LEC 16

CSE 121

Array Patterns

BEFORE WE START

Talk to your neighbors:

What are your spring break plans?

Music: ❄ CSE 121 26wi Lecture Tunes ❄

Instructors: Miya Natsuhara

TAs:
Amogh
William
Johnathan
Anant
Reese
Hayden
Aki
Spencer
Savannah
Tamsyn
Anum
Abdul
Janvi
Navya
Ruslana
Sam
Ethan
Jessica
Paul
Carson
Shayna
Jesse
Minh
Cayden

Questions during Class?

Raise hand or send here

sli.do #cse121

Announcements, Reminders

- Quiz 2 in section **tomorrow (Thursday, March 5th)**!
  - Reference sheet and practice quizzes on Ed
  - Email Miya before our quiz section if you need to request a makeup
- P3 will be released tonight & due **Tuesday, March 10th at 11:59pm**
- R5 due tomorrow (eligible: **P1**, C2, P2)
  - P1 cycling out of eligibility after R5
- Final Exam on **Wednesday, March 18th, 12:30-2:20pm**
  - Left-handed desk request form out now, due Mon, March 9th
  - More details posted soon, and will be discussed on Friday

(PCM) Why Discuss Array Patterns?

- Arrays are important! This is our fourth lecture covering arrays
- Analogy: tools in toolbox
- Helpful for your future in programming

(PCM) Counting Elements that Meet a Condition

"one"	"two"	"three"	"six"	"seven"	"eight"	"ten"

```java
public static int evenLength(String[] list) {
    int countEven = 0;
    for (int i = 0; i < list.length; i++) {
        if (                         ) {
            countEven++;
        }
    }

    return countEven;
}
```

(PCM) Modifying Elements of an Array

4	8	15	16	23	42

```
public static void clamp(int min, int max, int[] list) {
    for (int i = 0; i < list.length; i++) {
        if (        > max) {
                    = max;
        } else if (        < min) {
                    = min;
        }
    }
}
```

(PCM) Searching for an Element

“one"	"two"	"three"	"six"	"seven"	"eight"	"ten"

```
public static int indexOfIgnoreCase(String phrase, String[] list) {
    for (int i = 0; i < list.length; i++) {
        if (                                    ) {
            return i;
        }
    }

    return -1;
}
```

(PCM) Shifting Elements

9.6	-88.0	4.815	0.009	7.0184	42.9

```
public static void rotateRight(double[] list) {
    double lastElement = list[list.length - 1];

    for (int i = list.length - 1; i > 0; i--) {
        list[i] = list[i - 1];
    }

                         ;

}
```

(PCM) Analyzing Multiple Elements in an Array (isPalindrome)

0	1	9	1	0

```
public static boolean isPalindrome(int[] list) {
    for (int i = 0; i < list.length / 2; i++) {
        if (list[i] != list[list.length - 1 - i]) {
            return      ;
        }
    }

    return      ;
}
```

(PCM) Analyzing Multiple Elements in an Array (isMirrored)

4.1	5.5	4.1
6.0	-1.1	6.0

```
public static boolean isMirrored(double[][] arr) {
    for (int i = 0; i < arr.length; i++) {
        int rowLength =                          ;
        for (int j = 0; j <                          ; j++) {
            if (arr[i][j] != arr[i][rowLength - 1 - j]) {
                return                ;
            }
        }
    }
    return                         ;
}
```

(PCM) Array of Counters

```
0 1 2 2 0 2
```

```
public static int[] numCount(Scanner input, int numPrompts) {
    int[] counts =           ;
    for (int i = 0; i < numPrompts; i++) {
        int num = input.nextInt();

                    ;

    }

    return counts;
}
```

(PCM) Your Turn!

- Review the problems in the Array Patterns PCM
  - On Slido, vote for any problems you would like to go over together!
- If you have time, try some new problems!
  - [NEW] rotateLeft: Shifting Elements problem
  - [NEW] isAllPairs: Analyzing Multiple Elements problem
  - [NEW] transpose: 2D Analyzing Multiple Elements/Shifting Elements problem

(PCM) Questions to Ask Ourselves

- “Are we looking at each element in the array, one at a time?”
  - Loop traversal

- “Are we changing elements in the array?”
  - Update the array at a specific index

- “Do we only want to do a task if a certain condition is true?”
  - Conditional(s)

rotateLeft

0.1	0.2	0.3	0.4	0.5

isAllPairs

3	3	4	4	5	5	6	6	7	7

-1	-1	0	0	4	8	8	9	10	10

transpose

1	2	3
4	5	6
7	8	9