

## Assignment 01 - Random Walk Algorithm

## Implemented Functions marked as “FIXME” in RandomWalk.java class file

- move(int dx,int dy)
- randomWalk(int m)
- distance()
- Main method in RandomWalk.java for custom test cases

```
INFO6205 - RandomWalk.java
INFO6205 ~src/main/java/edu/neu/coe/info6205/randomwalk/RandomWalk.java
INFO6205 - RandomWalkTest.java

1  ...
2
3  package edu.neu.coe.info6205.randomwalk;
4
5  import java.util.Random;
6
7
8  12 usages
9  public class RandomWalk {
10
11    3 usages
12    private int x = 0;
13
14    3 usages
15    private int y = 0;
16
17    2 usages
18    private final Random random = new Random();
19
20    /**
21     * Private method to move the current position, that's to say the drunkard moves
22     * @param dx the distance he moves in the x direction
23     * @param dy the distance he moves in the y direction
24     */
25
26    1 usage
27    private void move(int dx, int dy) {
28        this.x+=dx;
29        this.y+=dy;
30    }
31
32    /**
33     * Perform a random walk of m steps
34     *
35     * @param m the number of steps the drunkard takes
36     */
37
38    1 usage
39    private void randomWalk(int m) {
40        //We are utilizing randomMove function which already moves the coordinates
41        for (int i = 0; i < m; i++) {
42            ...
43        }
44    }
45
46    /**
47     * Compute the distance from the origin to the current position
48     *
49     * @return the distance
50     */
51
52    1 usage
53    private int distance() {
54        return (int) Math.sqrt(x*x+y*y);
55    }
56
57    /**
58     * Main method to run some tests
59     *
60     * @param args command line arguments
61     */
62
63    public static void main(String[] args) {
64        RandomWalk walk = new RandomWalk();
65        walk.randomWalk(1000);
66        System.out.println("Distance: " + walk.distance());
67    }
68
69}
```

IntelliJ IDEA

INFO6205 - RandomWalk.java

```
INFO6205 | src | main | java | edu | neu | coe | info6205 | randomwalk | RandomWalk | randomWalk
```

```
INFO6205 ~|/Documents/GitHub/INFO6205
```

```
Project | INFO6205 | .idea | src | main | java | edu.neu.coe.info6205 | bsearchtree | bgs | codelength | coupling | dynamicProgramming.coins | equable | functions | graphs | greedy | hashtable | lab_1 | life | pq | randomwalk | RandomWalkTest.java | RandomWalk.java
```

```
27 * Perform a random walk of m steps
28 *
29 * @param m the number of steps the drunkard takes
30 */
31 usage;
32 private void randomWalk(int m) {
33     //We are utilizing randomMove function which already moves the coordinates
34     for (int i = 0; i < m; i++) {
35         randomMove();
36     }
37 }
38 /**
39 * Private method to generate a random move according to the rules of the situation.
40 * That's to say, moves can be (+-1, 0) or (0, +-1).
41 */
42 usage;
43 private void randomMove() {
44     boolean ns = random.nextBoolean();
45     int step = random.nextBoolean() ? 1 : -1;
46     move(ns ? step : 0, ns ? 0 : step);
47 }
48 /**
49 * Method to compute the distance from the origin (the lamp-post where the drunkard starts) to his current position.
50 */
51 * @return the (Euclidean) distance from the origin to the current position.
52 */
53 public double distance() {
54     //As the distance is asked from the origin we are not explicitly mentioning the origin coordinates
55     return Math.pow((Math.pow(x, 2)+Math.pow(y, 2)), 0.5);
56 }
57 /**
58 * Perform multiple random walk experiments, returning the mean distance.
59 */
60 * @param m the number of steps for each experiment
61 */

Element ... More UI...
```

Coverage: 100% 8/8 7/7 Notifications

Full Requests Commit

Structure Bookmarks

Git Run Debug TODO Problems Terminal Services Build Dependencies

Build completed successfully in 947 ms (3 minutes ago)

32:46 LF UTF-8 4 spaces Spring2022

```
INFO6205 ~src/main/java/edu/neu/coe/info6205/randomwalk/RandomWalk.java
INFO6205 ~src/main/java/edu/neu/coe/info6205/randomwalk/RandomWalkTest.java
```

```
/*
 * @param m the number of steps for each experiment
 * @param n the number of experiments to run
 * @return the mean distance
 */
3 usages
public static double randomWalkMulti(int m, int n) {
    double totalDistance = 0;
    for (int i = 0; i < n; i++) {
        RandomWalk walk = new RandomWalk();
        walk.randomWalk(m);
        totalDistance = totalDistance + walk.distance();
    }
    return totalDistance / n;
}

public static void main(String[] args) {
    if (args.length == 0){
        throw new RuntimeException("Syntax: RandomWalk steps [experiments]");
    }
    //m is the number of steps or moves that can be made in any given( here will be chosen randomly) direction.
    int m = Integer.parseInt(args[0]);
    //n is the number of experiments or trials that function will be run.
    // this will be the default value for number of experiments if we do not give the input to args
    int n = 30;

    if (args.length > 1) {
        n = Integer.parseInt(args[1]);
    }
    double meanDistance = randomWalkMulti(m, n);
    System.out.println(m + " steps: " + meanDistance + " over " + n + " experiments");

    // This commented main method will be used for testing custom input code
    public static void main(String[] args) {
        int numberOfTrials = 143;
        for (int steps = 0; steps < 65; steps++) {
    }
```

```
INFO6205 ~src/main/java/edu/neu/coe/info6205/randomwalk/RandomWalk.java
INFO6205 ~src/main/java/edu/neu/coe/info6205/randomwalk/RandomWalkTest.java
```

```
}
    return totalDistance / n;

}

public static void main(String[] args) {
    if (args.length == 0){
        throw new RuntimeException("Syntax: RandomWalk steps [experiments]");
    }
    //m is the number of steps or moves that can be made in any given( here will be chosen randomly) direction.
    int m = Integer.parseInt(args[0]);
    //n is the number of experiments or trials that function will be run.
    // this will be the default value for number of experiments if we do not give the input to args
    int n = 30;

    if (args.length > 1) {
        n = Integer.parseInt(args[1]);
    }
    double meanDistance = randomWalkMulti(m, n);
    System.out.println(m + " steps: " + meanDistance + " over " + n + " experiments");

}

// This commented main method will be used for testing custom input code
public static void main(String[] args) {
    int numberOfTrials = 143;
    for (int steps = 0; steps < 65; steps++) {
        double meanDistance = randomWalkMulti(steps, numberOfTrials);
        System.out.println(steps + " steps: " + meanDistance + " over " + numberOfTrials + " experiments");
    }
}
```

All Test Cases Are Running for the implemented code - Attached > Test Cases + Test Results

IntelliJ IDEA 2022.1.1

INFO6205 - RandomWalkTest.java

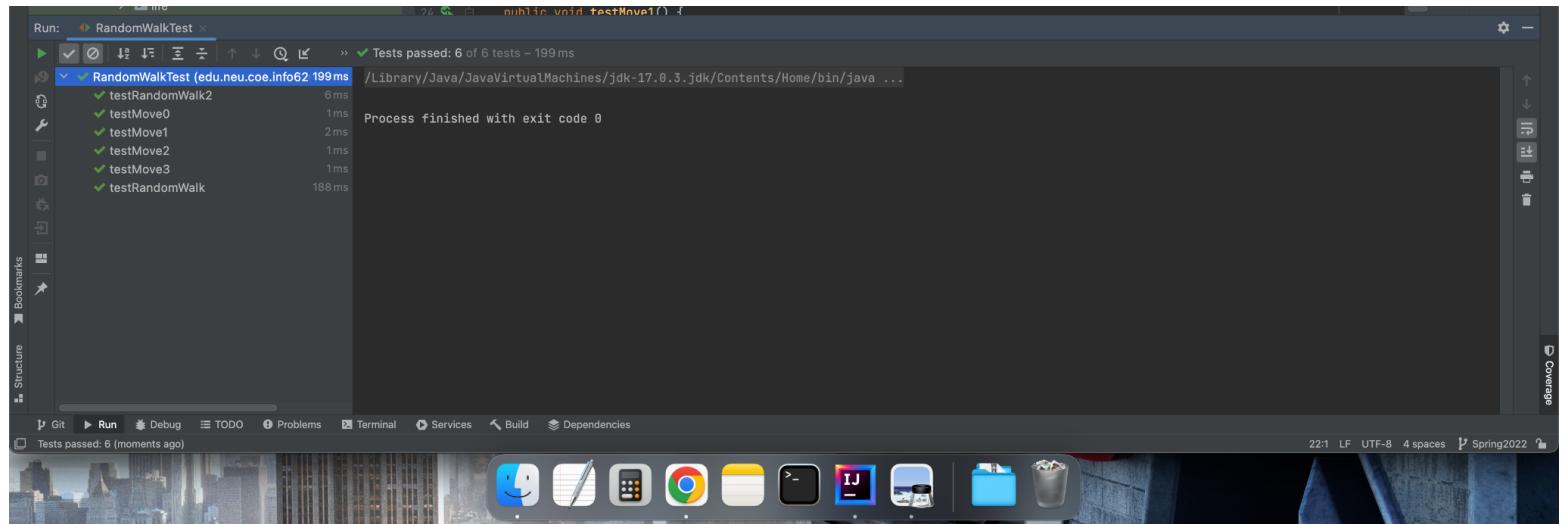
```
1 package edu.neu.coe.info6205.randomwalk;
2
3 import ...
4
5 public class RandomWalkTest {
6
7     @Test
8     public void testMove0() {
9         RandomWalk rw = new RandomWalk();
10        PrivateMethodTester pmt = new PrivateMethodTester(rw);
11        pmt.invokePrivate( name: "move", ...parameters: 1, 0);
12        assertEquals( expected: 1.0, rw.distance(), delta: 1.0E-7);
13    }
14
15    @Test
16    public void testMove1() {
17        RandomWalk rw = new RandomWalk();
18        PrivateMethodTester pmt = new PrivateMethodTester(rw);
19        pmt.invokePrivate( name: "move", ...parameters: 1, 0);
20        assertEquals( expected: 1.0, rw.distance(), delta: 1.0E-7);
21        pmt.invokePrivate( name: "move", ...parameters: 1, 0);
22        assertEquals( expected: 2.0, rw.distance(), delta: 1.0E-7);
23        pmt.invokePrivate( name: "move", ...parameters: -1, 0);
24        assertEquals( expected: 1.0, rw.distance(), delta: 1.0E-7);
25        pmt.invokePrivate( name: "move", ...parameters: -1, 0);
26        assertEquals( expected: 0.0, rw.distance(), delta: 1.0E-7);
27    }
28
29    /**
30     * 
31     */
32    @Test
33    public void testMove2() {
34        RandomWalk rw = new RandomWalk();
35        PrivateMethodTester pmt = new PrivateMethodTester(rw);
36    }
37
38    /**
39     * 
40     */
41    @Test
42    public void testMove3() {
43        RandomWalk rw = new RandomWalk();
44        PrivateMethodTester pmt = new PrivateMethodTester(rw);
45        pmt.invokePrivate( name: "move", ...parameters: 0, 1);
46        assertEquals( expected: 1.0, rw.distance(), delta: 1.0E-7);
47        pmt.invokePrivate( name: "move", ...parameters: 0, 1);
48        assertEquals( expected: 2.0, rw.distance(), delta: 1.0E-7);
49        pmt.invokePrivate( name: "move", ...parameters: 0, -1);
50        assertEquals( expected: 1.0, rw.distance(), delta: 1.0E-7);
51        pmt.invokePrivate( name: "move", ...parameters: 0, -1);
52        assertEquals( expected: 0.0, rw.distance(), delta: 1.0E-7);
53    }
54
55    /**
56     * 
57     */
58
59    @Test
60    public void testMove4() {
61        RandomWalk rw = new RandomWalk();
62        double root2 = Math.sqrt(2);
63        PrivateMethodTester pmt = new PrivateMethodTester(rw);
64        pmt.invokePrivate( name: "move", ...parameters: 1, 1);
65        assertEquals(root2, rw.distance(), delta: 1.0E-7);
66        pmt.invokePrivate( name: "move", ...parameters: 1, 1);
67        assertEquals( expected: 2 * root2, rw.distance(), delta: 1.0E-7);
68        pmt.invokePrivate( name: "move", ...parameters: 0, -2);
69        assertEquals( expected: 2.0, rw.distance(), delta: 1.0E-7);
70        pmt.invokePrivate( name: "move", ...parameters: -2, 0);
71        assertEquals( expected: 0.0, rw.distance(), delta: 1.0E-7);
72    }
73
74    /**
75     * 
76     */
77    @Test // Slow
78}
```

IntelliJ IDEA 2022.1.1

INFO6205 - RandomWalkTest.java

```
1 package edu.neu.coe.info6205.randomwalk;
2
3 import ...
4
5 public class RandomWalkTest {
6
7     @Test
8     public void testMove0() {
9         RandomWalk rw = new RandomWalk();
10        PrivateMethodTester pmt = new PrivateMethodTester(rw);
11        pmt.invokePrivate( name: "move", ...parameters: 0, 1);
12        assertEquals( expected: 1.0, rw.distance(), delta: 1.0E-7);
13        pmt.invokePrivate( name: "move", ...parameters: 0, 1);
14        assertEquals( expected: 2.0, rw.distance(), delta: 1.0E-7);
15        pmt.invokePrivate( name: "move", ...parameters: 0, -1);
16        assertEquals( expected: 1.0, rw.distance(), delta: 1.0E-7);
17        pmt.invokePrivate( name: "move", ...parameters: 0, -1);
18        assertEquals( expected: 0.0, rw.distance(), delta: 1.0E-7);
19    }
20
21    /**
22     * 
23     */
24    @Test
25    public void testMove1() {
26        RandomWalk rw = new RandomWalk();
27        double root2 = Math.sqrt(2);
28        PrivateMethodTester pmt = new PrivateMethodTester(rw);
29        pmt.invokePrivate( name: "move", ...parameters: 1, 1);
30        assertEquals(root2, rw.distance(), delta: 1.0E-7);
31        pmt.invokePrivate( name: "move", ...parameters: 1, 1);
32        assertEquals( expected: 2 * root2, rw.distance(), delta: 1.0E-7);
33        pmt.invokePrivate( name: "move", ...parameters: 0, -2);
34        assertEquals( expected: 2.0, rw.distance(), delta: 1.0E-7);
35        pmt.invokePrivate( name: "move", ...parameters: -2, 0);
36        assertEquals( expected: 0.0, rw.distance(), delta: 1.0E-7);
37    }
38
39    /**
40     * 
41     */
42
43    @Test
44    public void testMove2() {
45        RandomWalk rw = new RandomWalk();
46        PrivateMethodTester pmt = new PrivateMethodTester(rw);
47    }
48
49    /**
50     * 
51     */
52
53    @Test
54    public void testMove3() {
55        RandomWalk rw = new RandomWalk();
56        double root2 = Math.sqrt(2);
57        PrivateMethodTester pmt = new PrivateMethodTester(rw);
58        pmt.invokePrivate( name: "move", ...parameters: 1, 1);
59        assertEquals(root2, rw.distance(), delta: 1.0E-7);
60        pmt.invokePrivate( name: "move", ...parameters: 1, 1);
61        assertEquals( expected: 2 * root2, rw.distance(), delta: 1.0E-7);
62        pmt.invokePrivate( name: "move", ...parameters: 0, -2);
63        assertEquals( expected: 2.0, rw.distance(), delta: 1.0E-7);
64        pmt.invokePrivate( name: "move", ...parameters: -2, 0);
65        assertEquals( expected: 0.0, rw.distance(), delta: 1.0E-7);
66    }
67
68    /**
69     * 
70     */
71
72    @Test // Slow
73}
```

All Test Cases were passed



## Modified Main Method to run custom test cases

A screenshot of the IntelliJ IDEA IDE. The top navigation bar shows 'File', 'Edit', 'View', 'Navigate', 'Code', 'Refactor', 'Build', 'Run', 'Tools', 'Git', 'Window', and 'Help'. The title bar says 'INFO6205 – RandomWalk.java'. The main window shows the Java code for 'RandomWalk.java'. A comment block is present: // This commented main method will be used for testing custom input code. It contains a public static void main(String[] args) block with a loop from 0 to 65 steps, each step calling randomWalkMulti with 143 trials. The code editor has syntax highlighting and line numbers. The left sidebar shows the project structure with packages like 'edu.neu.coe.info6205' and 'test'. The status bar at the bottom right shows 'Thu May 19 9:41:45'. The background features a cityscape wallpaper.

Console Output of Main method with custom tests running for steps ranging from 0 to 65 and with each steps count running for 143 Trials ( Arbitrarily chosen)

Attached are Excel Files with the data and also the curve showing the trend of the mean distance

INFO6205 > src > main > java > edu > neu > coe > info6205 > randomwalk > RandomWalk.java

Project Run: RandomWalk

```
// This commented main method will be used for testing custom input code
public static void main(String[] args) {
    int numberofTrials = 143;
    for (int steps = 0; steps < 65; steps++) {
        double meanDistance = randomWalkMulti(steps, numberofTrials);
        System.out.println(steps + " steps: " + meanDistance + " over " + numberofTrials + " experiments");
    }
}
```

Run: RandomWalk

```
0 steps: 0.0 over 143 experiments
1 steps: 1.0 over 143 experiments
2 steps: 1.2476165239754535 over 143 experiments
3 steps: 1.5786550404544675 over 143 experiments
4 steps: 1.755867249184577 over 143 experiments
5 steps: 1.825120036991880 over 143 experiments
6 steps: 2.21527653868991 over 143 experiments
7 steps: 2.225634932749323 over 143 experiments
8 steps: 2.4623713740585598 over 143 experiments
9 steps: 2.677007712017024 over 143 experiments
10 steps: 2.7907290831825655 over 143 experiments
11 steps: 2.887348467229678 over 143 experiments
12 steps: 2.962889769830948 over 143 experiments
13 steps: 3.4297964990983838 over 143 experiments
14 steps: 3.0867061011024517 over 143 experiments
15 steps: 3.238097229497339 over 143 experiments
16 steps: 3.481507980076352 over 143 experiments
17 steps: 3.576394574664715 over 143 experiments
18 steps: 3.5978359062681633 over 143 experiments
19 steps: 3.7692796846420995 over 143 experiments
20 steps: 3.6611738950899784 over 143 experiments
21 steps: 3.9429186276449766 over 143 experiments
22 steps: 4.1057576100421 over 143 experiments
23 steps: 4.1019511581872345 over 143 experiments
24 steps: 4.374868175664384 over 143 experiments
25 steps: 4.503174039335038 over 143 experiments
26 steps: 4.891014546386989 over 143 experiments
27 steps: 4.619890549078666 over 143 experiments
28 steps: 4.93929111697967 over 143 experiments
29 steps: 4.562400511471586 over 143 experiments
30 steps: 4.870148575485079 over 143 experiments
31 steps: 4.45153802478822 over 143 experiments
32 steps: 5.218362984131392 over 143 experiments
33 steps: 5.225607592610178 over 143 experiments
34 steps: 5.37011782953883 over 143 experiments
35 steps: 5.60020173330867 over 143 experiments
36 steps: 5.4166187628806575 over 143 experiments
37 steps: 5.6423099235404434 over 143 experiments
38 steps: 5.221496937387628 over 143 experiments
39 steps: 4.9238464590687123 over 143 experiments
40 steps: 5.769411115634657 over 143 experiments
41 steps: 5.887216502016028 over 143 experiments
42 steps: 5.582205241682079 over 143 experiments
43 steps: 5.5671554088877055 over 143 experiments
44 steps: 5.721136495025584 over 143 experiments
```

Build completed successfully in 898 ms (moments ago)

INFO6205 > src > main > java > edu > neu > coe > info6205 > randomwalk > RandomWalk.java

Project Run: RandomWalk

```
// This commented main method will be used for testing custom input code
public static void main(String[] args) {
    int numberofTrials = 143;
    for (int steps = 0; steps < 65; steps++) {
        double meanDistance = randomWalkMulti(steps, numberofTrials);
        System.out.println(steps + " steps: " + meanDistance + " over " + numberofTrials + " experiments");
    }
}
```

Run: RandomWalk

```
22 steps: 4.148535476286246 over 143 experiments
23 steps: 4.1019511581872345 over 143 experiments
24 steps: 4.374868175664384 over 143 experiments
25 steps: 4.503174039335038 over 143 experiments
26 steps: 4.891014546386989 over 143 experiments
27 steps: 4.619890549078666 over 143 experiments
28 steps: 4.93929111697967 over 143 experiments
29 steps: 4.562400511471586 over 143 experiments
30 steps: 4.870148575485079 over 143 experiments
31 steps: 4.45153802478822 over 143 experiments
32 steps: 5.218362984131392 over 143 experiments
33 steps: 5.225607592610178 over 143 experiments
34 steps: 5.37011782953883 over 143 experiments
35 steps: 5.60020173330867 over 143 experiments
36 steps: 5.4166187628806575 over 143 experiments
37 steps: 5.6423099235404434 over 143 experiments
38 steps: 5.221496937387628 over 143 experiments
39 steps: 4.9238464590687123 over 143 experiments
40 steps: 5.769411115634657 over 143 experiments
41 steps: 5.887216502016028 over 143 experiments
42 steps: 5.582205241682079 over 143 experiments
43 steps: 5.5671554088877055 over 143 experiments
44 steps: 5.721136495025584 over 143 experiments
```

Build completed successfully in 898 ms (moments ago)

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help

INFO6205 > src > main > java > edu > neu > coe > info6205 > randomwalk > RandomWalk.java

Project RandomWalkTest.java RandomWalk.java

```

    // This commented main method will be used for testing custom input code
    public static void main(String[] args) {
        int numberofTrials = 143;
        for (int steps = 0; steps < 65; steps++) {
            double meanDistance = randomWalkMulti(steps, numberofTrials);
            System.out.println(steps + " steps: " + meanDistance + " over " + numberofTrials + " experiments");
        }
    }

```

Run: RandomWalk

44 steps: 0.721130473023500 over 143 experiments  
 45 steps: 6.36164783116609 over 143 experiments  
 46 steps: 5.7810147045841225 over 143 experiments  
 47 steps: 5.78764300973321 over 143 experiments  
 48 steps: 6.34176476567669 over 143 experiments  
 49 steps: 6.364891208915739 over 143 experiments  
 50 steps: 6.760931767503547 over 143 experiments  
 51 steps: 6.194998571564185 over 143 experiments  
 52 steps: 6.392503275437641 over 143 experiments  
 53 steps: 6.27252674801611 over 143 experiments  
 54 steps: 6.333519227952871 over 143 experiments  
 55 steps: 6.518216952729611 over 143 experiments  
 56 steps: 6.6018907833708855 over 143 experiments  
 57 steps: 6.2744403818501465 over 143 experiments  
 58 steps: 6.7462335622495715 over 143 experiments  
 59 steps: 7.1739244580499146 over 143 experiments  
 60 steps: 7.879818523399441 over 143 experiments  
 61 steps: 6.890805817750673 over 143 experiments  
 62 steps: 7.185755967457576 over 143 experiments  
 63 steps: 7.238873690148913 over 143 experiments  
 64 steps: 6.871111195874895 over 143 experiments

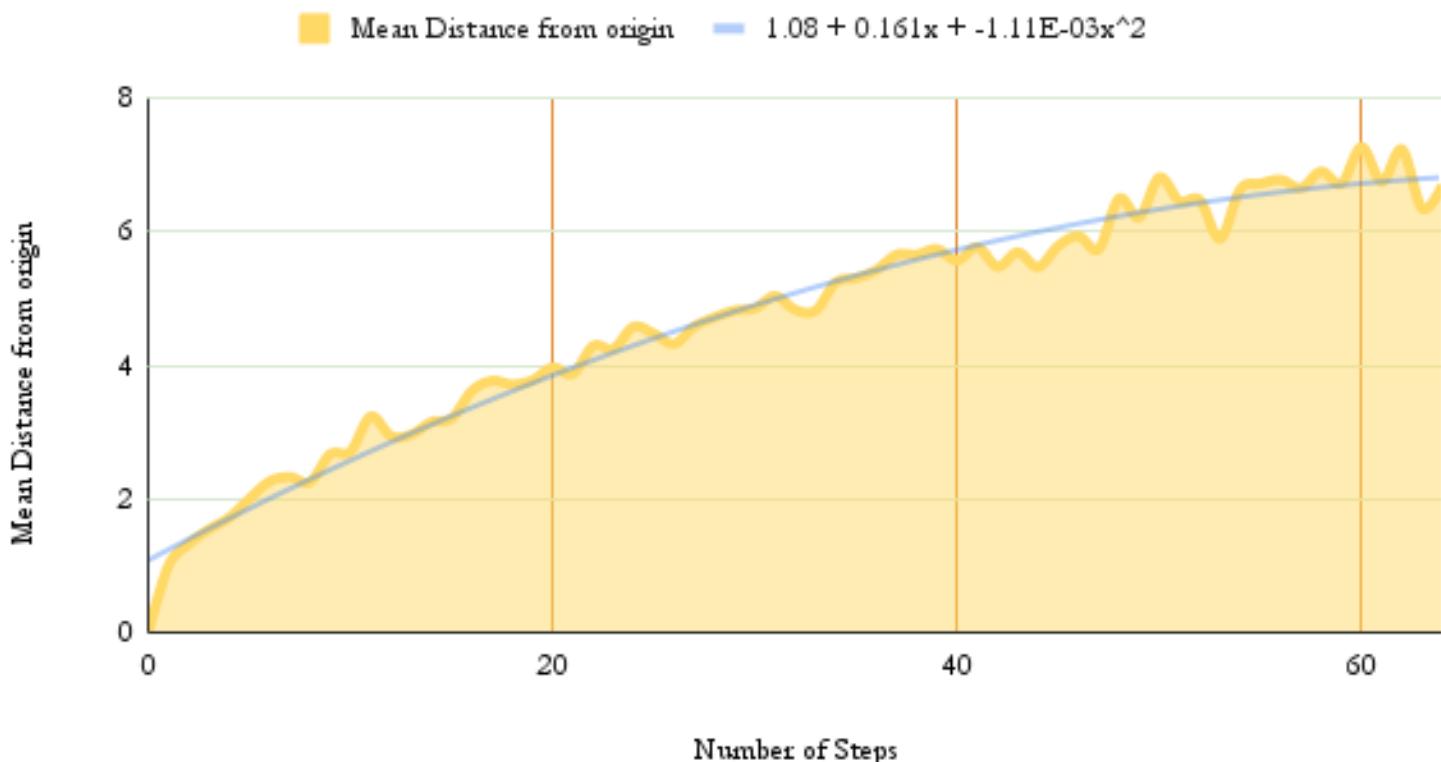
Process finished with exit code 0

Git Run Debug TODO Problems Terminal Services Build Dependencies

Build completed successfully in 898 ms (moments ago)

100:1 LF UTF-8 4 spaces Spring2022

## Mean Distance from origin vs Number of Steps @ 143 Trials



## **Excel File**

### **Assignment 01 Data**

## **Conclusion**

The mean distance showed a growth in small increments for the experiments run from 0 to 65 steps and each step running for 143 trials. From the graph above the trend also appears follows a curve

$$y = 1.08 + (0.161)(x) + (-1.11E-03)(x^2)$$

which is a second degree quadratic equation