Daanish Husain

Steve Lai

Anjani Mallampati

Paul Diaz

CS 187: Software Quality Engineering Team 09

Assignment #3

**Part I Progress Bar**

1. **Progress Bar: Code**

import java.util.\*;

public class ProgressBar {

public static void main(String[] args) {

System.out.println("(0, 0, 0) : " + (solveProgressBar(0, 0, 0)? "black" :

"white"));

System.out.println("(12, 55, 55) : " + (solveProgressBar(12, 55, 55)?

"black" : "white"));

System.out.println("(13, 55, 55) : " + (solveProgressBar(13, 55, 55)?

"black" : "white"));

System.out.println("(60, 90, 90) : " + (solveProgressBar(60, 90, 90)?

"black" : "white"));

System.out.println("(26, 54, 19) : " + (solveProgressBar(26, 54, 19)?

"black" : "white"));

}

public static boolean solveProgressBar(double P, double X, double Y) {

if (P == 0) {

return false;

} else {

double xVal = X - 50;

double yVal = Y - 50;

double powX = Math.pow(xVal, 2);

double powY = Math.pow(yVal, 2);

double pyth = Math.sqrt(powX + powY); //pythagorean theorem

if (pyth > 50) {

return false;

}

if (xVal == 0 && yVal == 0) {

return true;

}

double theta = Math.atan2(yVal, xVal);

double twoPi = 2 \* Math.PI;

if (theta < 0) {

theta += twoPi;

}

double percent = (twoPi - theta) / (twoPi);

percent += 0.25;

if (percent >= 1) {

percent -= 1;

}

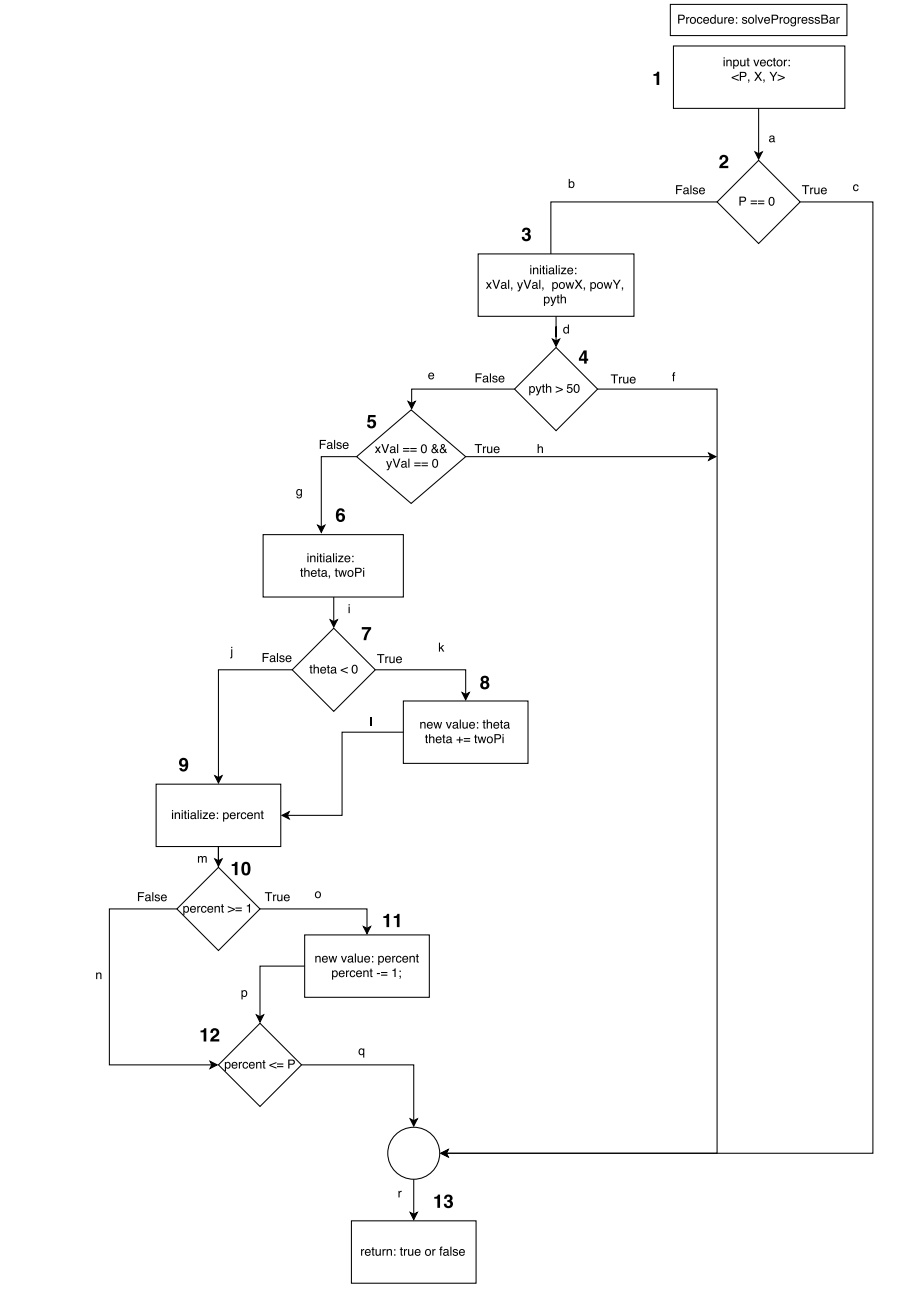
return percent <= P;

}

}

}

1. **Progress Bar: Control Flow Graph**



1. **Progress Bar: Test Cases & Output**
2. Input: (0,0,0)

Output : white

Generated Path: 1, 2(T), 13

Statement Coverage: 1, 2, 13

Branch Coverage: a, c, r

1. Input: (12,55,55)

Output : black

Generated Path: 1, 2(F), 3, 4(F), 5(F), 6, 7(F), 9, 10(T), 11, 12, 13

Statement Coverage: 1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13

Branch Coverage: a, b, d, e, g, i, j, m, o, p, q, r

1. Input: (13,55,55)

Output : black

Generated Path: 1, 2(F), 3, 4(F), 5(F), 6, 7(F), 9, 10(T), 11, 12, 13

Statement Coverage: 1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13

Branch Coverage: a, b, d, e, g, i, j, m, o, p, q, r

1. Input: (60,90,90)

Output : white

Generated Path: 1, 2(F), 3, 4(T), 13

Statement Coverage: 1, 2, 3, 4, 13

Branch Coverage: a, b, d, f, r

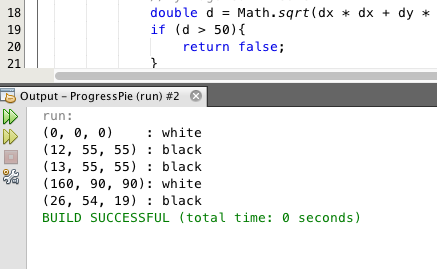
1. Input: (26,54,19)

Output : black

Generated Path: 1, 2(F), 3, 4(F), 5(F), 6, 7(T), 8, 9, 10(F), 12(T), 13

Statement Coverage: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13

Branch Coverage: a, b, d, e, g, i, k, l, m, n, q, r



1. **Progress Bar: Coverage**

No, the test cases do not achieve 100% statement and branch coverage. There needs to be a test case that follows the path 1, 2(F), 3, 4(F), 5(F), 6, 7(T), 8, 9, 10(T), 11, 12, 13 in order for 100% statement coverage. Also, there needs to be another test case that covers h in order to achieve 100% branch coverage. The path that the additional test case must go through to cover h is 1, 2(F), 3, 4(F), 5(F), 13.

**Part II Tower of Hanoi**

1. **Tower of Hanoi: Pseudocode**

There are three poles: start, middle, and end.

If there is only one disk at start (num = 1), move it to end.

If there is more than one disk at start,

1. Move the top (n-1) disk from start to middle.
2. Move the next disk from start to end.
3. Move the (n-1) disk at middle to end.
4. **Tower of Hanoi: Code**

//precondition: num > 0

public void towerOfHanoi(int num, String start, String middle, String end) {

if(num == 1) {

System.out.println(start + “ goes to” + end);

} else {

towerOfHanoi(num-1, start, end, middle);

System.out.println(start + “goes to” + end);

towerOfHanoi(num-1, middle, start, end);

}

}

**Tower of Hanoi: Control Flow Graph**

