Spring'22 PDS Lab 2 Assignment
Department of CSE, IIT Kharagpur

28th April 2022

Question-1

Consider a ball resting on the edge of a parabolic shaped container at height z. The bottom of the container is at $z = 0$. In this assignment, we will simulate the effect of sliding the ball with the aim of reaching at the bottom. Imagine that the sliding is controllable with defined step size. In order to do the simulation, we start with a user-given value of z, and slide down with a step size that is function of z. If z_i is the current position of the ball, then the next position z_{i+1} is computed using the following formula:

$$z_{i+1} = z_i - s_i,$$

where s_i is chosen as $\sqrt{z_i}$. That means, as the ball goes towards the bottom, the step size decreases since z decreases. However depending on the step size, the ball might "cross" the bottom and roll on the other side of the container. To simulate this effect, when the z value goes below zero, simply change the sign, and continue repeating the above mentioned process. You also need to consider the case when the z value does not change in consecutive iterations and gets constant (*"trapped situation"*). In this situation, change the s in above equation as $s = s - (1/k^2)$, where k is the current iteration number. This will decrease z by a small amount and will help in escaping from the *trapped situation*.

Your task will be to run a loop for k number of iterations starting at some z value (both provided by the user). In each iteration, you will be decreasing the height z according to the process describe above, and print the value of z, as well as the iteration number. At any stage of the simulation if the value of z reaches zero, immediately exit from the loop.

Example:
Input: $z = 100.0$, $k = 50$
Output:

```
(90.000000,1), (80.513167,2),   (71.540254,3),   (63.082107,4)   (55.139682,5),
(47.714072,6),   (40.806535,7),   (34.418536,8),   (28.551805,9),
(23.208416,10),   (18.390904,11),   (14.102443,12),   (10.347121,13),
(7.130427,14),   (4.460141,15),   (2.348236,16),   (0.815841,17),
```

```
(0.087398,18),   (0.208234,19),   (0.248093,20),   (0.249996,21),
(0.250000,22),   (0.250000,23),   (0.000000,24)
```

Question-2

Consider the movement of a particle starting at $x = 0$. The particle can move either in forward (towards $+x$), or backward (towards $-x$), which is randomly selected (like obtaining head or tail in coin tossing). This is a typical observation in several random real life phenomena (e.g. motion of atoms, Brownian motion, etc).

Assume that each step of the particle is equal to one unit of distance (i.e. +1 or -1). We wish to compute the *expected distance* of the particle from the starting point (or how far does the particle travel on the average) after n number of steps. Although apparently it might seem that the average progress will be zero, but actually as n increases, the total distance also increases. You have to write a code to simulate n number of random steps and calculate the final distance d_n from the initial step, repeat this process for a large number of iterations (k) and add the square of distance d_n in each iteration to get the total distance D. That is, $D = \sum_{n=1}^{k} d_n^2$. Finally the expected distance D_E is computed as, $D_E = \sqrt{D/k}$. Eventually you will see that the total distance covered is approximately the square root of the number of steps. Of course the answer will vary little bit in every execution of the program due to the randomness.

For your convenience, the random step generator code is provided as follows. Include the following headers in your code:

```
#include<time.h>, #include<stdlib.h>
```

Add this at the beginning of main( ) function: *srand(time(NULL));* Then the following line of code generates +1 or -1 randomly and assigns to variable p:

```
p = rand() & 1 ? -1 : 1;
```

Example:
Input: n = 25, k = 100000
Output: 5.00119

Input: n = 100, k = 100000
Output: 10.0023

Input: n = 50, k = 100000
Output: 7.0903

Question-3

Consider that n number of machines in the CSE department are broken that needs to be fixed. Suppose each machine has different configuration, and the department is planning to assign each machine to an individual vendor for repairing, and there are total n number of vendors. Now each of these vendors are demanding different prices for fixing different machines. The price for each machine demanded by each vendor is put in an array. To keep things simple, consider $n = 4$. For example, the price list of machine1 is stored as int $m1[4] = \{8200, 8300, 6900, 9200\}$. This means for machine1, the price quote of vendor1 is 8200, vendor2 it is 8300, vendor3 it is 6900, and vendor4 it is 9200. Similarly m2[ ], m3[ ], and m4 [ ] stores the price quotations of other machines for the 4 vendors.

Now the goal is to assign one machine to each vendor so that all tasks are completed with the smallest total cost. In order to do this, you have to consider all possible combinations of machines and vendors, and find the smallest cost and the corresponding combination of machine and vendor. Note that each vendor will be assigned exactly one machine.

Your program should output the total cost, as well as the assigned vendors to the machines. You may initialize the arrays m1, m2, m3, and m4 in the code (no need to take user input).

Example:

Input:

$m1 \rightarrow 8200, 8300, 6900, 9200$
$m2 \rightarrow 7700, 3700, 4900, 9200$
$m3 \rightarrow 1100, 6900, 500, 8600$
$m4 \rightarrow 800, 900, 9800, 2300$

Output:

Total cost = 14000
Machine1 - Vendor3
Machine2 - Vendor2
Machine3 - Vendor1
Machine4 - Vendor4