Simple Functions:
You will write a few python functions for practice with the language. In your submission file, include a comment at the top with your names, section, GTID/Email, and your collaboration statement. Also include each of the following functions:

1. letterGrade
2. countLetter
3. eyeForI
4. wordMirror
5. encryption
6. guessPassword
7. countDown
8. numberBowTie
9. printTimes
Function Name: **letterGrade** (10pts)

Parameters:
- `grade` – an integer/float representing the numerical grade

Return:
- A string – ‘You made a(n) [letter grade].’

Test Cases:
- `letterGrade(92)` returns: “You made a(n) A.”
- `letterGrade(72.5)` returns: “You made a(n) C.”
- `letterGrade(43)` returns: “You made a(n) F.”

Description:
Write a function, using conditionals, that determines the letter grade from the numerical grade, passed in as a parameter. The letter grade is determined by the following: [90, 100] - A, [80, 90) - B, [70, 80) - C, [60, 70) - D, [0, 60) - F (Assume the numerical grade argument is between 0 and 100.) After converting the numerical grade to a letter grader, return the exact string, ‘You made a(n) [letter grade].’

Function Name: **countLetter** (10pts)

Parameters:
- `aWord` – a string representing a word
- `aLetter` – a string representing the character to count in `aWord`

Return:
- `count` – integer representing the number of times `aLetter` appears in `aWord`

Test Cases:
- `countLetter("There are many cows in the field", "e")` returns 5
- `countLetter("Where is Waldo?", "w")` returns 0
- `countLetter("How much wood can a woodchuck chuck?", "c")` returns 6

Description:
Write a function that takes in a word and letter as parameters. The function will then count the number of times that the letter appears in the word, using a loop. Remember that capital letters are DIFFERENT from lower case letters! Return the number of times that the letter appears in the word or sentence as an integer.

Function Name: **eyeForI** (10pts)

Parameters:
- `aString` – any string

Return:
- The modified string

Test Cases:
- `eyeForI("William")` returns "Weyelleyeam"
- `eyeForI("I do not like physics.")` returns "eye do not leyeke physeyecs."
- `eyeForI("This is so much fun!")` returns "Theyes eyes so much fun!"

Description:
Write a function that takes in a string as a parameter. Replace every “I” and “i” in the string with “eye” and return the resulting string. The function MUST use a for-loop or while loop to build up the new string while processing the original string letter by letter, not take advantage of the built in str.replace function.
Function Name: wordMirror (10pts)
Parameters:
   aString – any string
Return:
   The mirrored string (reversed)
Test Cases:
   wordMirror("CS 1301") returns “CS 13011031 SC”
   wordMirror("Atlanta") returns “AtlantaatnaltA”
   wordMirror("georgiatech") returns “georgiatechhcetaigroeg”
Description:
   Write a function that takes in a string as its only parameter. You will then return a new string that is the original string concatenated with the reflection of the original string (reverse the string). It may be helpful to use a for-loop to reflect (reverse) the string. Hint: Add each letter to the FRONT of the new string you are building up!

Function Name: encryption (10pts)
Parameters:
   aString – a string that you want to encrypt
Return:
   None
Test Cases:
   encryption("I love cs 1301") prints “The encrypted code is: I 1o\(^\() c$ 1301”
   encryption("Why is it freezing?") prints “The encrypted code is: W#y i$ it f+()zelfing ?”
   encryption("I play the saxophone") prints “The encrypted code is: I p1@y t#() $@*op#on()”
Description:
   Write a function that encrypts a message into a secret code. The encryption should reflect the following conversions:
   1. a - @
   2. e - ()
   3. h - #
   4. l - 1 (lowercase-L to number one)
   5. r - + (plus sign)
   6. s - $
   7. v - ^
   8. x - *
You may use a loop to complete the following function, or you may find the .replace method in the string module useful. After encrypting the message, print “The encrypted code is: [encrypted string]”.
Function Name: **guessPassword** (10pts)
Parameters:
    password – a string that represents the secret password to be guessed.
Return:
    None
Test Case:
>>> guessPassword("abc123")

>>> Incorrect Password!

>>> Incorrect Password!

>>> You entered the correct password!

Description:
Write a function that uses `input` to prompt the user to guess the password that is passed into the function as a parameter. The function will continuously prompt the user to guess the password until the correct password has been entered. The function should prompt the user for the password: “Please enter the password: “. If the user enters an incorrect password, print “Incorrect password!” and prompt the user again for the password. When the user correctly enters the password into the prompt, print “You entered the correct password!” You **MUST use a while-loop or recursion!**

Function Name: **countDown** (10pts)
Parameters:
startNum – an integer that is the starting number to count down from
countBy– an integer that is the number you count down by

Return:
None

Test Cases:
>>> countDown(10, 2)
  10
  8
  6
  4
  2
  Blast Off!

>>> countDown(23, 7)
  23
  16
  9
  2
  Blast Off!

>>> countDown(21, 4)
  21
  17
  13
  9
  5
  1
  Blast Off!

Description:
Write a function to count down from the first parameter (startNum) by the second parameter (countBy). The function should print the numbers from the given number to 1 (decreasing by the second parameter each time...if you go past 1, don’t print it!) in descending order, with each number being printed on its own line. After printing the required numbers, on a separate line, print the string ‘Blast off!’

Function Name: numberBowTie (10pts)
Parameters:
  aNum – an integer (between 2 – 9) specifying ½ of the length of the longest row in the bow tie
Return:
  None
Test Cases:
  >>> numberBowTie(5)
    1    1
Description:
Write a function that takes in half the number of rows of the bow tie as a parameter. The function will then draw a number bow tie on screen using the `print` function. See screenshots above in the test cases for clarification. **DO NOT HARD CODE THE 8 different printouts**, you should have one set of code that will work for any number between 2 and 9. In order to correctly code this function, the spacing between elements in a row must be calculated mathematically.

**Function Name:** `printTimes` *(20pts)*

**Parameters:**
- `start` – an integer that limits the LOWER bound of the times table (inclusive)
- `end` – an integer that limits the UPPER bound of the times table (inclusive)
- `inc` – a positive integer (less than the end value)

**Return:**
None

**Description:**
Write a `printTimes(start, end, increment)` function that will print a times table from ‘start’ up to ‘end' by increments of ‘inc’, for any positive number. Note that your function must print a header (Times: start...end) and a first column number that goes from ‘start’...‘end’, while the interior of the grid is the X * Y value. Hint: Using two loops (one inside of the other) is an easy (but not the only) way to accomplish this. You may want to use tab characters ( "\t") to space your grid out correctly.

Test Cases:

```python
>>> printTimes(3, 34, 4)
Times:  3  7  11  15  19  23  27  31
     3  9  21  33  45  57  69  81  93
     7 21  49  77 105 133 161 189 217
    11 33  77 121 165 209 253 297 341
    15 45 105 165 225 285 345 405 465
    19 57 133 209 285 361 437 513 589
    23 69 161 253 345 437 529 621 713
    27 81 189 297 405 513 621 729 837
    31 93 217 341 465 589 713 837 961
```

```python
>>> printTimes(23, 45, 2)
Times:  23  25  27  29  31  33  35  37  39  41  43  45
   23  529  575  621  667  713  759  805  851  897  943  989  1035
   25  575  625  675  725  775  825  875  925  975 1025 1075 1125
   27  621  675  729  783  837  891  945  999 1053 1107 1161 1215
   29  667  725  783  841  899  957 1015 1073 1131 1189 1247 1305
   31  713  775  837  899  961 1023 1085 1147 1209 1271 1333 1395
   33  759  825  891  957 1023 1089 1155 1221 1287 1353 1419 1485
   35  805  875  945 1015 1085 1155 1225 1295 1365 1435 1505 1575
   37  851  925  999 1073 1147 1221 1295 1369 1443 1517 1591 1665
   39  897  975 1053 1131 1209 1287 1365 1443 1521 1599 1677 1755
   41  943 1025 1107 1189 1271 1353 1435 1517 1599 1681 1763 1845
   43  989 1075 1161 1247 1333 1419 1505 1591 1677 1763 1849 1935
   45 1035 1125 1215 1305 1395 1485 1575 1665 1755 1845 1935 2025
```

Ok
<table>
<thead>
<tr>
<th>Function</th>
<th>Points</th>
<th>Description</th>
</tr>
</thead>
</table>
| letterGrade     | 10 pts | - Function name, parameters correct                                         
|                 |        | - Correct use of conditionals (if…elif…else)                               |
|                 |        | - Letter grade calculated correctly                                          |
|                 |        | - Return the string exactly as specified                                    |
| countLetter     | 10 pts | - Function name, parameters correct                                         |
|                 |        | - Function examines each letter in the string                               |
|                 |        | - Returns an integer                                                       |
|                 |        | - Correctly counts the number of times the letter appears in the string     |
| eyeForI         | 10 pts | - Function name, parameters correct                                         |
|                 |        | - Proper use of a for-loop or while-loop                                    |
|                 |        | - Correctly replaces all instances of “I” and “i” with “eye”                |
|                 |        | - Returns a string                                                         |
| wordMirror      | 10 pts | - Function name, parameters correct                                         |
|                 |        | - String is correctly mirrored                                             |
|                 |        | - Returns a string                                                         |
| encryption      | 10 pts | - Function name, parameters correct                                         |
|                 |        | - Correctly replaces each letter with the corresponding conversion          |
|                 |        | - Prints string, formatted exactly as instructed                             |
| guessPassword   | 10 pts | - Function header correct                                                  |
|                 |        | - Uses a while-loop, or recursion                                           |
|                 |        | - Prompts user until correct password is entered                            |
|                 |        | - Prints out correct statements when password is correct/incorrect          |
| countDown       | 10 pts | - Function name, parameters correct                                         |
|                 |        | - A loop is used print one number per line                                 |
|                 |        | - Numbers are decremented correctly by the 2nd parameter                   |
|                 |        | - “Blast Off!” is printed last                                              |
| numberBowTie    | 10 pts | - Function name, parameters correct                                         |
|                 |        | - Use of a loop                                                            |
|                 |        | - Correct spacing, length, and number of rows                              |
|                 |        | - Prints correct shape                                                     |
|                 |        | - Hardcoding all 8 possible printouts will result in 0 pts for this function|
| printTimes      | 20 pts | - Function name, parameters correct                                         |
|                 |        | - Correctly prints times table with correct start, end, and incrementing numbers |
|                 |        | - Table is nicely formatted                                                |
|                 |        | - Returns nothing                                                          |