

# A - Haiku

Time Limit: 2 sec / Memory Limit: 256 MB

Score : 100 points

## Problem Statement

As a New Year's gift, Dolphin received a string  $s$  of length 19.

The string  $s$  has the following format: [five lowercase English letters],[seven lowercase English letters],[five lowercase English letters].

Dolphin wants to convert the comma-separated string  $s$  into a space-separated string.

Write a program to perform the conversion for him.

## Constraints

- The length of  $s$  is 19.
- The sixth and fourteenth characters in  $s$  are `,` and `.`
- The other characters in  $s$  are lowercase English letters.

## Input

The input is given from Standard Input in the following format:

$s$

## Output

Print the string after the conversion.

## Sample Input 1

happy,newyear,enjoy

## Sample Output 1

happy newyear enjoy

Replace all the commas in `happy,newyear,enjoy` with spaces to obtain `happy newyear enjoy`.

## Sample Input 2

```
haiku,atcoder,tasks
```

## Sample Output 2

```
haiku atcoder tasks
```

---

## Sample Input 3

```
abcde,fghihgf,edcba
```

## Sample Output 3

```
abcde fghihgf edcba
```

# B - Sum of Three Integers

---

Time Limit: 2 sec / Memory Limit: 256 MB

Score : 200 points

## Problem Statement

You are given two integers  $K$  and  $S$ .

Three variable  $X$ ,  $Y$  and  $Z$  takes integer values satisfying  $0 \leq X, Y, Z \leq K$ .

How many different assignments of values to  $X$ ,  $Y$  and  $Z$  are there such that  $X + Y + Z = S$ ?

## Constraints

- $2 \leq K \leq 2500$
- $0 \leq S \leq 3K$
- $K$  and  $S$  are integers.

---

## Input

The input is given from Standard Input in the following format:

$K$   $S$

## Output

Print the number of the triples of  $X$ ,  $Y$  and  $Z$  that satisfy the condition.

---

## Sample Input 1

2 2

## Sample Output 1

6

There are six triples of  $X$ ,  $Y$  and  $Z$  that satisfy the condition:

- $X = 0, Y = 0, Z = 2$
- $X = 0, Y = 2, Z = 0$
- $X = 2, Y = 0, Z = 0$
- $X = 0, Y = 1, Z = 1$
- $X = 1, Y = 0, Z = 1$
- $X = 1, Y = 1, Z = 0$

## Sample Input 2

5 15

## Sample Output 2

1

The maximum value of  $X + Y + Z$  is 15, achieved by one triple of  $X$ ,  $Y$  and  $Z$ .

# C - Back and Forth

Time Limit: 2 sec / Memory Limit: 256 MB

Score : 300 points

## Problem Statement

Dolphin resides in two-dimensional Cartesian plane, with the positive  $x$ -axis pointing right and the positive  $y$ -axis pointing up.

Currently, he is located at the point  $(sx, sy)$ . In each second, he can move up, down, left or right by a distance of 1.

Here, both the  $x$ - and  $y$ -coordinates before and after each movement must be integers.

He will first visit the point  $(tx, ty)$  where  $sx < tx$  and  $sy < ty$ , then go back to the point  $(sx, sy)$ , then visit the point  $(tx, ty)$  again, and lastly go back to the point  $(sx, sy)$ .

Here, during the whole travel, he is not allowed to pass through the same point more than once, except the points  $(sx, sy)$  and  $(tx, ty)$ .

Under this condition, find a shortest path for him.

## Constraints

- $-1000 \leq sx < tx \leq 1000$
- $-1000 \leq sy < ty \leq 1000$
- $sx, sy, tx$  and  $ty$  are integers.

## Input

The input is given from Standard Input in the following format:

```
 $sx$   $sy$   $tx$   $ty$ 
```

## Output

Print a string  $S$  that represents a shortest path for Dolphin.

The  $i$ -th character in  $S$  should correspond to his  $i$ -th movement.

The directions of the movements should be indicated by the following characters:

- U: Up
- D: Down
- L: Left
- R: Right

If there exist multiple shortest paths under the condition, print any of them.

### Sample Input 1

```
0 0 1 2
```

### Sample Output 1

```
UURDDLLUUURRDRDDLLU
```

One possible shortest path is:

- Going from  $(sx, sy)$  to  $(tx, ty)$  for the first time:  $(0, 0) \rightarrow (0, 1) \rightarrow (0, 2) \rightarrow (1, 2)$
- Going from  $(tx, ty)$  to  $(sx, sy)$  for the first time:  $(1, 2) \rightarrow (1, 1) \rightarrow (1, 0) \rightarrow (0, 0)$
- Going from  $(sx, sy)$  to  $(tx, ty)$  for the second time:  $(0, 0) \rightarrow (-1, 0) \rightarrow (-1, 1) \rightarrow (-1, 2) \rightarrow (-1, 3) \rightarrow (0, 3) \rightarrow (1, 3) \rightarrow (1, 2)$
- Going from  $(tx, ty)$  to  $(sx, sy)$  for the second time:  $(1, 2) \rightarrow (2, 2) \rightarrow (2, 1) \rightarrow (2, 0) \rightarrow (2, -1) \rightarrow (1, -1) \rightarrow (0, -1) \rightarrow (0, 0)$

### Sample Input 2

```
-2 -2 1 1
```

### Sample Output 2

```
UURRURRDDDLLDLLULUUURRURRDDDLLDL
```

# D - Candidates of No Shortest Paths

Time Limit: 2 sec / Memory Limit: 256 MB

Score : 400 points

## Problem Statement

You are given an undirected connected weighted graph with  $N$  vertices and  $M$  edges that contains neither self-loops nor double edges.

The  $i$ -th ( $1 \leq i \leq M$ ) edge connects vertex  $a_i$  and vertex  $b_i$  with a distance of  $c_i$ .

Here, a *self-loop* is an edge where  $a_i = b_i$  ( $1 \leq i \leq M$ ), and *double edges* are two edges where  $(a_i, b_i) = (a_j, b_j)$  or  $(a_i, b_i) = (b_j, a_j)$  ( $1 \leq i < j \leq M$ ).

A *connected graph* is a graph where there is a path between every pair of different vertices.

Find the number of the edges that are not contained in any shortest path between any pair of different vertices.

## Constraints

- $2 \leq N \leq 100$
- $N - 1 \leq M \leq \min(N(N - 1)/2, 1000)$
- $1 \leq a_i, b_i \leq N$
- $1 \leq c_i \leq 1000$
- $c_i$  is an integer.
- The given graph contains neither self-loops nor double edges.
- The given graph is connected.

## Input

The input is given from Standard Input in the following format:

```
N M
a1 b1 c1
a2 b2 c2
:
aM bM cM
```

## Output

Print the number of the edges in the graph that are not contained in any shortest path between any pair of different vertices.

## Sample Input 1

```
3 3
1 2 1
1 3 1
2 3 3
```

## Sample Output 1

```
1
```

In the given graph, the shortest paths between all pairs of different vertices are as follows:

- The shortest path from vertex 1 to vertex 2 is: vertex 1  $\rightarrow$  vertex 2, with the length of 1.
- The shortest path from vertex 1 to vertex 3 is: vertex 1  $\rightarrow$  vertex 3, with the length of 1.
- The shortest path from vertex 2 to vertex 1 is: vertex 2  $\rightarrow$  vertex 1, with the length of 1.
- The shortest path from vertex 2 to vertex 3 is: vertex 2  $\rightarrow$  vertex 1  $\rightarrow$  vertex 3, with the length of 2.
- The shortest path from vertex 3 to vertex 1 is: vertex 3  $\rightarrow$  vertex 1, with the length of 1.
- The shortest path from vertex 3 to vertex 2 is: vertex 3  $\rightarrow$  vertex 1  $\rightarrow$  vertex 2, with the length of 2.

Thus, the only edge that is not contained in any shortest path, is the edge of length 3 connecting vertex 2 and vertex 3, hence the output should be 1.

## Sample Input 2

```
3 2
1 2 1
2 3 1
```

## Sample Output 2

```
0
```

Every edge is contained in some shortest path between some pair of different vertices.