

[Start Assignment](#)

**Due** Monday by 11:59pm    **Points** 41    **Submitting** a file upload    **File Types** java  
**Available** Mar 10 at 12am - Apr 19 at 11:59pm about 1 month

## Unit 6: Recursion &amp; Java Generics

8 of 8

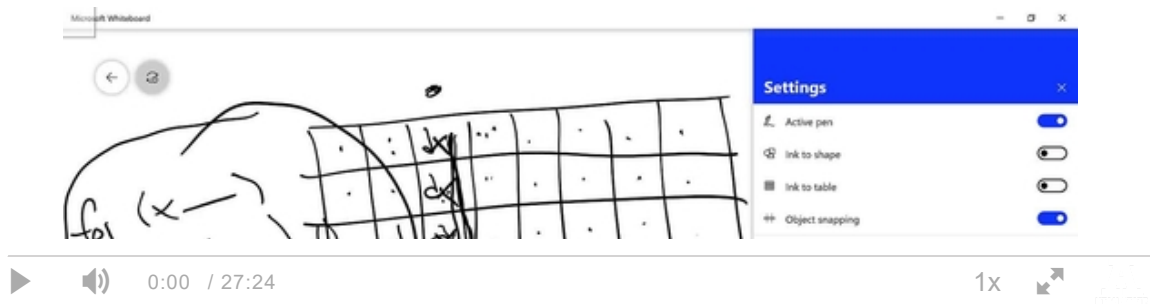
## 8 Assignment 8 - Recursion Exercises

Computer  
SCIENCE

**IMPORTANT:** Please review the statement on Academic Integrity. Submitting code, without permission, that is not your own may result in an automatic F for the class: [Statement on Academic Integrity](#)

In this assignment you'll write code to solve various problems using recursion. The purpose of this assignment is to help you learn to think and solve problems recursively. All of these problems could be solved using iteration (any recursive solution can be performed using iteration), but recursion often times provides an *easier* or more *elegant* solution.

Watch the following video for a detailed discussion on important details for this assignment...



# Assignment

Write a program that includes the following recursive methods:

- Test a string to determine if it is word symmetric.
  - Signature: `public static boolean isWordSymmetric(String[] words, int start, int end)`
  - Description: Similar to a palindrome, a phrase is symmetric if the words placed forward or backwards say the same thing. For example, "Fall leaves as soon as leaves fall" is word symmetric.
  - Write a recursive method that determines if the phrase is word symmetric. To solve this, you'll need to split it into the individual words before calling the recursive method. Inside the recursive method be sure to do the word comparison while ignoring the case of the string.
  - Please note that the *end* argument represents the last array index to check.
- Return the sum of all elements in a array.
  - Signature: `public static long arraySum(int[] data, int position)`
  - Description: Write a recursive method that returns the total of all elements in the array.
- Return the minimum value in an unordered array.
  - Signature: `public static int arrayMin(int[] data, int position)`
  - Description: Write a recursive function that returns the minimum value of the elements in the array. You may assume there is at least one value in the array.
- Compute the weights supported at each object in a pyramid.
  - Signature: `public static double computePyramidWeights(double[][] weights, int row, int column)`
  - Description: Consider the following pyramid of objects...

```
A
B C
D E F
G H I J
```

- The weight supported at each object is the weight of the object itself, plus half of the supported weight of the objects above it.
  - The weight supported by A is the weight of A itself.
  - The weight supported by B is the weight of B plus 1/2 the weight of A.
  - The weight supported by C is the weight of C plus 1/2 the weight of A.
  - The weight supported by E is the weight of E itself, plus 1/2 of the weight supported by B and 1/2 of the weight supported by C.
- The method must be able to handle incorrect row or column arguments
  - The method should never leave the bounds of each row or column
- The data is given to you in the following format (there may be any number of rows, this example shows only 4).

```
0 1 2 3 (col)
0 A
1 B, C
```

```

2  D, E, F
3  G, H, I, J
(row)

```

- Determine the number of organisms in an image (2d array), and label each organism.
  - Signature: `public static int howManyOrganisms(char[][] image)`
  - Description: Given a 2d array of characters, where an asterisk (\*) marks a non-empty cell. An organism is defined as all cells connected directly to other cells in the up/down/left/right (not diagonal) directions.
    - The method signature above is **not** a recursive method. Instead, it iterates through the image and as soon as it encounters a new organism, **it calls a recursive method** (that you define) that labels the newly found organism.
    - Look at the provided driver code to see how the organism is stored and compare that with the output of the labeled organisms below. In the original image, the organisms are identified by asterisks (\*), but the resulting image has each organism labeled with a different character in the alphabet.
    - The image is NOT guaranteed to be rectangular (i.e. it may be ragged).

## Example Driver Output

The provided driver code produces the following output...

```

Array Sum: 232
Array Min: -42

You can cage a swallow can't you but you can't swallow a cage can you
Is word symmetric: true

I still say cS 1410 is my favorite class
Is word symmetric: false

--- Weight Pyramid ---
51.18
81.49 156.84
109.79 252.83 211.24
108.33 320.92 366.09 227.25


--- Labeled Organism Image ---
aa      b
a       b
        cc
d  ccc
dd c c c
dd ccccc
      c
      c
    eee c
      e c

There are 5 organisms in the image.

```



## Notes & Submission

- Turn in the following Java source file: **Recursion.java**
- You are provided with a skeleton project to work from that has JUnit setup and includes the unit tests for this assignment. The project can be found at this [link](#)  [\\_ \(https://usu.instructure.com/courses/639065/files/80136227/download?download\\_frd=1\)](https://usu.instructure.com/courses/639065/files/80136227/download?download_frd=1).
- Your code must compile without any warnings or compiler errors. See syllabus regarding code that has compiler errors.
- Your code must adhere to the CS 1410 coding standard: [link](#)
- Java SDK Docs: [link](#) [\\_ \(https://docs.oracle.com/en/java/javase/15/\)](https://docs.oracle.com/en/java/javase/15/).

### Assignment 8 - Rubric

Criteria	Ratings		Pts
Compiles without any warnings	<b>2 pts</b> <b>Full Marks</b>	<b>0 pts</b> <b>No Marks</b>	2 pts
Follows required course code style	<b>2 pts</b> <b>Full Marks</b>	<b>0 pts</b> <b>No Marks</b>	2 pts
Passes all unit tests	<b>2 pts</b> <b>Full Marks</b>	<b>0 pts</b> <b>No Marks</b>	2 pts
isWordSymmetric	<b>5 pts</b> <b>Full Marks</b>	<b>0 pts</b> <b>No Marks</b>	5 pts
arraySum	<b>5 pts</b> <b>Full Marks</b>	<b>0 pts</b> <b>No Marks</b>	5 pts
arrayMin	<b>5 pts</b> <b>Full Marks</b>	<b>0 pts</b> <b>No Marks</b>	5 pts
computePyramidWeights	<b>10 pts</b> <b>Full Marks</b>	<b>0 pts</b> <b>No Marks</b>	10 pts
howManyOrganisms counting & labeled output	<b>10 pts</b> <b>Full Marks</b>	<b>0 pts</b> <b>No Marks</b>	10 pts
			<b>Total Points: 41</b>