Animesh has empty candy jars, numbered from to , with infinite capacity. He performs operations. Each operation is described by integers, , , and . Here, and are indices of the jars, and is the number of candies to be added inside each jar whose index lies between and (both inclusive). Can you tell the average number of candies after operations?

Example

The array has elements that all start at . In the first operation, add to the first elements. Now the array is . In the second operation, add to the last elements (3 – 5). Now the array is and the average is 10. Sincd 10 is already an integer value, it does not need to be rounded.

Function Description

Complete the solve function in the editor below.

Solve has the following parameters:

Int n: the number of candy jars

Int operations[m][3]: a 2-dimensional array of operations

Returns

Int: the floor of the average number of canidies in all jars

Input Format

The first line contains two integers, and , separated by a single space.

Lines follow. Each of them contains three integers, , , and , separated by spaces.

Constraints

Sample Input

STDIN Function

----- --------

5 3 n = 5, operations[] size = 3

1 2 100 operations = [[1, 2, 100], [2, 5, 100], [3, 4, 100]]

2 5 100

3 4 100

Sample Output

160

Explanation

Initially each of the jars contains 0 candies

0 0 0 0 0

First operation:

100 100 0 0 0

Second operation:

100 200 100 100 100

Third operation:

100 200 200 200 100

Total = 800, Average = 800/5 = 160

Submissions: 169

Max Score: 10

Difficulty: Easy

Rate This Challenge:

More

Return output//n

1

#!/bin/python3

2

3

Import math

4

Import os

5

Import random

6

Import re

7

Import sys

8

9

#

10

# Complete the ‘solve’ function below.

11

#

12

# The function is expected to return an INTEGER.

13

# The function accepts following parameters:

14

# 1. INTEGER n

15

# 2. 2D\_INTEGER\_ARRAY operations

16

#

17

18

Def solve(n, operations):

19

# Write your code here

20

Output=0

21

For o in operations:

22

Output+=o[2]\*(o[1]-o[0]+1)

23

Return output//n

24

25

If \_\_name\_\_ == ‘\_\_main\_\_’:

26

Fptr = open(os.environ[‘OUTPUT\_PATH’], ‘w’)

27

28

First\_multiple\_input = input().rstrip().split()

29

30

N = int(first\_multiple\_input[0])

31

32

M = int(first\_multiple\_input[1])

33

34

Operations = []

35

36

For \_ in range(m):

37

Operations.append(list(map(int, input().rstrip().split())))

38

39

Result = solve(n, operations)

40

41

Fptr.write(str(result) + ‘\n’)

42

43

Fptr.close()

44

Line: 23 Col: 21

Run Code Submit CodeUpload Code as File

Test against custom input

Testcase 0

Congratulations, you passed the sample test case.

Click the Submit Code button to run your code against all the test cases.

Input (stdin)

5 3

1 2 100

2 5 100

3 4 100

Your Output (stdout)

160

Expected Output

160