# Problem 5 - Bits at Crossroads

Bits are usually very boring. They walk only left <-> right and up <-> down. Your task is to write a program which builds diagonal roads to break the monotonous bits’ habits.

You are given a **square board** of bits (**size NxN**). Bit positions on **each line** are counted from **right to left**. Line numbers are counted from **top to bottom**. Initially all bits are **set to zero**. You can build **two diagonal roads** easily if you know the coordinates of the **roads’ intersection** (line number, bit position). A crossroad is an intersection between two roads.

Example: The line number is 2 and the bit position is 5: (2, 5). There are two diagonal roads – from (0, 7) to (7, 0) and from (0, 3) to (4, 7) and one crossroad (2, 5) (*see Fig. 1*). Cells shaded grey are the roads and cells shaded black are crossroads.

Example 2: We have two predefined crossroads (2, 5) and (3, 2). Now there are 4 crossroads in total – the start points and two additional points (1, 4) and (4, 3) *(see Fig. 2)*.

Your task is to write a program that prints the integer representation of each row from the final board and finds the number of crossroads on the board.

*Fig. 1 Fig. 2*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Number |
| 0 | **1** | 0 | 0 | 0 | **1** | 0 | 0 | 0 | 136 |
| 1 | 0 | **1** | 0 | **1** | 0 | 0 | 0 | 0 | 80 |
| 2 | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 32 |
| 3 | 0 | **1** | 0 | **1** | 0 | 0 | 0 | 0 | 80 |
| 4 | **1** | 0 | 0 | 0 | **1** | 0 | 0 | 0 | 136 |
| 5 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 4 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 2 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 1 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Number |
| 0 | **1** | 0 | **1** | 0 | **1** | 0 | 0 | 0 | 168 |
| 1 | 0 | **1** | 0 | **1** | 0 | 0 | 0 | **1** | 81 |
| 2 | 0 | 0 | **1** | 0 | **1** | 0 | **1** | 0 | 42 |
| 3 | 0 | **1** | 0 | **1** | 0 | **1** | 0 | 0 | 84 |
| 4 | **1** | 0 | 0 | 0 | **1** | 0 | **1** | 0 | 138 |
| 5 | 0 | 0 | 0 | **1** | 0 | **1** | 0 | **1** | 21 |
| 6 | 0 | 0 | **1** | 0 | 0 | 0 | **1** | 0 | 34 |
| 7 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | **1** | 65 |

**Input**

* On the first line, you are given an integer number N that represents the size of the board.
* Each of the next lines will hold the position of a predefined crossroad – two integer numbers, separated with a single space:
  + The first integer will be the line number.
  + The second integer will be the bit position.
* When you read the “**end**” command from the console print the result.

The input data will always be valid and in the format described. There is no need to check it explicitly.

**Output**

The output data must be printed on the console.

* On the first N lines print the integer representations of each row of the board.
* On the last line print the total count of all crossroads on the board.

**Constrains**

* **The size N of the board** is an integer in the range [3 ... 32].
* Each **start point** will **always** be a **zero bit**.
* Each **start point** will **always** be a **valid crossroad - the line number and bit position will both be in the range [0 … N)**.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 10  3 1  0 1  5 2  end | 146  77  45  19  47  76  154  305  608  192  4 |

|  |  |
| --- | --- |
| **Input** | **Output** |
| 16  2 5  3 2  8 5  12 3  end | 41128  20561  10282  5205  2698  1301  682  1361  2208  4433  8874  17684  35338  5141  10274  20545  14 |

|  |  |
| --- | --- |
| **Input** | **Output** |
| 8  2 5  3 2  end | 168  81  42  84  138  21  34  65  4 |