CS 1301 – Homework 4

Due: Monday February 22nd, before 11:55pm

This is a paired programming assignment!

You are expected to work with the person you have been paired with in class, and you are **both** responsible for submitting the exact same code to T-Square. Follow good pair-programming practices by working together at a single computer and switching the driver/navigator role frequently.

Your pair may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. You should not exchange code or write code for other pairs. Collaboration at a reasonable level will not result in substantially similar code.

For pair programming assignments, you and your partner should turn in identical assignments.

Files to submit: hw4.py

For help:

-TA Helpdesk – Schedule posted on class website
-Email TAs
-Piazza

Notes:

- **Don't forget to include the required comments and collaboration statement (as outlined on the course syllabus)**
- **Do not wait until the last minute** to do this assignment in case you run into problems.

If you find a significant error in the homework assignment, please let a TA know immediately.

Part I – List Manipulation

For this part your group will be creating 5 functions.

(5pts) Define a function called **youAndThe6th** that takes in a single parameter: a list of integers. The function should return a new list containing all of the numbers that are evenly divisible by six. If there are no numbers divisible by six, return an empty list.

Example:

youAndThe6th([1,2,3,4]) should return []
youAndThe6th([12,37,42,66,72]) should return [12,42,66,72]

(10pts) Define a function called **union** that takes in two lists that contain numbers and returns the union of the two lists. **The function should not change either of the input lists.** The elements in the returned list must be in increasing order, and there must be no repeats in the returned list. Note: Numbers include both floats and integers, although our example below only shows integers.
(Hint: The aList.sort() function may help)

Example:

```
>>> a = [1,3,2]
>>> b = [4,2,5]
>>> c = union(a,b)
>>> print(c)
[1,2,3,4,5]
```

(10pts) Define a function called **sentenceGenerator** that takes in a list. The function should take all of the string elements from the list and *print* out a sentence. The sentence will consist of every single string with a **single space** in between each string. If there are no string elements in the list *print* out *“No strings from inputted list.”*
Note: Your sentence isn’t going to make sense in most cases, just follow the instructions.

Example:

```
>>> sentenceGenerator([“Mulan”, “and”, 5, “Aladdin”, “is”, True, “Quality”])
Mulan and Aladdin is Quality
>>> sentanceGenerator([6, 7, False, None, 9])
No strings from inputted list.
```

(15pts) Define a function called **powerLevel** that takes in a list of 3 integers. Goku is training to raise his power level so he can defeat Prince Vegeta. Goku's current power level is **3000**. However, he must raise it to at least **9000.** The input is 3 integers. The first number is number of reps of his workout, the second number is the number of sets, and the third is the weight(lbs) of his bench-press. Multiplying these three numbers should yield his **additional power level**, which is to be added to the original value of 3000. If his power level is below 9000, print out a statement describing how much more Goku should increase his power level by. If his power level is at or over 9000 print the statement *"It's over 9000!"*

Examples:

```
>>> powerLevel( [3, 8, 100] )
Goku needs to increase his power-level by 3600!
>>> powerLevel(  [ 4, 6, 300]  )
It's over 9000!
```

(15pts) Define a function called **multiplyNums** that takes in a list and returns the product of all of the numbers from that list. The list may contain elements which are neither integers nor floats, and any non-numeric data should be ignored. If there is a nested list within a list, multiply all of the numbers in the nested list by your overall product as well (If the nested list itself has a nested list, do the same!). Your returned value can be either a float or an integer depending upon if the input list had floats. If the input list has no numbers, your function should return None.

You must use recursion for this function.

Example:

```
>>> a = ['Gon', 1, 'Killua', 7, [1,2, [2.0,'HxH'], 3]]
>>> b = multiplyNums(a)
>>> print(b)
84.0
```

Part II – String Manipulation

For this part, your group will be creating 4 functions.

(5pts) Write a function called **backwards** that takes in one string as a parameter. The function return a new string. The returned string should just be the original string, but in reverse order.

Examples:

```
>>> a = backwards(“School”)
>>> print(a)
loohcS
>>> b = backwards(“cat”)
>>> print(b)
tac
```

(10pts) Write a function called **abbreviator** that takes in one string as a parameter. The function should return a new string. The returned string should contain only the capital letters and numbers from the original string. Make sure that your returned string does not contain any special characters such as periods, semicolons, apostrophe’s, spaces, etc.
(Hint: the isalpha() and isupper() functions may help)

Examples:

```
>>> a = “If You’re Reading This It’s Too Late”
>>> b = abbreviator(a)
>>> print(b)
IYRTITL
>>> c = “Hotline Bling part 2”
>>> d = abbreviator(c)
>>> print(d)
HB2
```

(15pts) Write a function called **evenOdd** that takes in a list of random elements. This function should return a new string. For every string located at an even position within the list (count the 0^{th} index as even), you will take the **first vowel (a,e,i,o,u)** out of the string and add it to the new string. For every string located at an odd position in the list, you will take **first consonant** out of the string and add it to the string. **Treat y as a consonant.** The returned string must be all lowercase. Do not process non-strings elements and ignore characters in strings that are not letters.

Examples:

```
>>> a = evenOdd([‘cat’, ’Dog’, False,  9, ’treat’])
>>> print(a)
ade
>>> b = evenOdd([‘Perfume’, ‘Kyary’, ‘pon-pon’, True, None])
>>>print(b)
eko
```

(15pts) Write a function called **parse** that takes in two parameters: a string and a delimiter. Your function should return a list of separated strings depending on the delimiter. You may assume the delimiter will be a single character. Each string in the list should not contain the delimiter itself. **You may not use any built-in Python functions such as the split function, instead, use iteration and conditionals.**

Examples:

parse("This is a Sentence", " ") should return ['This', 'is', 'a', 'Sentence']
parse("f-r-i-e-n-d-s", "-") should return ['f', 'r', 'i', 'e', 'n', ‘d’, ‘s’]
parse("f,.5,.5",",") should return ['f', '.5', '.5']
parse(“f,,.5”,”,”) should return ['f', '', '.5'] ***(The 2nd element is an empty string).***

Rubric

List Manipulation Functions

youAndThe6th	**5**
Returns an empty list when input has zero multiples of six	2
Returns a list with numbers of only multiples of six	3
union	**10**
The original two lists remained the same	3
Numbers are in ascending order in the returned list	3
No repeated numbers in the returned list	4
sentenceGenerator	**10**
Prints out any statement	3
Prints out the correct statement for all cases	7
powerLevel	**15**
Calculates the additional power level correctly	3
Accounts for current power level	5
Prints out the correct statement for all cases	7
multiplyNums	**15**
Returns a float or integer	3
Returns the correct value	5
Uses recursion to solve the problem	7

String Manipulation Functions

backwards		**5**
Returns any string	2	
Returns the correct string	3	
abbreviator		**10**
Returns a string with uppercase alpha characters	2	
String contains numbers	4	
String contains zero special characters	4	
evenOdd		**15**
Returns any string	3	
Returns a string with lowercase alpha characters	4	
Element in the string is correct according to its index	4	
Elements in the string are in the correct order	4	
parse		**15**
Returns a list	3	
Returns a list of separate strings	5	
Separates the strings correctly by the delimiter	7	