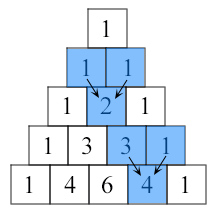
# Problem 3

Create a function called **get\_magic\_triangle** which will receive a **single parameter** (integer **n**) and it should create a magic triangle which follows those **rules**:

* We start with this simple triangle **[[1], [1, 1]]**
* We generate the **next rows** until we reach **n** amount of rows
* **Each number** in each row is equal to the **sum** of the **two numbers** right **above it** in the triangle
* If the current number has **no neighbor** to the upper **left/rigth**, we just take the **existing neighbor**

After you create the magic triangle, **return** it as a **multidimensional list**. Here is an example with **n = 5**



***Note: Submit only the function in the judge system***

### Input

* There will be **no inputs**
* The function will be tested by passing different values of n
* You can test your function with the test code below

### Constraints

* N will be in range **[2, 100]**

### Examples

|  |  |
| --- | --- |
| **Test Code** | **Output** |
| get\_magic\_triangle(5) | [[1], [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1]] |