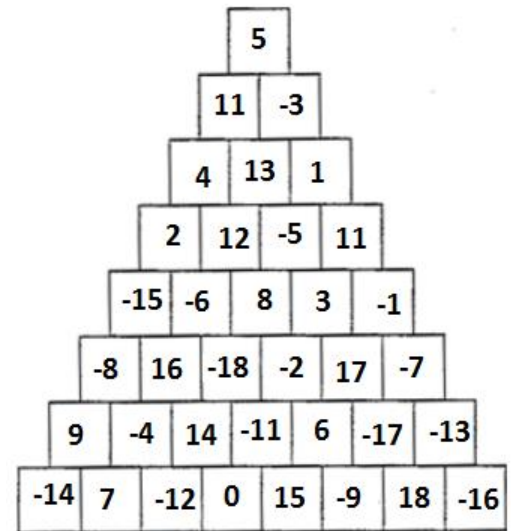


Pyramid Coordinates

This problem deals with type of Coordinate System called the Pyramid Coordinate System. A sample of the Pyramid Coordinate System with 8 rows can be seen in the figure to the right. Indexing starts at the top of the Pyramid with the coordinates of (1, 1) and the associated value 5. Beneath coordinate (1, 1) are coordinates (2, 1) and (2, 2). The value associated with (2, 1) is 11 and the value associated with (2, 2) is -3. The value associated with coordinate (4, 3) is -5.

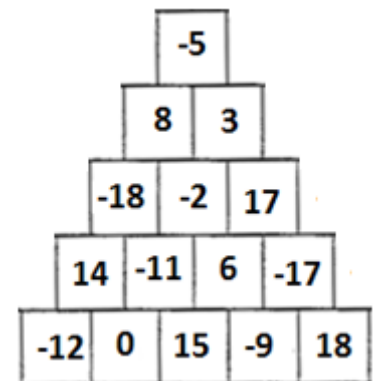
It is noted:

- The associated values for all Pyramids (in this problem) will ALWAYS be a String.
- all Pyramids will have at least three rows
- The second row of every pyramid has two columns
- Third row has three columns
- The n^{th} row has n columns



Each Coordinate in the Pyramid has neighbors. In general, the neighbors of a Coordinate are the two Coordinates above, the Coordinates on either side, and the two Coordinates beneath it. Some of these neighbor may not exist if the original Coordinate is on the 'edge' In the sample Pyramid, the neighbors of Coordinate (4, 2) include (3, 1), (3, 2), (4, 1), (4, 3), (5, 2), and (5, 3).

Given a starting Coordinate, a subPyramid can be extracted from a given Pyramid. In the sample Pyramid, the subPyramid starting at Coordinate (4, 3) is the Pyramid shown to the right.



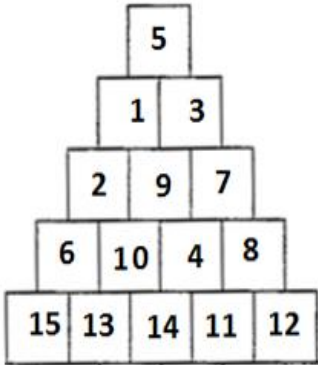
In this problem you will complete the three methods in the `Pyramid` class. The three methods are:

- `get`
- `getNeighbors`
- `and getSubPyramid`

Sample code of the `Pyramid` class and the three methods are on the following pages.

Pyramid Coordinates

The following code is used to create the indicated Pyramid (which is different from the sample Pyramid) and will be used in the following examples.

Code to create a Pyramid	Pyramid
<pre>List<ArrayList<String>> objs = new ArrayList<ArrayList<String>>(); ArrayList row1 = new ArrayList<String>(); row1.add("5"); objs.add(row1); ArrayList row2 = new ArrayList<String>(); row2.add("1"); row2.add("3"); objs.add(row2); ArrayList row3 = new ArrayList<String>(); row3.add("2"); row3.add("9"); row3.add("7"); objs.add(row3); ArrayList row4 = new ArrayList<String>(); row4.add("6"); row4.add("10"); row4.add("4"); row4.add("8"); objs.add(row4); ArrayList row5 = new ArrayList<String>(); row5.add("15"); row5.add("13"); row5.add("14"); row5.add("11"); row5.add("12"); objs.add(row5); PyramidCoordinates myP = new PyramidCoordinates(objs);</pre>	

The following table show sample results of the `get(m, n)` method using `PyramidCoordinates myP` from the previous table.

You may assume : `1 <= m <= pyramind.size()`
 `1 <= n <= pyramind.get(m).size()`

The following code	Returns
<code>myP.get(3, 3)</code>	"7"
<code>myP.get(4, 3)</code>	"4"
<code>myP.get(5, 1)</code>	"15"

Sample code of the `getNeighbors` and `getSubPyramid` methods are on the following pages.

Pyramid Coordinates

The following table show sample results of the `getNeighbors(x_i , x_j)` method using `PyramidCoordinates myP` from the first table.

You may assume: `1 <= x_i < neighbors.size()`
`1 <= x_j < neighbors.get(x_i).size()`

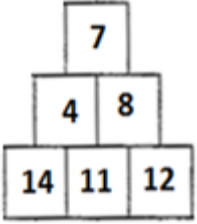
The following code	Returns
<pre>String[] neighbors = myP.getNeighbors(4, 2) List<String> answers = new ArrayList<String>(); answers.add("2"); answers.add("9"); answers.add("6"); answers.add("4"); answers.add("13"); answers.add("14"); String[] neighbors = myP.getNeighbors(4, 2);</pre>	
<code>neighbors.length</code>	6
<code>answers.contains(neighbors[0]);</code> <code>answer.remove(neighbors[0]);</code>	true
<code>answers.contains(neighbors[1]);</code> <code>answer.remove(neighbors[1]);</code>	true
<code>answers.contains(neighbors[2]);</code> <code>answer.remove(neighbors[2]);</code>	true
<code>answers.contains(neighbors[3]);</code> <code>answer.remove(neighbors[3]);</code>	true
<code>answers.contains(neighbors[4]);</code> <code>answer.remove(neighbors[4]);</code>	true
<code>answers.contains(neighbors[5]);</code> <code>answer.remove(neighbors[5]);</code>	true

Sample code of the `getSubPyramid` method is on the following page.

Pyramid Coordinates

The following table show sample results of the `getSubPyramid(xi, xj)` method using `PyramidCoordinates myP` from the first table.

You may assume: $1 \leq x_i < \text{neighbors.size()} - 2$
 $1 \leq x_j < \text{neighbors.get}(x_i).\text{size}()$

The following code	Returns
<pre>PyramidCoordinates result = myP.getSubPyramid(3, 3);</pre>	
<pre>result.get(1, 1);</pre>	"7"
<pre>result.get(2, 1);</pre>	"4"
<pre>result.get(2, 2);</pre>	"8"
<pre>result.get(3, 1);</pre>	"14"
<pre>result.get(3, 2);</pre>	"11"
<pre>result.get(3, 3);</pre>	"12"