Simple Functions
You will write a few python functions for practice with the language. In your HW2.py file, include a comment at the top with your name, section, GTId/Email, and your collaboration statement. Also, include each of the following functions below. For purpose of this homework, you may assume that all inputs will be valid.

1. countVowels(10pts)

Description:
Write a function that takes in one parameter: a string. Your function will analyze the string and print out the number of occurrences of each vowel respectively (a, e, i, o, and u). Note with caution that your function should be able to recognize both uppercase and lowercase vowels! The format of the information to be printed can be found under “Test
Cases”.

**Parameters:**
aString (String): A string

**Return Value:**
None

**Test Cases:**
1. countVowels(“so it goes”) prints “a: 0, e: 1, i: 1, o: 2, u: 0”
2. countVowels(“Are you enjoying this so far?”) prints “a: 2, e: 2, i: 2, o: 3, u: 1”
3. countVowels(“CS is sOoOoOo fun!”) prints “a: 0, e: 0, i: 1, o: 6, u: 1”

2. **finalGrade (10pts)**

**Description:**
Write a function that takes in a list of exam grades and prints the final grade (using 90-100 for A, 80-89 for B, etc.) after performing a grade replacement policy. The grade replacement policy takes the lowest grade from the list and replaces it with the second lowest grade in the list. There will be at least 2 numbers in the list, but there is no upper bound for the length of the list.

**Parameters:**
gradeList (List): A list of exam grades as integers

**Return:**
None

**Test Cases:**
1. finalGrade([100,90,80,70]) prints “B”
2. finalGrade([100,100,100,100]) prints “A”
3. finalGrade([90, 80, 80, 90, 85]) prints “B”
4. finalGrade([30, 80, 44, 90, 85]) prints “D”

3. **palindrome (15 points)**
Description:
Write a function that checks to see if the parameter is a palindrome. In the string you should ignore spaces and the cases of the letter. The function should return a Boolean.

Parameter:
aStr (String)

Return:
Boolean

Test Cases:
palindrome('Race car') → True
palindrome('taco Cat') → True
palindrome('No x in Nixon') → True
palindrome('This is a palindrome') → False
palindrome('Jake is the best TA') → False

4. pumpkinEye (15 points)

Description:
This function will take in a parameter called stars. This function should print a hollow triangle composed of asterisks. You can assume only valid integer parameters will be used. Use a loop and DO NOT HARDCODE!

Parameter:
stars (Integer): the number of asterisks per side (i.e. the height)

Return:
None

Test Cases:
pumpkinEye(4) → *   *   *   *
           * * * *
           *   *   * 
pumpkinEye(6) → * *   * *   * *   * *   * *
5. **nextRow (10pts)**

**Description:**
This is the beginning of pascals triangle:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

You can calculate any row of Pascal's triangle after the first two from the previous row. Create a new row that starts with a 1, filling in the inner values such that each number is the sum of the two values to the upper left and upper right above the in the previous row, and then adding a 1 to the end.

For example, the 3rd row of Pascal's triangle is [1,3,3,1]. So the 4th row would be calculated as [1, 1+3, 3+3, 3+1,1] = [1,4,6,4,1].

Write a function called `nextRow` that takes in one parameter, a list of numbers representing a row in a Pascal's triangle (of at least 2 numbers) and return a list representing the next row.

**Parameters:**
aRow (List): A row of integer numbers.

**Return Values:**
(List): A list of numbers representing the NEXT row of the triangle.

**Examples:**
nextRow([1,1]) returns [1,2,1]
nextRow([1,2,1]) returns [1,3,3,1]

6. **frequencyCount (15pts)**
Description: This function will take in some compound data type and count the amount of times each element occurs. It returns a dictionary with the elements as keys and their frequencies as values.

Parameters:
array: Some string, list, or tuple

Return Value:
aDict: Dictionary

Test Cases:
frequencyCount([1,1,2,3,1,2,4,3]) returns: {1: 3, 2: 2, 3: 2, 4: 1}
frequencyCount("Hello World!") returns :
{'e': 1, '!': 1, 'l': 3, 'd': 1, 'r': 1, 'o': 2, ' ': 1, 'W': 1, 'H': 1}
frequencyCount(("Hello",2,4,3.0,4,1,2,3.0,1,1)) returns:
{1: 3, 2: 2, 3.0: 2, 4: 2, 'Hello': 1}

7. noahsArk (15 points)

Description:
This function will take a dictionary as a parameter and alphabetize the keys and the values associated with the keys. It will then print out the dictionary. Refer to the test cases for the proper format (the values are led by tab characters).

Hint: Dictionaries are not sequences so they do not retain order. You will have to sort the keys separately (aDict.keys()).

Parameters:
aDict: The dictionary to be sorted. Assume the keys are sortable and the values are sortable lists.

Test Case:

```python
>>> dict1 = {'Animals': ['Anteaters', 'Bulls', 'Mountain Lion', 'Camel', 'Polar Bear', 'Kangaroo'], 'Amphibians': ['Frog', 'Toads', 'Salamanders'], 'Insects': ['Butterflies', 'Ants', 'Beetles', 'Bee', 'Centipede']}
```
>>> noahsArk(dict1)
Amphibians
   Frog
   Salamanders
   Toads
Animals
   Anteaters
   Bulls
   Camel
   Kangaroo
   Mountain Lion
   Polar Bear
Insects
   Ants
   Bee
   Beetles
   Butterflies
   Centipede
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Points</th>
</tr>
</thead>
</table>
| **countVowels (10pts)** | - Loops through input string 2pts  
- Identifies lowercase vowels 2pts  
- Identifies uppercase vowels 2pts  
- Keeps track of the number of occurrences of each vowel 2pts  
- Prints correct information in the correct format 2pts |        |
| **finalGrade(10pts)**  | - Correctly identifies the lowest grade 2pts  
- Correctly identifies the second lowest grade 2pts  
- Replaces lowest grade with second lowest grade successfully 4pts  
- Prints correct grade 2pts |        |
| **palindrome(15pts)** | - Correctly ignores spaces and cases 5pts  
- Correctly identifies palindrome 5pts  
- Returns Boolean 5pts |        |
| **pumpkinEye(15pts)** | - Correct number of *’s per side 5pts  
- Correct format of pumpkin eye 10pts  
- Hard coded printouts -15pts |        |
| **nextRow(10pts)**    | - Returns correct nextRow for all possible input rows 10pts |        |
| **frequencyCount(15pts)** | - Creates a key for all elements 5pts  
- Contains no keys equal to zero 2pts  
- Works for strings, lists, and tuples 3pts  
- Correct frequency counts 5pts |        |
| **noahsArk (15 pts.)** | - Alphabetizes keys 4pts  
- Alphabetizes values 4pts  
- Iterates through values 4pts  
- Print-out is correctly formatted 3pts |        |