x >= ((Point) p.get(min)).x.

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 Number	**Test Cases**	**Iterations**
Test Case 1	{}	Loop 1 -> 0 Loop2 -> 0
Test Case 2	{point(1, 0)}	Loop 1 -> 0 Loop2 -> 1
Test Case 3	{point(0, 1), point(1, 0)}	Loop 1 -> 1 Loop2 -> 2
Test Case 4	{point(2, 2), point(6, 7), point(1, 0)	Loop 1 -> 2 Loop2 -> 3

LAB EXERCISE

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. (In your submission document, mention only "Yes" or "No" for each tool).
Ans:

- Control Flow Graph Factory Tool: Yes
- Eclipse Flow Graph Generator: Yes

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

Test Case Number	**Test Case**	**Coverage**
Test Case 1	[]	Covers no-iteration paths
Test Case 2	[(1, 1)]	Covers single iteration in second loop
Test Case 3	[(2, 3), (1, 1)]	Covers true branch in first loop
Test Case 4	[(1, 1), (2, 1)]	Covers both conditions true in second loop
Test Case 5	[(3, 2), (1, 1), (2, 1)]	Covers both true/false conditions and all branches

3. This part of the exercise is very tricky and interesting. The test cases that you have derived in Step 2 are then used to identify the fault when you make some modifications in the code. Here, you need to insert/delete/modify a piece of code that will result in failure but it is not detected by your test set – derived in Step 2. Write/identify a mutation code for each of the three operations separately, i.e., by deleting the code, by inserting the code, by modifying the code.
Ans:

Mutation Type	**Mutation**	**Reason**
Deletion	Delete min = i; in the second loop's if condition	Test cases don't check min updates at each loop iteration
Insertion	Insert condition && (p.get(i)).x < 0 in the first if	Test cases don't include points with negative x values
Modification	Change < to <= in the first loop's if condition	Test cases don't validate the exact behavior for points

		with equal y values

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

Path Number	**Test Cases**	**First Loop Condition**	**Second Loop Condition**
Path 1	[]	N/A (0 iterations)	N/A (0 iterations)
Path 2	[(1, 1)]	N/A (0 iterations)	False
Path 3	[(2, 3), (1, 1)]	True	False
Path 4	[(1, 1), (2, 1)]	False	True
Path 5	[(3, 2), (1, 1), (2, 1)]	True, False	True, False
Path 6	[(1, 1), (3, 2), (4, 3)]	False	False

.