MSSES INFO6205 13718 PROGRAM STRUCTURES & ALGORITHMS SEC 01- FALL2021
ASSIGNMENT 1: HIRAL RAJESH NAGDA (001560027)

---

1. Your conclusion about the relationship between *d* and *n?*

- **d ∝ √(n). L** where, d: distance, n: number of steps and L: length of step(here in our case L=1)

-According to the experiment, we perform multiple experiments on the number of steps hence, we can deduce the relationship between the distance of man from lamp post is approximately proportional to product of square root of number of steps and length of each step.

-Each unit step is equally likely to be in any direction (θ_j and θ_k). The displacements are random variables with identical means of zero, and their difference is also a random variable. Averaging over this distribution, which has equally likely positive and negative values yields an expectation value of 0, so

$$\langle |z|^2 \rangle = N.$$

The root-mean-square distance after N unit steps is therefore

$$|z|_{\mathrm{rms}} = \sqrt{N},$$

so with a step size of l, this becomes

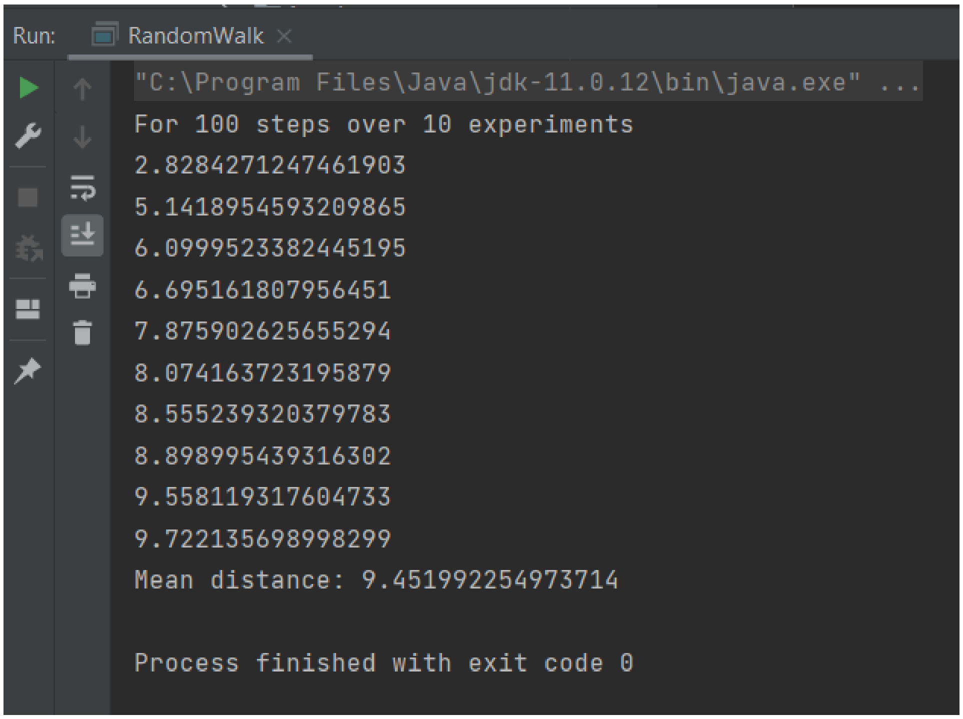
$$d_{\mathrm{rms}} = l\sqrt{N}.$$

In order to travel a distance d

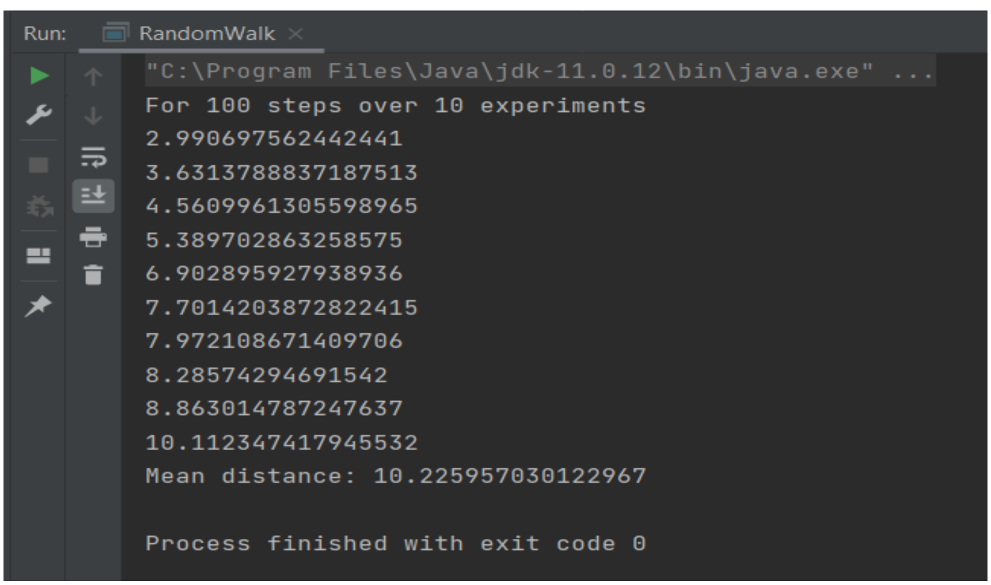
-We can conclude that mean distance is directly proportional to square root of number of steps, however accuracy may vary for higher values of n.

2. Your evidence to support that relationship (screenshot and/or graph and/or spreadsheet)**<u>For 100 steps over 10 experiments:</u>**

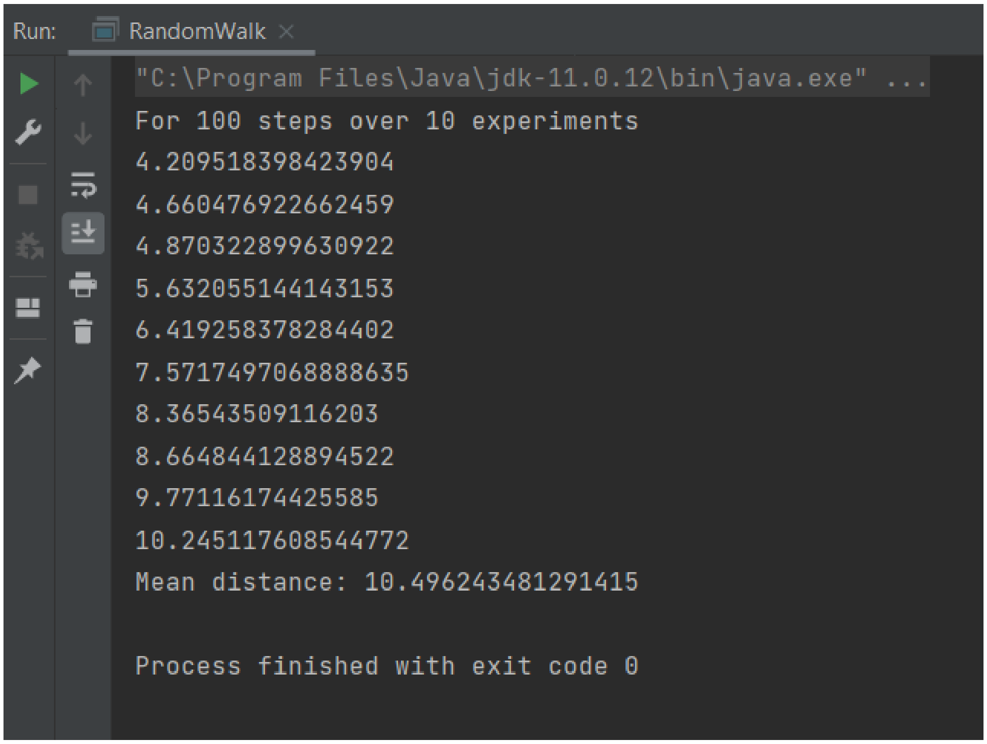
<u>Output 1:</u>



<u>Output 2:</u>

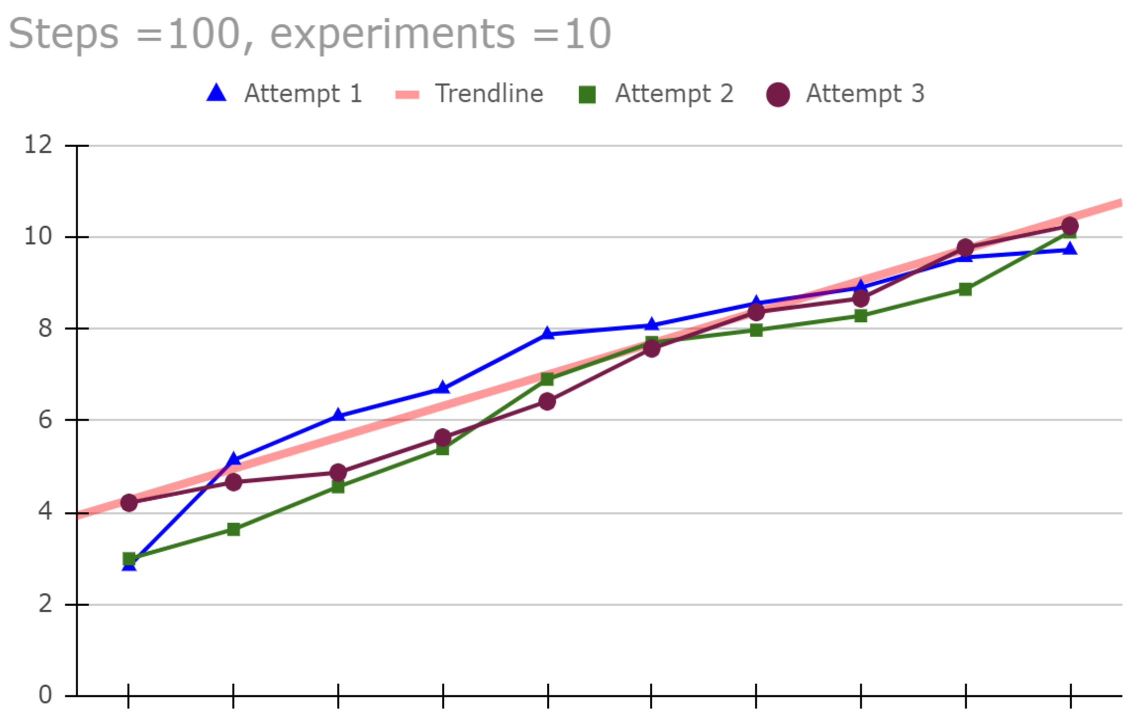


Output 3:



```
Run:  RandomWalk
"C:\Program Files\Java\jdk-11.0.12\bin\java.exe" ...
For 100 steps over 10 experiments
4.209518398423904
4.660476922662459
4.870322899630922
5.632055144143153
6.419258378284402
7.5717497068888635
8.36543509116203
8.664844128894522
9.77116174425585
10.245117608544772
Mean distance: 10.496243481291415

Process finished with exit code 0
```

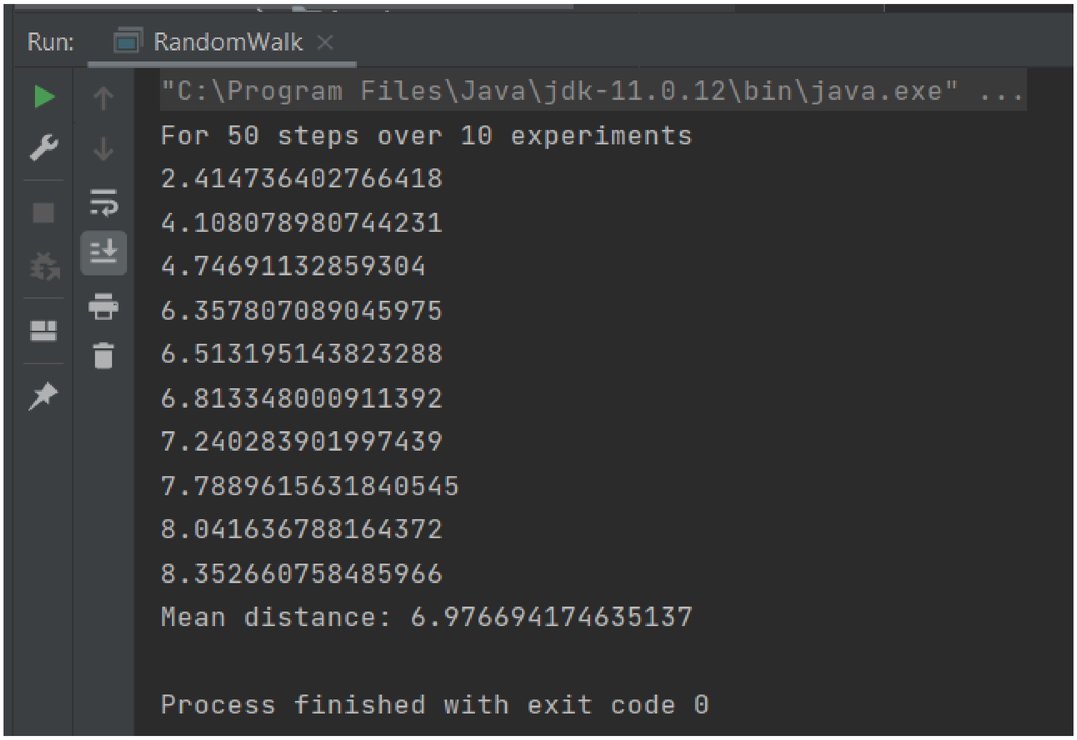
Analysing above 3 output using line graph inorder to derive relationship between d and n



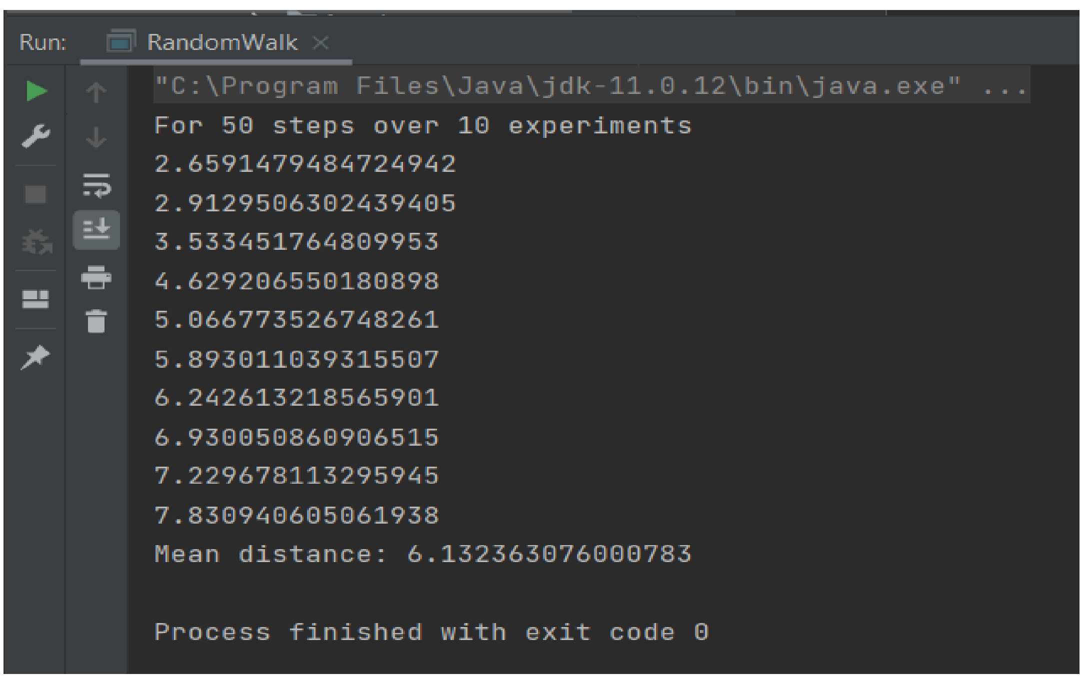
Conclusion: From the above line graph, the red line shows the average behaviour from which we can conclude that d ∝ √(n). L

For 50 steps over 10 experiments:

Output 1:



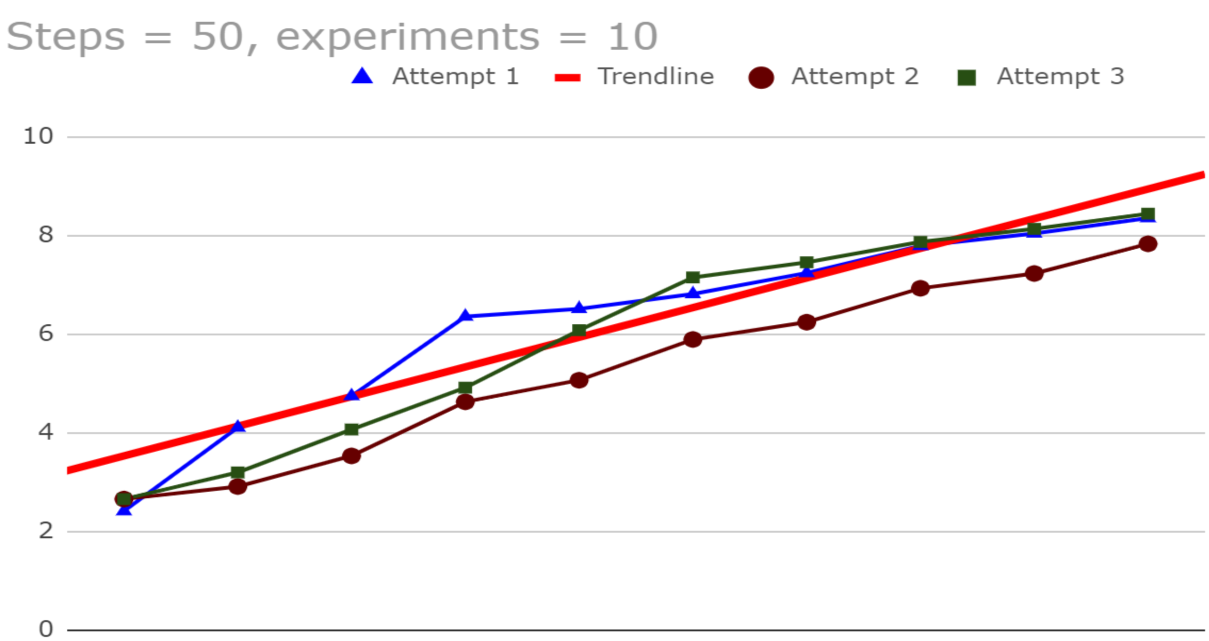
Output 2:



Output 3:

```
Run:  RandomWalk
"C:\Program Files\Java\jdk-11.0.12\bin\java.exe" ...
For 50 steps over 10 experiments
2.6591479484724942
3.1989600610251223
4.069140055634681
4.9166730517937145
6.081111935583839
7.149969090620684
7.456151416906094
7.868846756175107
8.133965205192778
8.441588089505409
Mean distance: 7.126040947287957

Process finished with exit code 0
```

Analysing above 3 output using line graph inorder to derive relationship between d and n



Conclusion: From the above line graph, the red line shows the average behaviour from which we can conclude that d ∝ √(n). L

3. Your code (*RandomWalk.java* plus anything else that you changed or created)

Pseudo-code

While Walking

Pick a random direction

Take a step in that direction

CODE:

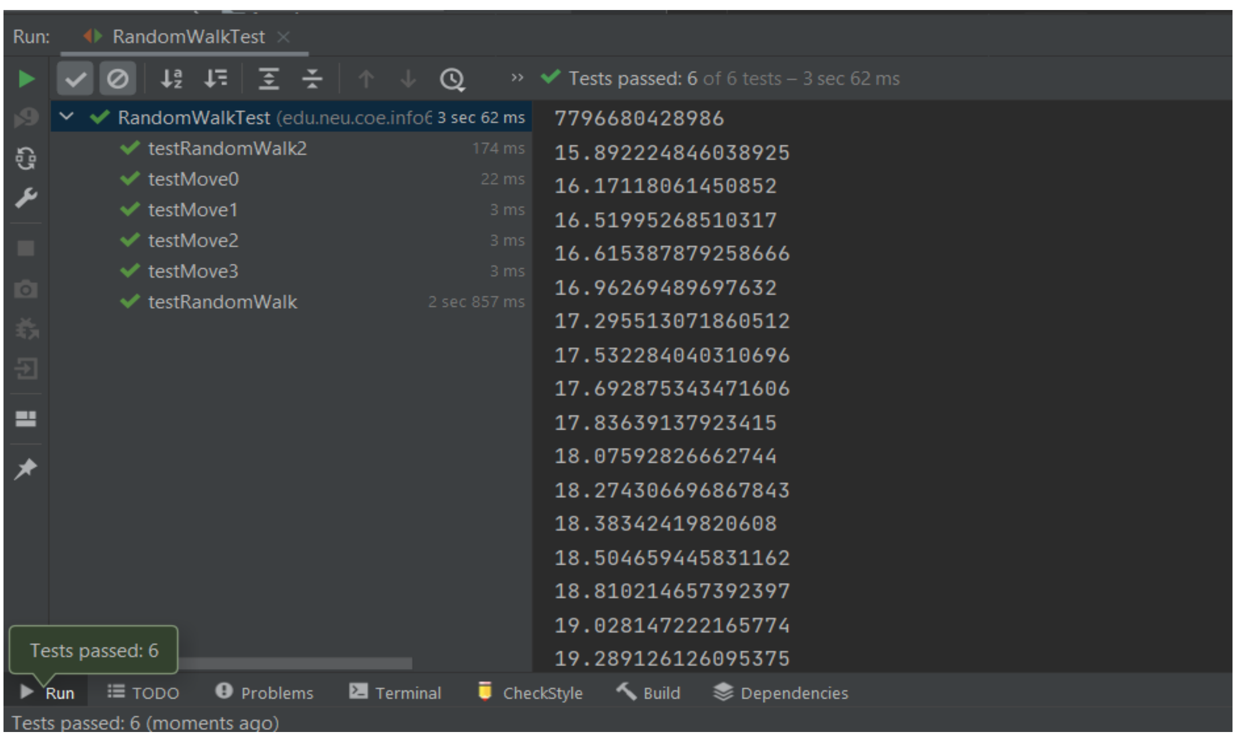
```
package edu.neu.coe.info6205.randomwalk;

import java.util.Random;

public class RandomWalk {

    private int x = 0;

    private int y = 0;

    private final Random random = new Random();

    /**

     * Private method to move the current position, that's to say the drunkard
moves

     * @param dx the distance he moves in the x direction

     * @param dy the distance he moves in the y direction

     */

    private void move(int dx, int dy) {

        // TODO you need to implement this

        this.x += dx;

        this.y += dy;

    }

    /**
```

```
     * Perform a random walk of m steps
     * @param m the number of steps the drunkard takes
     */
    private void randomWalk(int m) {
        // TO BE IMPLEMENTED
         for(int i = 0; i < m; i++)
                randomMove();  //for each step this method will calculate possible
moves randomly
    }
    /**
     * Private method to generate a random move according to the rules of the
situation.
     * That's to say, moves can be (+-1, 0) or (0, +-1).
     */
    private void randomMove() //method implementing random generation of x, y
co-ordinates
    {
        boolean ns = random.nextBoolean();
        int step = random.nextBoolean() ? 1 : -1;
        move(ns ? step : 0, ns ? 0 : step);
    }

    /**
     * Method to compute the distance from the origin (the lamp-post where the
drunkard starts) to his current position.
     * @return the (Euclidean) distance from the origin to the current position.
     */
```

```
    public double distance() {
        // TO BE IMPLEMENTED
        //System.out.println("Drunk man's final position on the x-y plane will be:
");
         //System.out.println("Final co-ordinates of x and y : "+ this.x+"
"+this.y);
         double distance = Math.sqrt(Math.pow(0-x, 2) + Math.pow(0-y, 2));
//euclidean's formula to calculate distance
        return distance;
    }

    /**
     * Perform multiple random walk experiments, returning the mean distance.
     * @param m the number of steps for each experiment
     * @param n the number of experiments to run
     * @return the mean distance
     */
    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();
              System.out.println(Math.sqrt(totalDistance));
        }
        return totalDistance / n;
```

```
    }
    public static void main(String[] args) {
        if (args.length == 0)
            throw new RuntimeException("Syntax: RandomWalk steps
[experiments]");
        int m = Integer.parseInt(args[0]);
        int n = 10;
        if (args.length > 1) n = Integer.parseInt(args[1]);
         System.out.println("For "+ m + "steps over 10 experiments\n");
        double meanDistance = randomWalkMulti(m, n);
        System.out.println("Mean Distance: " + meanDistance);
    }
}
```

4. A screenshot of the unit tests all passing.



Screenshots of different step value over same number of experiments:

Steps = 100, experiments =10

```
Run:    RandomWalk
"C:\Program Files\Java\jdk-11.0.12\bin\java.exe" ...
For 100 steps over 10 experiments
2.990697562442441
3.6313788837187513
4.560996130559896
5.389702863258575
6.902895927938936
7.7014203872822415
7.972108671409706
8.28574294691542
8.863014787247637
10.112347417945532
Mean distance: 10.225957030122967

Process finished with exit code 0
```

Steps = 200, experiments =10

```
Run:    RandomWalk
"C:\Program Files\Java\jdk-11.0.12\bin\java.exe" ...
For 200 steps over 10 experiments
4.605779351596907
6.802435711411007
7.1674368585619606
7.4729035238300865
8.241803992456184
8.748123131112715
9.377175084645732
10.465106101374396
11.365092538884674
11.737446642454001
Mean distance: 13.776765368445473

Process finished with exit code 0
```

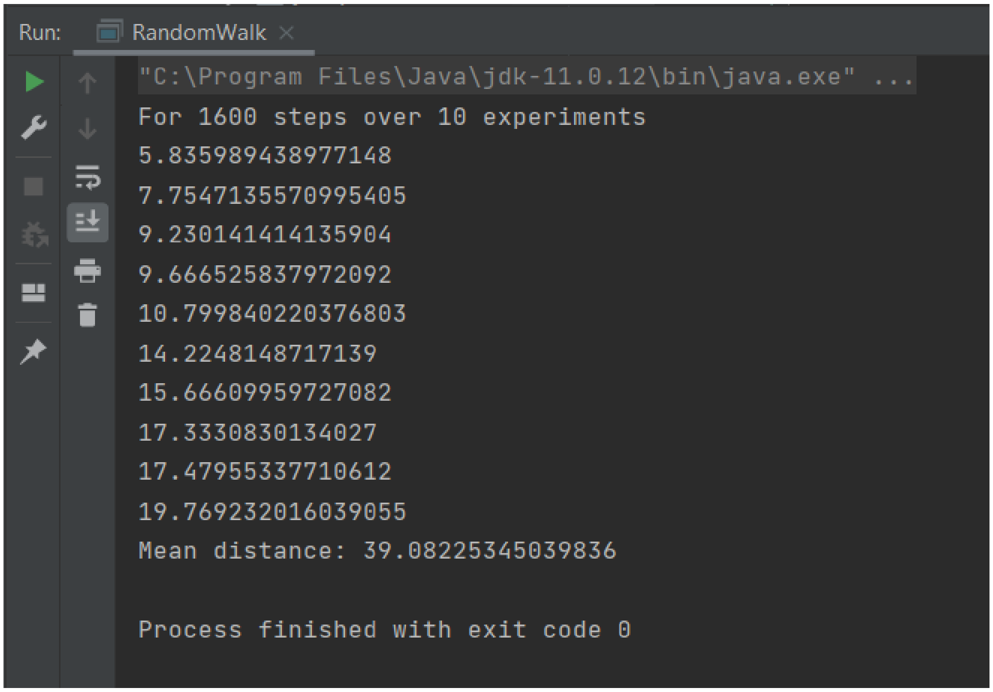
Steps = 400, experiments =10

```
Run:  RandomWalk
"C:\Program Files\Java\jdk-11.0.12\bin\java.exe" ...
For 400 steps over 10 experiments
2.6591479484724942
5.946035575013606
7.640488722965963
8.176940093947652
9.782983159738166
10.480882363973535
11.27437677342227
13.064364257424373
13.3374928679117
13.986020204056079
Mean distance: 19.56087611482648

Process finished with exit code 0
```

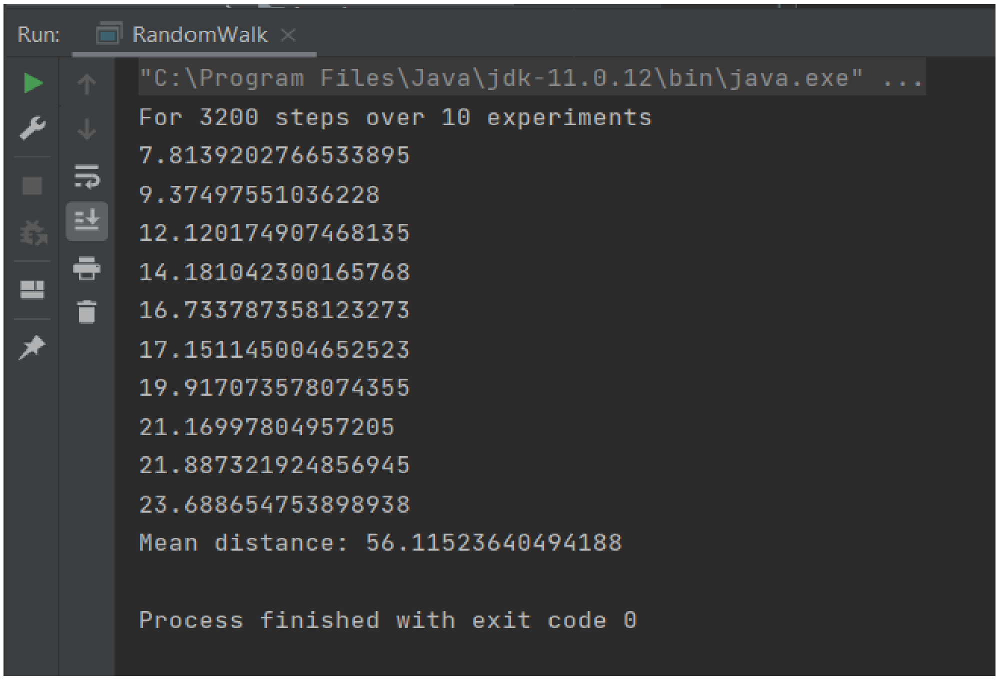
Steps = 800, experiments =10

```
"C:\Program Files\Java\jdk-11.0.12\bin\java.exe" ...
For 800 steps over 10 experiments
7.113117640155691
9.076174289139422
11.451710836717014
12.937960388274716
13.279122370042561
14.334917174256407
14.473764209520317
15.261036996686382
15.554891042767478
16.406022563980546
Mean distance: 26.91575763698388

Process finished with exit code 0
```

Steps = 1600, experiments = 10



Steps = 3200, experiments =10


Run:
RandomWalk
"C:\Program Files\Java\jdk-11.0.12\bin\java.exe" ...
For 3200 steps over 10 experiments
7.8139202766533895
9.37497551036228
12.120174907468135
14.181042300165768
16.733787358123273
17.151145004652523
19.917073578074355
21.16997804957205
21.887321924856945
23.688654753898938
Mean distance: 56.11523640494188
Process finished with exit code 0