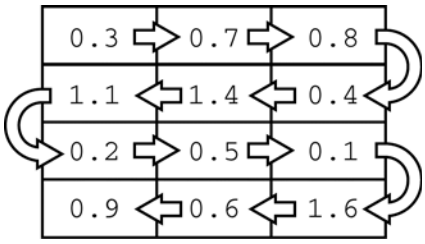
4. A telescope scans a rectangular area of the night sky and collects the data into a 1-dimensional array. Each data value scanned is a number representing the amount of light detected by the telescope. The telescope scans back and forth across the sky (alternating between left to right and right to left) in the pattern indicated below by the arrows. The back-and-forth ordering of the values received from the scan is called *telescope order*.



The telescope records the data in telescope order into a 1-dimensional array of `double` values. This 1-dimensional array of information received from a single scan will be transferred into a 2-dimensional array, which reconstructs the original view of the rectangular area of the sky. This 2-dimensional array is part of the `SkyView` class, shown below. In this question you will write a constructor and a method for this class.

```
public class SkyView
{
  /** A rectangular array that holds the data representing a rectangular area of the sky. */
  private double[][] view;

  /** Constructs a SkyView object from a 1-dimensional array of scan data.
   *  @param numRows the number of rows represented in the view
   *         Precondition: numRows > 0
   *  @param numCols the number of columns represented in the view
   *         Precondition: numCols > 0
   *  @param scanned the scan data received from the telescope, stored in telescope order
   *         Precondition: scanned.length == numRows * numCols
   *  Postcondition: view has been created as a rectangular 2-dimensional array
   *                 with numRows rows and numCols columns and the values in
   *                 scanned have been copied to view and are ordered as
   *                 in the original rectangular area of sky.
   */
  public SkyView(int numRows, int numCols, double[] scanned)
  {  /* to be implemented in part (a) */  }

  /** Returns the average of the values in a rectangular section of view.
   *  @param startRow the first row index of the section
   *  @param endRow the last row index of the section
   *  @param startCol the first column index of the section
   *  @param endCol the last column index of the section
   *  Precondition: 0 <= startRow <= endRow < view.length
   *  Precondition: 0 <= startCol <= endCol < view[0].length
   *  @return the average of the values in the specified section of view
   */
  public double getAverage(int startRow, int endRow,
                           int startCol, int endCol)
  {  /* to be implemented in part (b) */  }

  // There may be instance variables, constructors, and methods that are not shown.
}
```

(a) Write the constructor for the `SkyView` class. The constructor initializes the `view` instance variable to a 2-dimensional array with `numRows` rows and `numCols` columns. The information from `scanned`, which is stored in the telescope order, is copied into `view` to reconstruct the sky view as originally seen by the telescope. The information in `scanned` must be rearranged as it is stored into `view` so that the sky view is oriented properly.

For example, suppose `scanned` contains values, as shown in the following array.

	0	1	2	3	4	5	6	7	8	9	10	11
scanned	0.3	0.7	0.8	0.4	1.4	1.1	0.2	0.5	0.1	1.6	0.6	0.9

Using the `scanned` array above, a `SkyView` object created with `new SkyView(4, 3, scanned)`, would have `view` initialized with the following values.

view	0	1	2
0	0.3	0.7	0.8
1	1.1	1.4	0.4
2	0.2	0.5	0.1
3	0.9	0.6	1.6

For another example, suppose `scanned` contains the following values.

	0	1	2	3	4	5
scanned	0.3	0.7	0.8	0.4	1.4	1.1

A `SkyView` object created with `new SkyView(3, 2, scanned)`, would have `view` initialized with the following values.

view	0	1
0	0.3	0.7
1	0.4	0.8
2	1.4	1.1

Complete the `SkyView` constructor below.

```
/** Constructs a SkyView object from a 1-dimensional array of scan data.
 *  @param numRows the number of rows represented in the view
 *         Precondition: numRows > 0
 *  @param numCols the number of columns represented in the view
 *         Precondition: numCols > 0
 *  @param scanned the scan data received from the telescope, stored in telescope order
 *         Precondition: scanned.length == numRows * numCols
 *  Postcondition: view has been created as a rectangular 2-dimensional array
 *                 with numRows rows and numCols columns and the values in
 *                 scanned have been copied to view and are ordered as
 *                 in the original rectangular area of sky.
 */
public SkyView(int numRows, int numCols, double[] scanned)
```

Part (b) begins on page 19.

(b) Write the `SkyView` method `getAverage`, which returns the average of the elements of the section of `view` with row indexes from `startRow` through `endRow`, inclusive, and column indexes from `startCol` through `endCol`, inclusive.

For example, if `nightSky` is a `SkyView` object where `view` contains the values shown below, the call `nightSky.getAverage(1, 2, 0, 1)` should return `0.8`. (The average is `(1.1 + 1.4 + 0.2 + 0.5) / 4`, which equals `0.8`). The section being averaged is indicated by the dark outline in the table below.

view	0	1	2
0	0.3	0.7	0.8
1	1.1	1.4	0.4
2	0.2	0.5	0.1
3	0.9	0.6	1.6

Class information repeated from the beginning of the question

```
public class SkyView

private double[][] view
public SkyView(int numRows, int numCols, double[] scanned)
public double getAverage(int startRow, int endRow,
                         int startCol, int endCol)
```

WRITE YOUR SOLUTION ON THE NEXT PAGE.

Complete method `getAverage` below.

```
/** Returns the average of the values in a rectangular section of view.
 *  @param startRow the first row index of the section
 *  @param endRow the last row index of the section
 *  @param startCol the first column index of the section
 *  @param endCol the last column index of the section
 *  Precondition: 0 <= startRow <= endRow < view.length
 *  Precondition: 0 <= startCol <= endCol < view[0].length
 *  @return the average of the values in the specified section of view
 */
public double getAverage(int startRow, int endRow,
                         int startCol, int endCol)
```

STOP

END OF EXAM