All assignment work will be done in the included template files on Github this week. Please do not forget to fill out the metadata at the top of each template! At *least* the field of if you worked with anyone!

Get Assignment link: https://classroom.github.com/a/CWwP6zW9

1. *(Chapter 2, Exercise 8)*

   Here your goal is to write a function named `draw_console_pyramid` which takes a single argument (function input) of the pyramid height. Your function should then print to the screen a pyramid (constructed of `*` characters) of the specified height in which the width of each row increases by two as you move downward in the console (toward the base of the pyramid). Each of the rows should be centered with respect to the others, and the bottom line should begin at the left margin. Calling `draw_console_pyramid(8)` should thus produce the following:

```
› python Prob1.py
        *
       ***
      *****
     *******
    *********
   ***********
  *************
***************
```

   Note how on the bottom row the pyramid starts immediately, with no space between it and the left margin, and that each row of the pyramid is centered on the others, increasing by 2 each time.

   I've already set up the template to call this function to draw pyramids of height 8 for your testing purposes, but you should absolutely test it with other heights as well to ensure it is working as planned. *Hint: In order to keep things centered, you are going to need to add an amount of space characters to the front of most lines. Think about how you could assemble these strings, and how you'll know how many characters to add. Writing some helper functions can be useful!*

2. *(From Chapter 3, Exercise 6)*

   Your task here is to write a Python function called `largest_two` that reads in integers from a *user's input* up until they enter a blank line, at which point the program should print out both the largest and second-largest values from all the integers that the user had input. You can assume at least two integers will always be entered. For example, running the program could look like:

```
› python Prob1.py

This program finds the two largest integers.
Enter a blank line to stop.
  ? 223
  ? 251
  ? 317
  ? 636
  ? 766
  ? 607
  ? 607
  ?
The largest value is 766.
The second-largest value is 636.
```

   The values of this sample run are the number of pages in the British hardcover editions of J. K. Rowling's *Harry Potter* series. The output tells us that the longest book is *Harry Potter and the Order of the Phoenix* at 766 pages, and the second-longest book is *Harry Potter and the Goblet of Fire* at 636 pages.

   You should write your program so that it exactly duplicates the format above, with instructions, prompt, and final output looking the same. You can (and should!) of course enter in whatever numbers you like when testing it (including negative numbers)!

   Hint: While you could keep track of all the numbers entered by the user, you only ever really need to keep track of two: the largest and second largest...

3. (a) As you will soon discover, implementing the Wordle project requires you to take account of the possibility that a word might contain more than one instance of the same letter. In preparation of that, your task here is to write a predicate function called `contains_repeated_letters(word)`, that returns `True` if any of the characters in the word appear more than once, not necessarily consecutively. Thus,

```
contains_repeated_letters("single")
```

should return `False`, as no letter ever shows up twice. Conversely,

```
contains_repeated_letters("repeating")
```

should return `True`, as the letter `e` shows up twice. Test that this function is working as intended before moving on to the next part!

(b) Now use `contains_repeated_letters` together with the `english.py` library to write a function called `longest_no_repeats` that finds and returns the longest word in the English dictionary that contains no repeated letters.