LEC 06

CSE 122

2D Arrays

BEFORE WE START

Talk to your neighbors:
What is your go-to study snack?

Music: Miya's 23wi CSE 122 Playlist

Instructor Miya Natsuhara

TAs

Ayush	Atharva	Ernie	Ambika
Connor	Julia	Di	Elizabeth
Poojitha	Megana	Logan	Joe
Andrew A	Joey	Shivani	Jin
Andrew C	Eesha	Michelle	Ben
Jasmine	Lilian	Steven	Evelyn
Darel	Thomas	Kevin	Kent
Gabe	Leon	Ken	
Karen	Melissa	Vivek	
Colton	Audrey	Autumn	

Questions during Class?

Raise hand or send here

sli.do #cse122

Lecture Outline

- Announcements
- 2D Arrays Review
- Images
- Images with 2D Arrays!

Announcements

- First round of Quiz 0 Retakes happened yesterday!
  - Maximum one retake per quiz
  - Quiz 0 retakes still possible 1/31 and 2/7
  - Grades for Quiz 0 Retakes will be released after 2/7
- Programming Assignment 1 is due tomorrow (Thurs, Jan 26)
- Creative Project 1 released on Friday (Jan 27)
- Quiz 1 scheduled for Tuesday, Feb 7

Lecture Outline

- Announcements
- **2D Arrays Review**
- Images
- Images with 2D Arrays!

(PCM) Arrays

- The type of an array is *ElementType*[]
    - *ElementType* can be any type!
- Can store multiple elements *of the same type*
- Need to specify length of array and type of elements it will store at creation

```
int[] arr = new int[4];
```

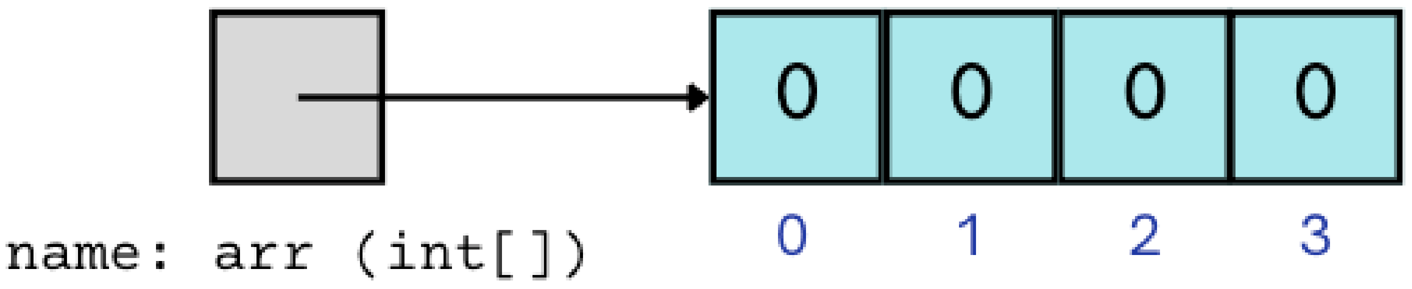
type name array creation code

```
int[]
double[]
String[]
boolean[]
char[]
```

(PCM) Arrays

- The type of an array is *ElementType*[]
  - *ElementType* can be any type!
- Can store multiple elements *of the same type*
- Need to specify length of array and type of elements it will store at creation

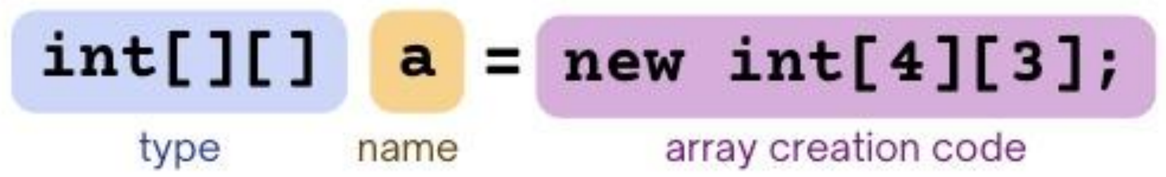
```
int[] arr = new int[4];
```



(PCM) 2D Arrays

An array of arrays!

- The *ElementType* of the array is another array itself!
  - Your first example of “nested data structures”
    - There will be more!



```
int[][] a = new int[4][3];
```

`int[][]`

`double[][]`

`String[][]`

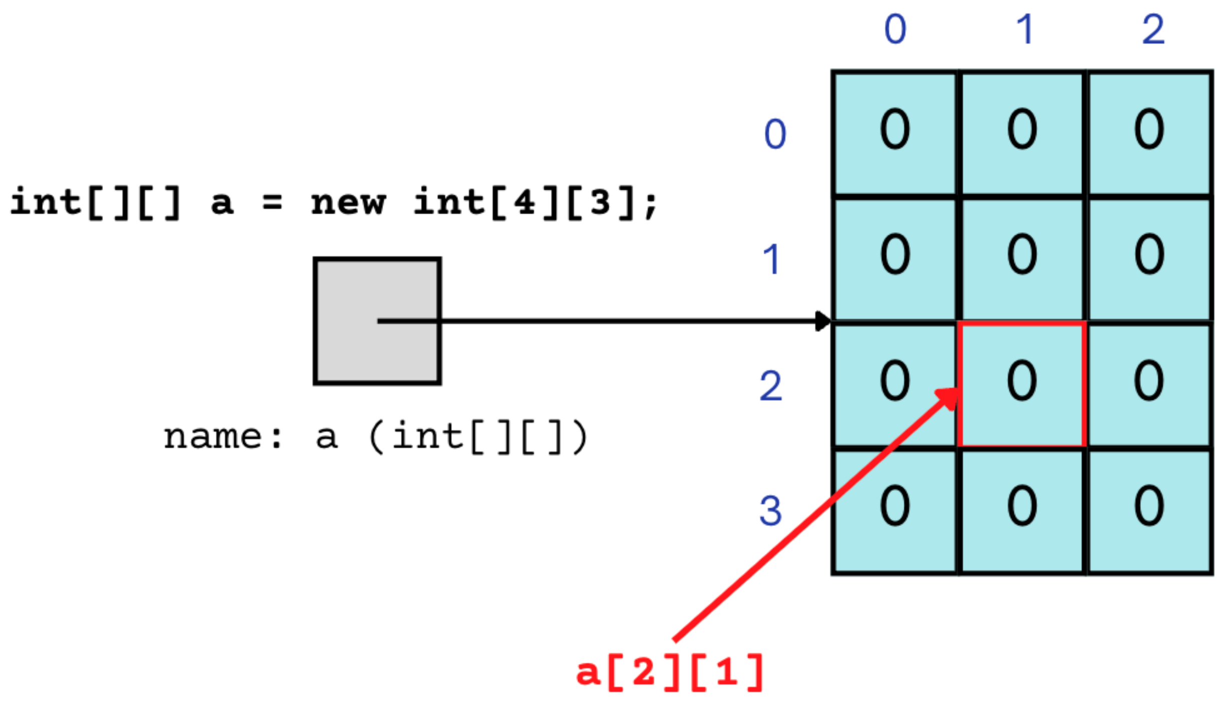
`boolean[][]`

`char[][]`

(PCM) 2D Arrays

An array of arrays!

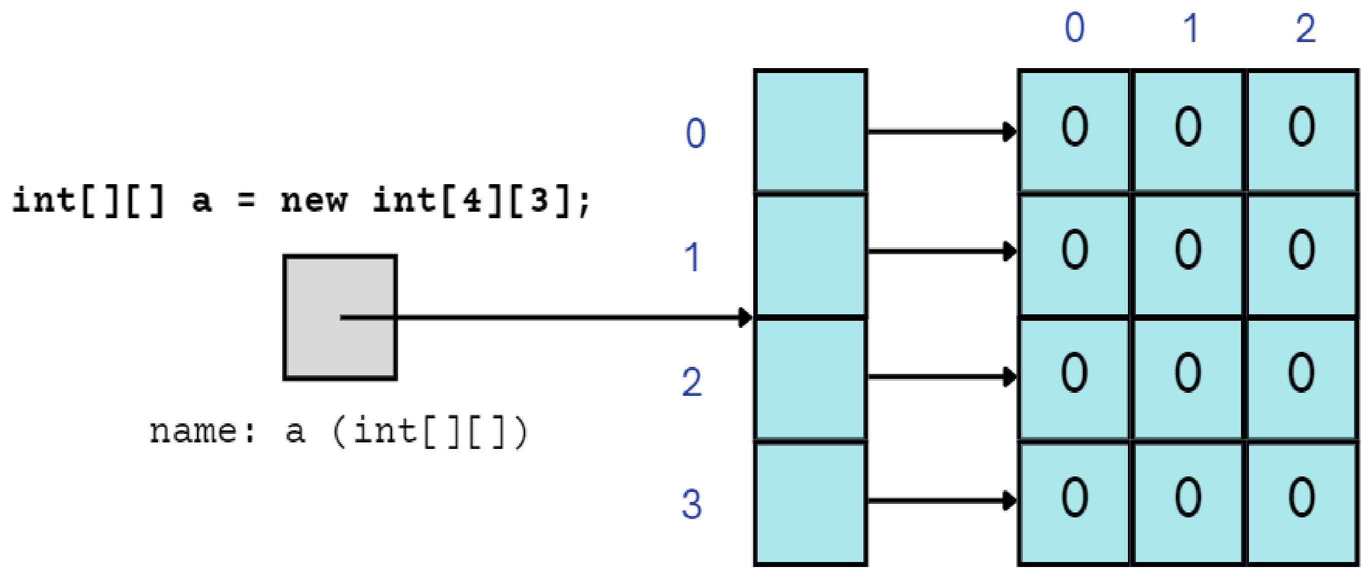
The two dimensions are "rows" and "columns"



(PCM) 2D Arrays

A slightly more accurate view…

reference semantics



(PCM) 2D Array Traversals

```
for (int i = 0; i < list.length; i++) {
    for (int j = 0; j < list[i].length; j++) {
        // do something with list[i][j]
    }
}
```

Arrays Utility Class

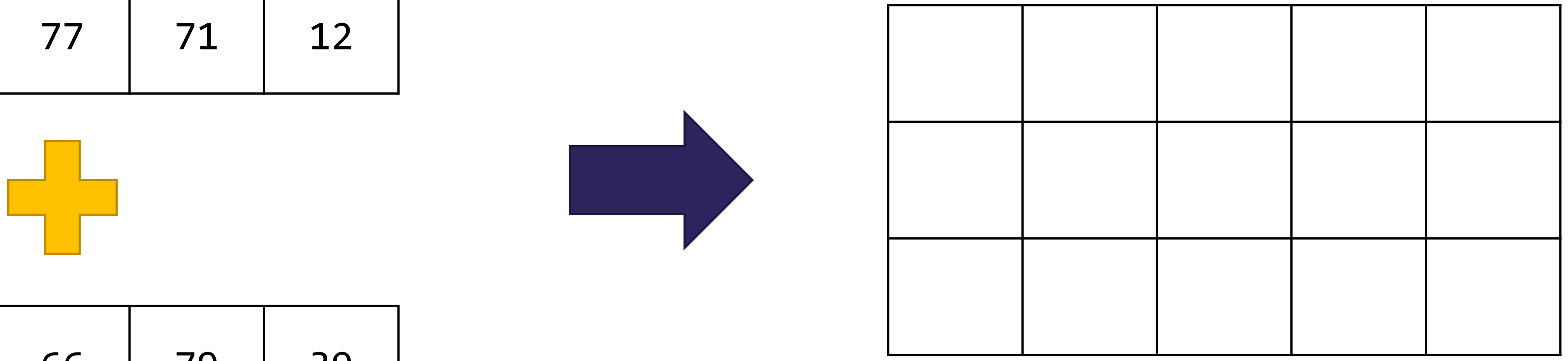
Method	Description
`Arrays.toString(array);`	Returns a `String` representing the array, such as `"[10, 30, -25, 17]"`
`Arrays.fill(array, value);`	Sets every element to the given value
`Arrays.equals(array1, array2);`	Returns `true` if the two arrays contain the same elements in the same order
`Arrays.deepToString(array);`	Returns a `String` representing the array; if the array contains other arrays as elements, the `String` represents their contents, and so on. For example, `"[[99, 151], [30, 5]]"`
`Arrays.deepEquals(array1, array2);`	Returns `true` if the two arrays contain the same elements in the same order; if the array(s) contain other arrays as elements, their contents are tested for equality, and so on.

Applications of 2D Arrays

- Matrices
  - Useful in various applications requiring complex math!
- Board games
  - (e.g., chess/checkerboard, tic tac toe, sudoku)
- Representing information in a grid or table
  - (e.g., scorekeeping, gradebook)
- Image processing

matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12



70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12

+

70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

→

matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12

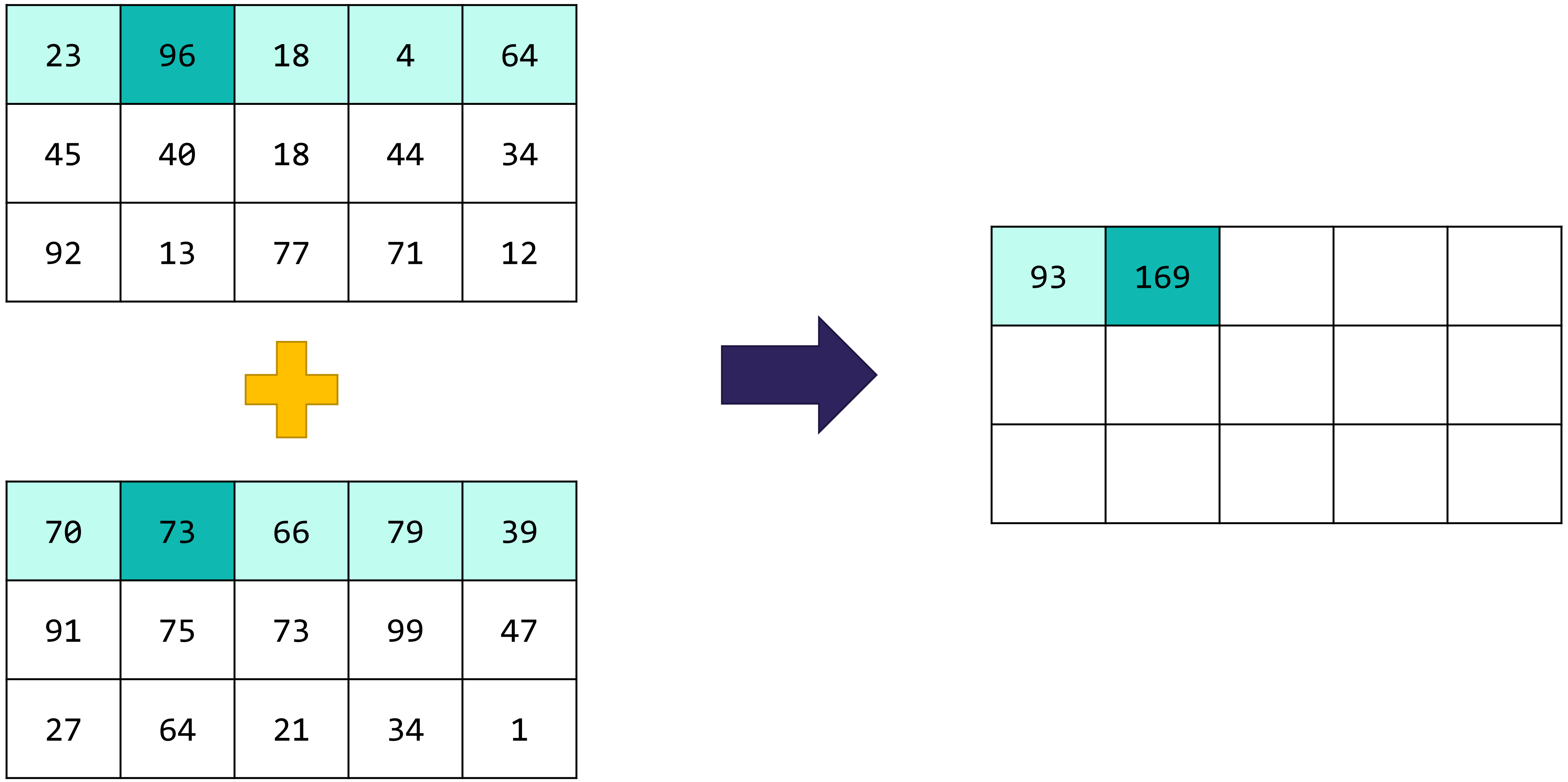
+

70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

→

93				

matrixAdd

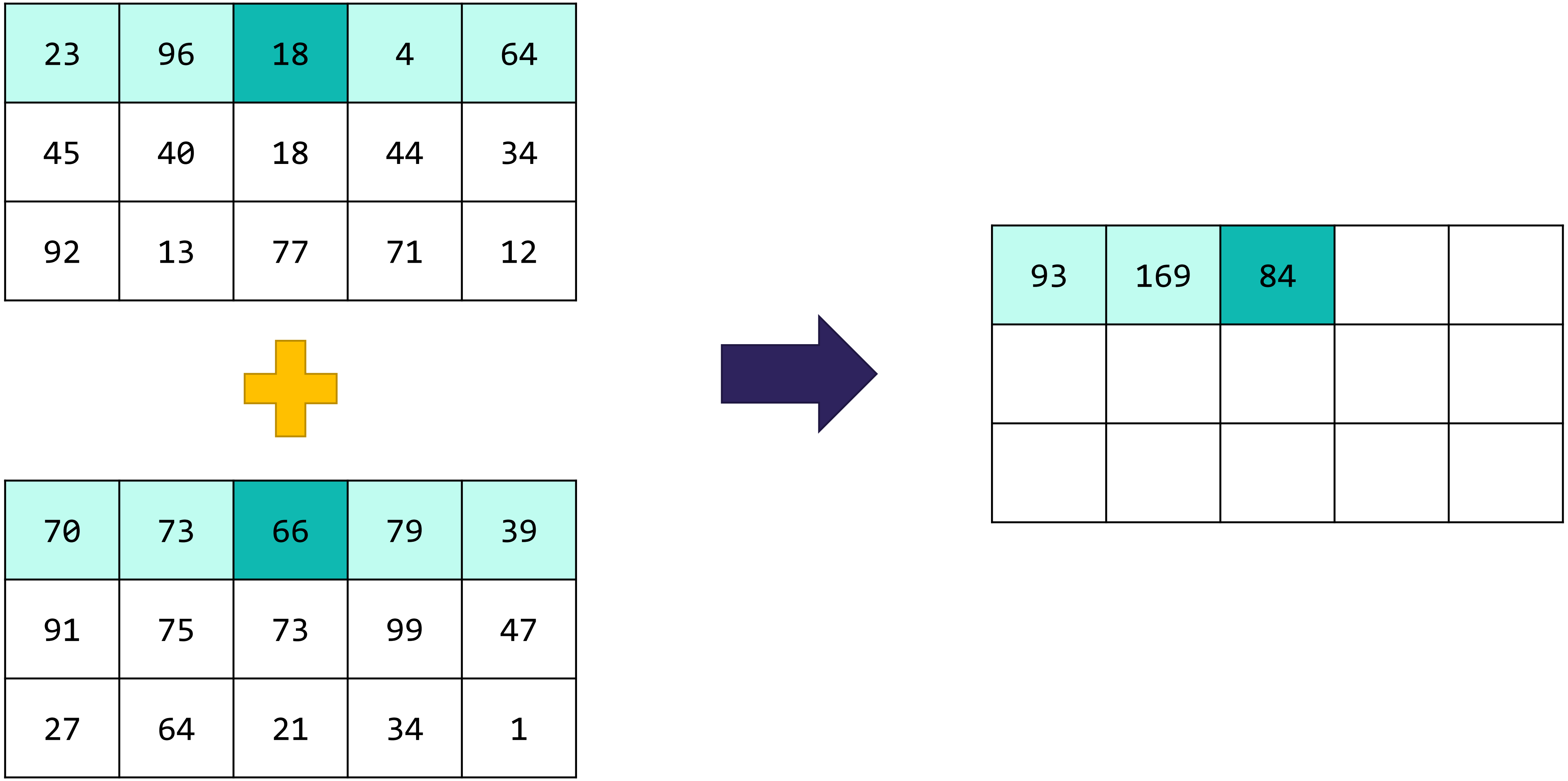


23	96	18	4	64
45	40	18	44	34
92	13	77	71	12

70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

93	169			

matrixAdd



23	96	18	4	64
45	40	18	44	34
92	13	77	71	12

70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

93	169	84		

matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12

+

70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

→

93	169	84	83	

matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12

+

70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

→

93	169	84	83	103

matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12

+

70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

→

93	169	84	83	103
136				

matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12

+

70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

→

93	169	84	83	103
136	115			

matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12

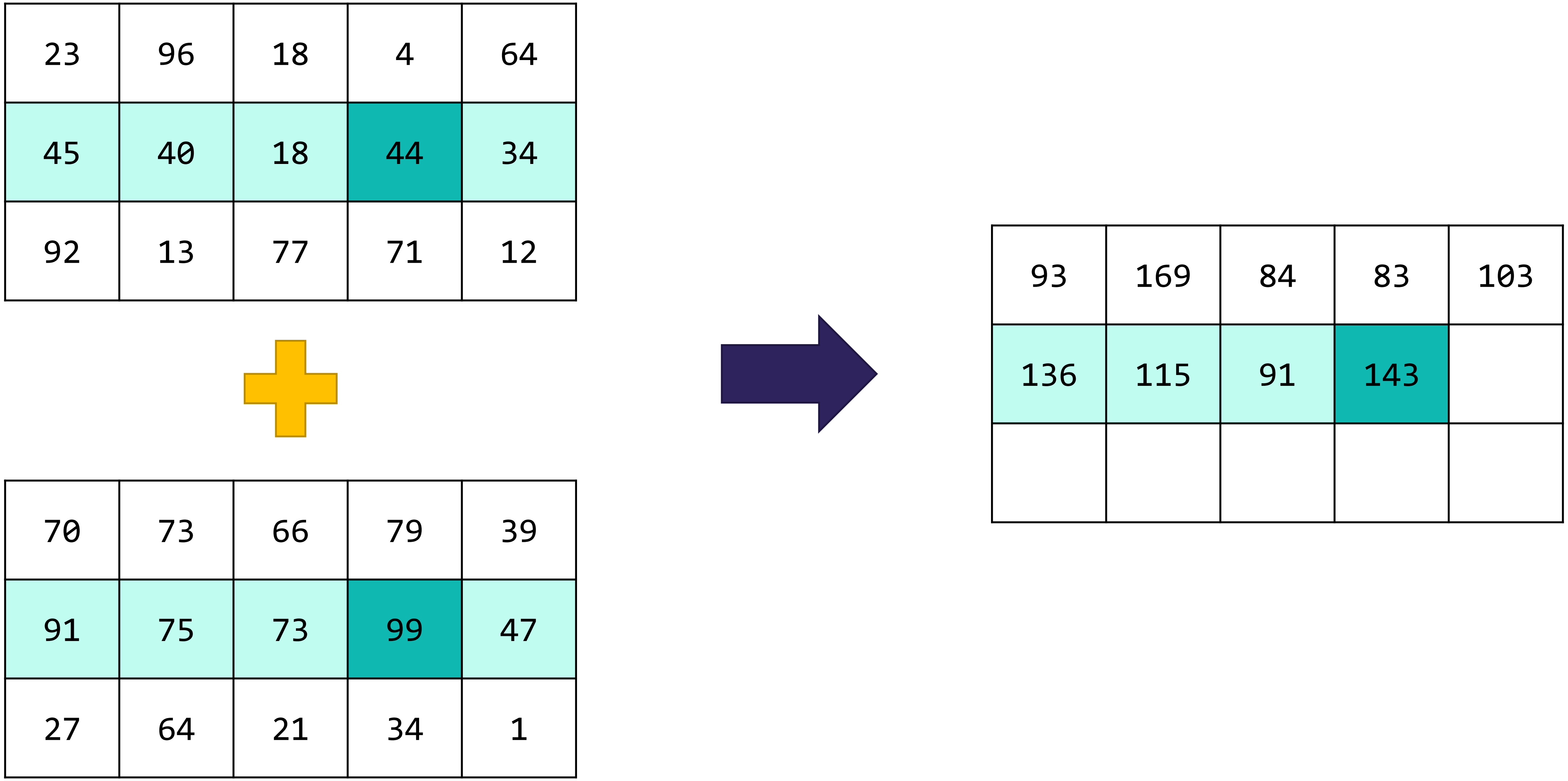
+

70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

→

93	169	84	83	103
136	115	91		

matrixAdd

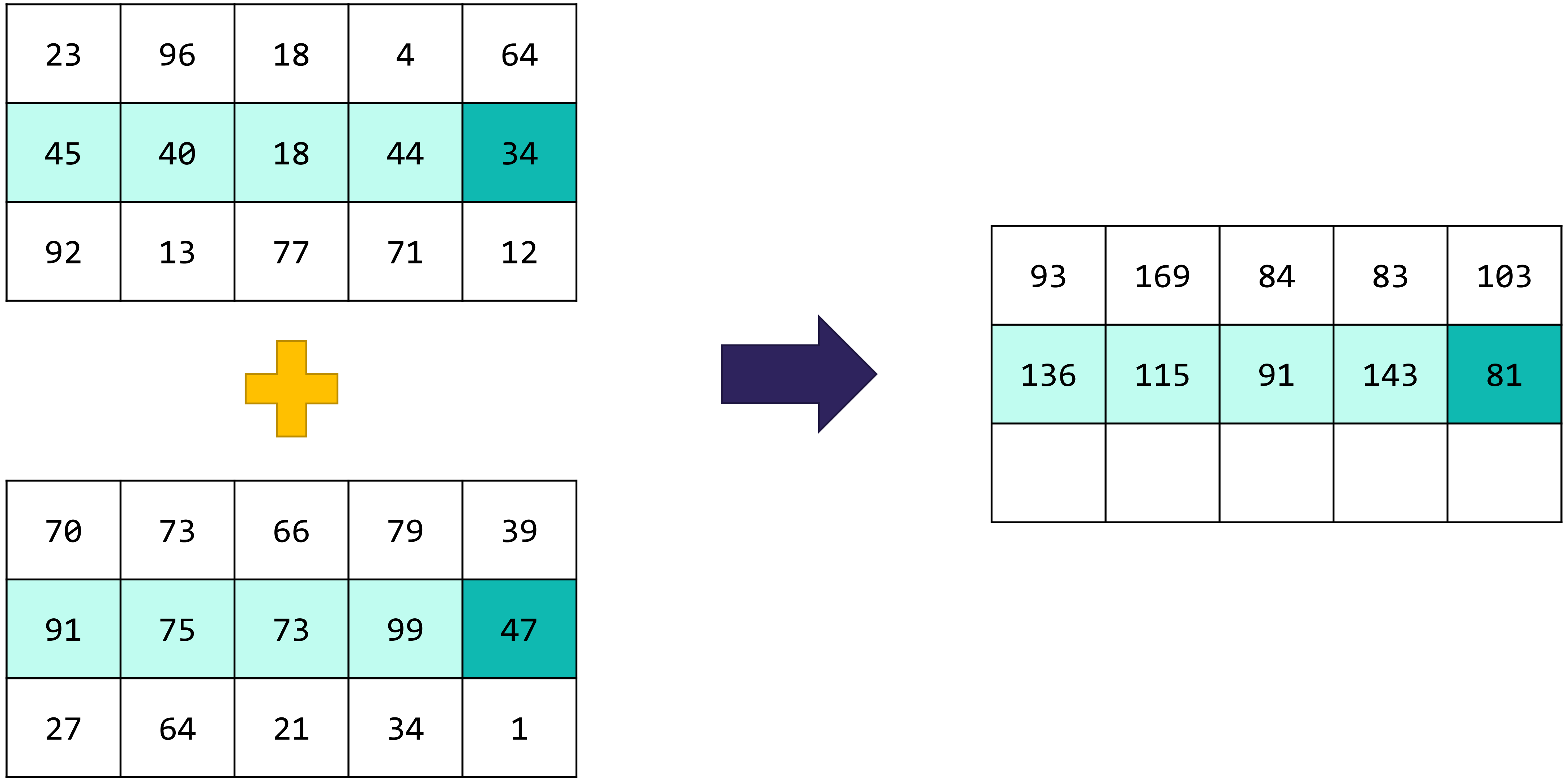


23	96	18	4	64
45	40	18	44	34
92	13	77	71	12

70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

93	169	84	83	103
136	115	91	143	

matrixAdd



23	96	18	4	64
45	40	18	44	34
92	13	77	71	12

+

70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

→

93	169	84	83	103
136	115	91	143	81

matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12

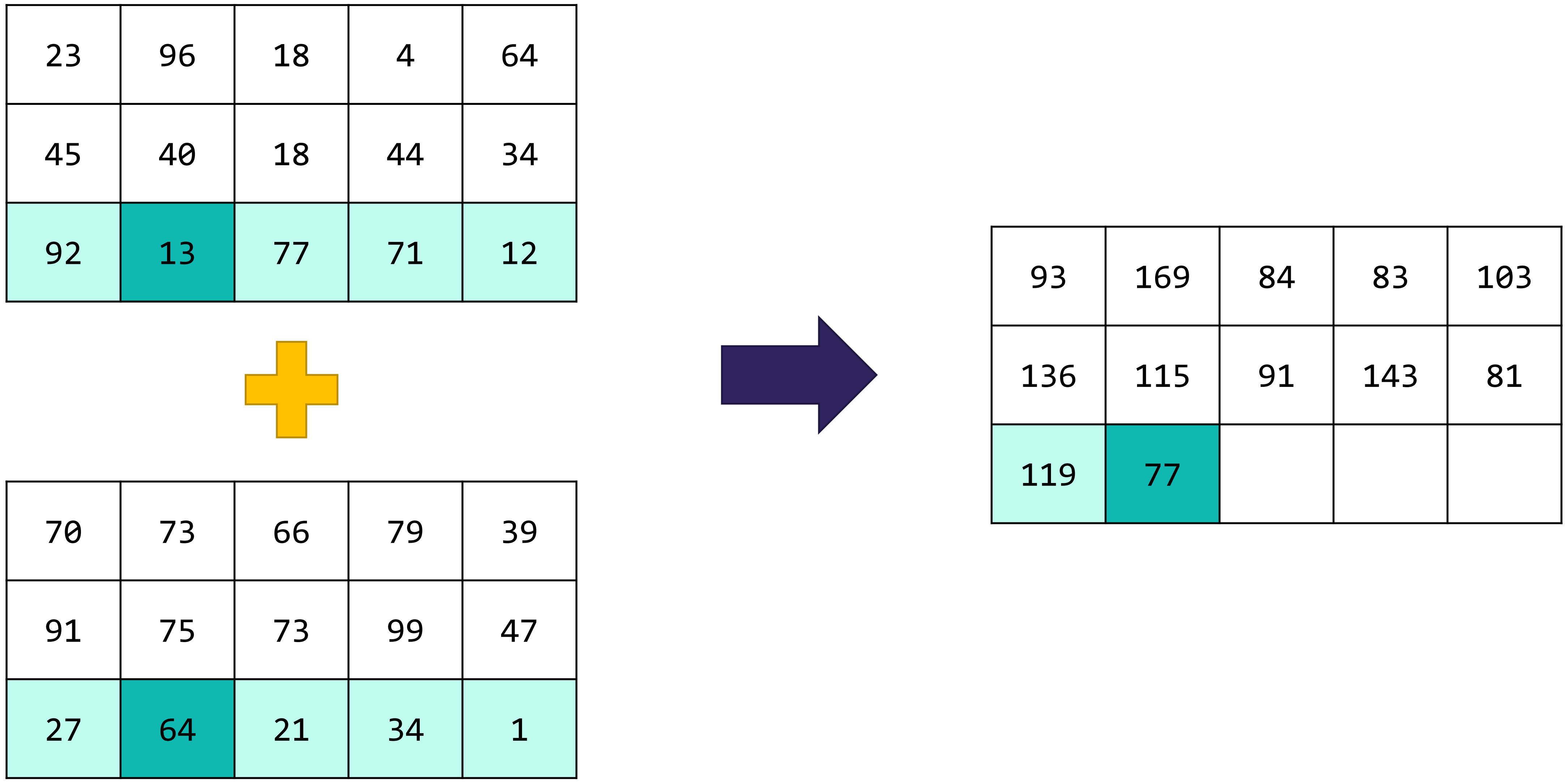
$+$

70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

$\rightarrow$

93	169	84	83	103
136	115	91	143	81
119				

matrixAdd

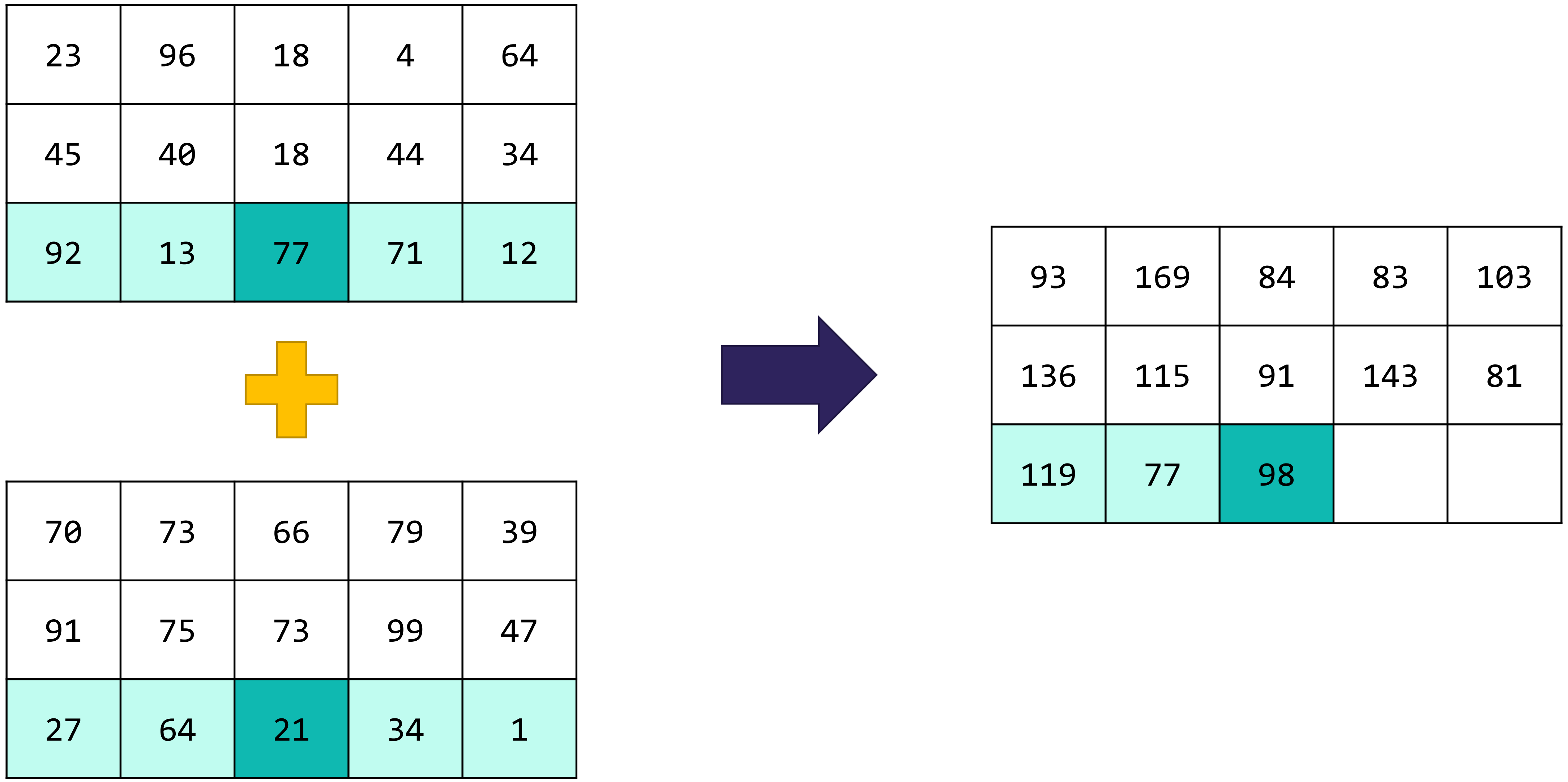


23	96	18	4	64
45	40	18	44	34
92	13	77	71	12

70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

93	169	84	83	103
136	115	91	143	81
119	77			

matrixAdd



23	96	18	4	64
45	40	18	44	34
92	13	77	71	12

70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

93	169	84	83	103
136	115	91	143	81
119	77	98		

matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12

+

70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

→

93	169	84	83	103
136	115	91	143	81
119	77	98	105	

matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12

70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

93	169	84	83	103
136	115	91	143	81
119	77	98	105	13

matrixAdd

23	96	18	4	64
45	40	18	44	34
92	13	77	71	12

70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

70	73	66	79	39
91	75	73	99	47
27	64	21	34	1

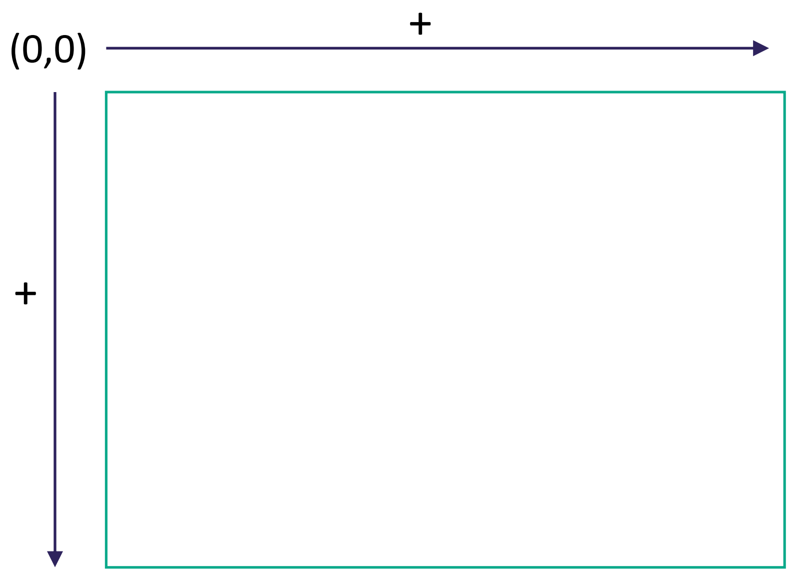
Lecture Outline

- Announcements
- 2D Arrays Review
- **Images** ◀
- Images with 2D Arrays!

Images

From the computer's perspective, images are just a big grid of values called **pixels**.

Each pixel shows a different color based on a specified value.



Images

If images are just grids of pixels, and we can think of 2D arrays as grids,

We can represent images as 2D arrays of pixels!

Further, since each pixel is shown as a specific color,

We can represent images as 2D arrays of colors!

Images in Java

- `Picture.java`
  - Represents the idea of a picture in your program
- `Color.java`
  - Represents colors in your program!
  - Uses the RGB color scheme where each color is made up of some amount (0-255) of red, green, and blue

Images in Java: Picture.java

```
Picture pic = new Picture("gumball.png");
```

Methods	Descriptions
`pic.getPixels();`	Returns a `Color[][]` representing the colors in the grid of pixels.
`pic.setPixels(colorArray);`	Sets the grid of pixels in the picture based on the given `colorArray`.
`pic.save(fileName);`	Saves the current picture to a file with the given `fileName`.
`pic.show();`	Shows the current picture in a window on the screen.*

* This functionality doesn’t work perfectly on Ed, it’s probably easier to use the save() method!

Images in Java: `Color.java`

```
Color color = new Color(redVal, greenVal, blueVal);
```

Methods	Descriptions
`color.getRed();`	Returns the color amount for red.
`color.getGreen();`	Returns the color amount for green.
`color.getBlue();`	Returns the color amount for blue.

Lecture Outline

- Announcements
- 2D Arrays Review
- Images
- **Images with 2D Arrays!**