Softuniada 2019

Grid Voyage

Doctor Sanity's pet pokemon – Slifer, is training his sense of direction with a game that Doc designed for him, called Grid Voyage. You should help Slifer, as he is not the smartest pokemon in existence. Try to write an algorithm which simulates the Grid Voyage game.

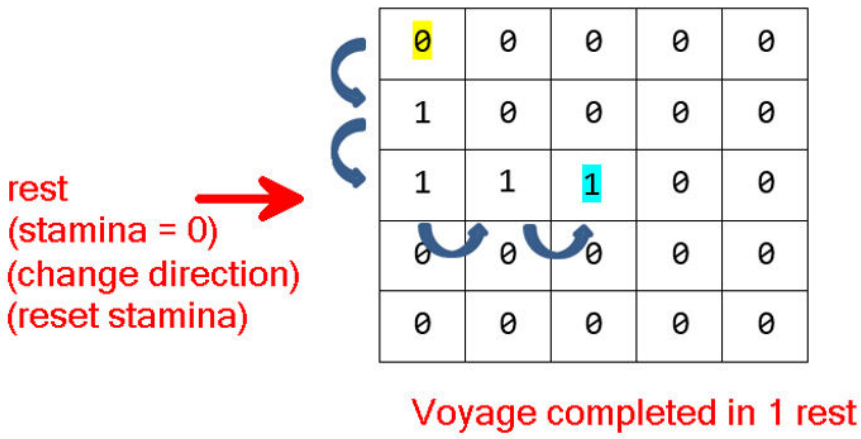
You will be given **N** – an **integer**. You must create a 2-D matrix of integers with **N rows** and **N columns**, each cell with value – **`0`**. You will then receive the **coordinates** of the **start point**.

Afterwards, you will start receiving lines, containing **coordinates** of a **destination point**, the **initial direction** (**`left`**, **`right`**, **`up`**, **`down`**) and **current stamina**. You must **reach** the **destination point** doing a grid-based movement, only changing directions when you run out of stamina.

- You consume **1 stamina** per **step** (per **cell movement**).
- When you run out of **stamina**, you **MUST change** the **direction** and **reset** your **stamina**.
- You must **increase** the **value** of **each cell** you **step on** with **1**.

Example: Initial position – **`[0, 0]`**.

First destination **`[2, 2]`**. Initial direction – **down**. Stamina – **2**.



Slifer is always looking for the **fastest path**, which means that he should **always move** in directions, **towards** his **next destination**.

Slifer is fat – he can **ONLY turn left** or **right**. He **CANNOT** completely **turn around** in a reversed direction.

Slifer possesses a bit of intellect – if he is in a **dilemma** (he can go several directions, and all of them are leading to his target), he will try to **turn LEFT first**.

- If he **cannot turn left** (not enough space in matrix, to make the needed steps before the next change in direction), he will try to **turn right**.
- If he **can** turn **neither left**, **nor right**, he will deem the current Voyage – **impossible** and return to his previous position. In this case, there should be **NO cells affected** by **increased value**.

If it **is possible** to reach the destination, you must print **how many rests** (how many times you've **ran out** of **stamina** and you've **changed direction**) Slifer took to reach it. If it is **NOT possible** to reach the destination, you must print "**Voyage `impossible`**".

When you receive the command "**`eastern odyssey`**", the input sequence should end, and you should print the **whole matrix**.

Input

The input consists of several input lines:

- On the **first** input line you will receive **N** – the **dimensions** of the 2-D matrix.
- On the **second** input line you will receive **X** and **Y**, separated by a **space** – the **initial position** of Slifer.
- On the next **several** input lines you will receive **coordinates** of a **destination point**, a **direction** and **stamina** in the following format: **`{destinationX} {destinationY} {direction} {stamina}`**
- When you receive the command "**`eastern odyssey`**", the input sequence should end.

Output

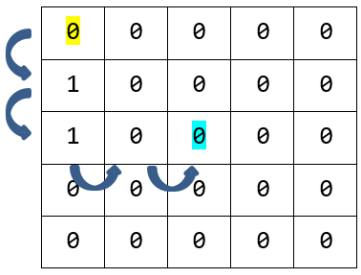
As output you must print:

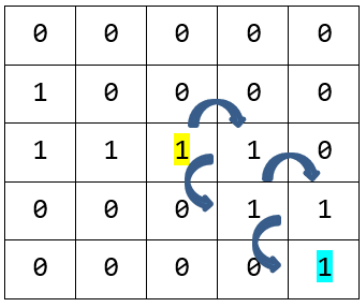
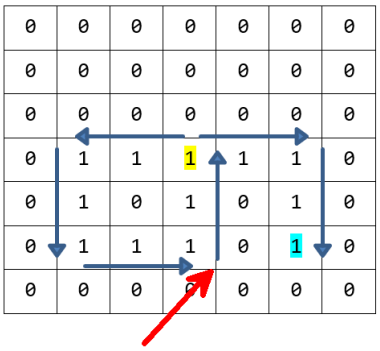
- For **every voyage**:
  - If it is **possible**, the number of **rests** it took to **reach** the **destination point**.
  - If it is **NOT possible** – you should print "**`Voyage impossible`**".
- At the end of the program:
  - The **whole matrix** – each **row** on a **new line**, each **column** separated by a **space**, from the others.

Constraints

- The integer **N** will be in **range [0, 50]**.
- The **coordinates** of the **initial point** and **destination points** will **always** be **VALID** and inside the matrix.
- The **directions** will **always** be **valid**.
- The **stamina** will always be in **range [1, 50]**.

Examples

Input	Output	Comment
5 0 0 2 2 down 2 4 4 right 1 eastern odyssey	1 3 0 0 0 0 0 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 0 0 0 0 1	First Voyage:  Second Voyage:

		 The **stamina** is **1** this time. We must **change direction** after **each step**.
7 3 3 5 5 left 2 6 6 right 2 eastern odyssey	5 Voyage impossible 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 1 0 1 0 1 0 0 1 1 1 0 1 0 0 0 0 0 0 0 0	 Slifer reached a point where he can go both left and right. He always chooses left first.