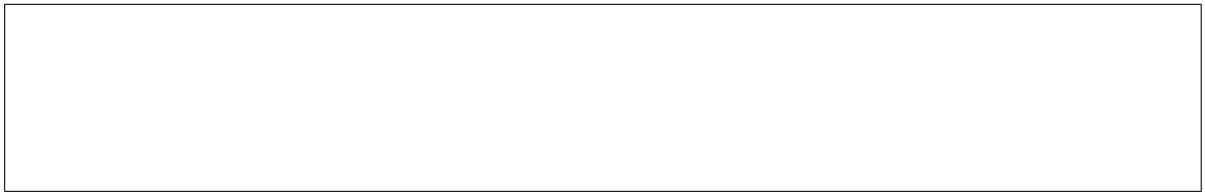
COS125 - Precept 7 (Arrays II)

1 Code Tracing

For each box, draw out a visualization of how all array memory is represented at each point during the execution of a program. Then write down what prints at the end of each code snippet.

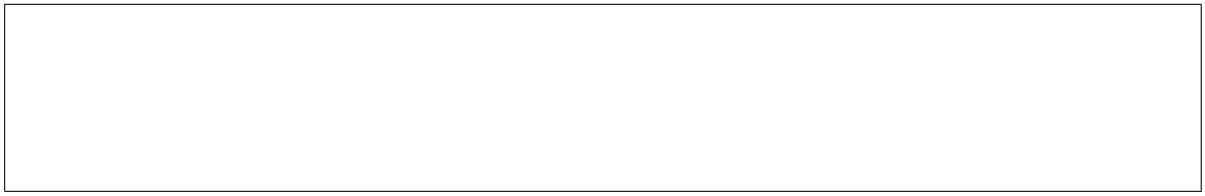
```
1. int[] a = new int[4];
2. int[] b = {3,5,4};
3. System.out.println(a == b);
```

```
4. b = a;
5. a[2] = 125;
6. System.out.println(a == b);
```



```
7. b = new int[5];
8. System.out.println(a[2] + ", " + b[2]);
```

```
int[][] c = {a, b};
```



2 Matrix Transpose

Write a program `MatrixTranspose.java` that starts out with the following 3×3 matrix

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

and transposes it, obtaining the result

$$A^T = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}.$$

Notice that, to transpose a matrix, you reflect the matrix along its diagonal. Equivalently, you can write the rows of the original matrix as columns.

More concretely, the element at row i and column j in matrix A^T corresponds to element at row j and column i in matrix A. Formally,

$$[A^T]_{ij} = [A]_{ji}.$$

3 Digital Numbers

3.1 Digital Display

Digital clocks often display each digit in the current time as a grid of 7 characters. We can recreate this with a 3 x 3 grid. For example, the number two can be written as:

```
 _
 _|
|_
```

In Java, we can represent this grid in a 2D array (only TWO shown for brevity):

```
char EMPTY = ' ';
char HORIZONTAL = '_';
char VERTICAL = '|';
char [][] TWO = new char[][]{{EMPTY, HORIZONTAL, EMPTY}, {EMPTY, HORIZONTAL, VERTICAL},
     {VERTICAL, HORIZONTAL, EMPTY}};
```

Given the starter code with all hardcoded numbers from 0-9 in 2D arrays, write a program `DigitalDisplay.java` that takes one integer number and displays this number in digital clock format on the same line. (Hint: You may want to use an array to store each digital-clock represented digit.)

```
~/cos125/$ javac-introcs DigitalDisplay.java
~/cos125/$ java-introcs DigitalDisplay 123
    _  _
 |  _| _|
 | |_  _|
```

3.2 Digital Clock

Write a program `DigitalClockDifferences.java` that takes in four integers, a pair of two-digit hour and minutes, each representing two different times that might display on a clock. Think of each character or lack of character as a binary off/on switch. Your program must print out the minimum number of binary switches (i.e., number of characters that must be changed) the clock must make to convert the first hour/minutes pair into the second hour/minutes pair.

```
~/cos125/$ javac-introcs DigitalClockDifferences.java
~/cos125/$ java-introcs DigitalClockDifferences 12 01 12 02
4
~/cos125/$ java-introcs DigitalClockDifferences 12 59 12 60
3
```

4 Wordle Revisited

In the Conditionals precept, you made a simplified Wordle game. In this precept, you will make your Wordle game closer to the actual playable game with a number of improvements.

4.1 Wordle Advanced

Write a program called `WordleAdvanced.java` that takes in a word as input and compares it to a secret hard-coded 5-letter word of your choice. This time, instead of just printing letters that are in the correct spot, we additionally want to print letters that appear in the word, but are in the wrong spot. For instance, assuming the secret word is "SHARE":

```
~/cos125/$ javac-introcs WordleAdvanced.java
~/cos125/$ java-introcs WordleAdvanced EATEN
EATEN
-----
xx x

~/cos125/$ java-introcs WordleAdvanced LEASE
LEASE
```

```
-----
 xYxY
```

Your program should print out the letter 'x' if the letter appears in the secret word, but is in the wrong spot. It should print 'Y' if the letter is in the correct spot. Notice that this leads to two 'E's in "LEASE" being marked despite the secret word only containing one instance of 'E'.

4.2 Randomizing Secret Words

Playing wordle with one hard-coded secret word you already know is not fun. Create a hard-coded array of 5-letter words in your program. Then you can try a few methods to "randomize" the word choice. One method you can use is selecting a word with `Math.random()`. Another method is to take in any integer and use the remainder operator (%) to select one of the words in the array. What are the advantages and disadvantages of each?

4.3 Bonus: Wordle Hard

In Subsection 4.1 our program marked repeat letters as valid letters no matter how many times the letter showed up in the secret word. Now you will try to remove repeats in the program `WordleHard.java`. A letter should only be marked if it is a) in the correct spot, or b) not already accounted for in the secret word and is in the wrong spot.

```
~/cos125/$ javac-introcs WordleHard.java
~/cos125/$ java-introcs WordleHard LEASE
LEASE
-----
  YxY
```

Notice that the 'E' that was in the correct spot took priority as the marked letter over the 'E' that came before it which was in the wrong spot.

5 Bonus: Color Contrast

Arrays along with loops help to condense repeated code. Revisit your original ColorContrast.java program from the Conditionals assignment and see if you can *refactor* the code with arrays. What code can you refactor with arrays? What code cannot be refactored? Which version of the program do you prefer (i.e., with or without arrays)?