CS 1301 – Homework 4

List and String Manipulation
Due: Monday September 26th, 11:55pm
File to submit to T-Square: HW4.py

You should work individually on this assignment. You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. Students may only collaborate with fellow students currently taking CS 1301, the TA's and the lecturer. You should not exchange code or write code for others. For individual assignments, each student must turn in a unique program. Your submission must not be substantially similar to another student's submission. **Collaboration at a reasonable level will not result in substantially similar code.**

For Help:

- TA Helpdesk – Schedule posted on class website.
- Email TA's or use Piazza Forums

Notes:

- **Do not forget to include the required collaboration statement** (outlined on the syllabus).
- **Do not wait until last minute** to do this assignment in case you run into problems.
- **Read the entire specifications document before starting this assignment.**

Part 1 – List Manipulation

Function Name: `multiplyNums` (10 points)

Description: Define a function that multiplies each item in a list of numbers (either floats or ints) with the next element. If an item is the last element in the list, the item should be multiplied with itself. The function should return the modified list.

Parameter(s):

1. A list of floats and/or integers. The list may be empty.

Return Value: The modified list.

Example(s):

```
>>> test_1 = multiplyNums([1, 2, 3])
>>> print(test_1)
[2, 6, 9]
>>> test_2 = multiplyNums([9])
>>> print(test_2)
[81]
>>>test_3 = multiplyNums([100, 38.5, 12])
>>>print(test_3)
[3850.0, 462.0, 144]
```

Function Name: `multiplyNums2` (10 points)

Description: Define a function that performs exactly like `multiplyNums`, **but leaves the original list unmodified.** You may use a call to `multiplyNums` in your implementation.

Parameter(s):

1. A list of floats and/or integers. The list may be empty.

Return Value: A new list with the updated values.

Example(s):

```
>>> a = [1, 2, 3]
>>> b = multiplyNums2(a)
>>> print(a)
[1, 2, 3]
>>> print(b)
[2, 6, 9]
```

Function Name: `pairExists` (15 points)

Description: Define a function that accepts one list of integers and an additional number as parameters. The function should return `True` if there exists two elements in the list that, if added together, are equal to the value of the second parameter.

Parameter(s):
1. A list of integers.
2. A single integer.

Return Value: A boolean representing the result.

Example(s):
```
>>> test_1 = pairExists([5, 10, 15, 20], 30)
>>> print(test_1)
True
>>> test_2 = pairExists([5, 10, 15, 20], 31)
>>> print(test_2)
False
```

Function Name: `listSymmetric` (15 points)

Description: Define a function that accepts one list of integers as a parameter and returns a boolean representing whether the list is symmetric. For the purpose of this problem, a list is considered symmetric if and only if, there is an item of equal value on the opposite side of the list and equidistant from the nearest end (similar in concept to a palindrome).

Parameter(s):
1. A list of integers.

Return Value: A boolean representing whether the list is symmetric.

Example(s):
```
>>> test_1 = listSymmetric([1, 2, 3, 3, 2, 1])
>>> print(test_1)
True
>>> test_2 = listSymmetric([1, 2, 3, 2, 1])
>>> print(test_2)
True
>>> test_3 = listSymmetric([1, 2, 3, 4])
>>> print(test_3)
False
```

Part 2 – String Manipulation

Function Name: `replaceStr` (10 points)

Description: Define a function that accepts three strings as parameters. The function should replace any instance of the second string in the first string with the third string. The resulting string should be returned. Remember that strings are immutable, so you will need to construct a new string for each replacement.

(Hint: you may find the `index` function useful)

Parameter(s):

1. A string to alter.
2. A string representing the source to search for in the first string.
3. A string representing the value that the second string should be replaced with.

Return Value: The new altered string.

Example(s):

```
>>> test_1 = replaceStr("It was all yellow", "yellow", "red")
>>> print(test_1)
It was all red
```

Function Name: `polyAlphaCipher` (25 points)

Description: Define a function that encrypts a string using a polyalphabetic cipher. Recall that for a single character in python, you can determine the ASCII character code by using the `ord()` function. By changing the ASCII character code and then using `chr()` to restore the character to a string, you can change the character itself. Use this concept to increment the ASCII character code of each character in the given string by an offset that starts at the value of 1. Each time the offset is used, increment it using the increment number given as a parameter. Use the modulus operator (%) to make sure the ASCII value is never greater than 255. As you change each character, build up a new string. Return this string as the encrypted result.

Parameter(s):

1. A string to cipher.
2. An integer to represent the offset increment.

Return Value: The ciphered string.

Example(s):

```
>>> increment = 10
>>> test_1 = polyAlphaCipher("Welcome to New York", increment)
>>> print(test_1)
Xp ¢gÅÊ½Þúð!              *(see note)
>>> increment = 2
>>> test_2 = polyAlphaCipher("Welcome to New York", increment)
>>> print(test_2)
Xhqjxxr/5e~=x             *(see note)
```

*(note: your output may be slightly different depending on your operating system. It should be very similar, however.)

Function Name: `polyAlphaDecipher` (15 points)

Description: Define a function that accepts a ciphered string (presumably the output of `polyAlphaCipher`) and the original increment as parameters. The function should reverse the effects of `polyAlphaCipher` to return the original string.

Parameter(s):

1. A string to decipher.
2. An integer to represent the offset increment.

Return Value: The deciphered string.

Example(s):

```
>>> increment = 10
>>> test_1 = polyAlphaCipher("Welcome to New York", increment)
>>> print(test_1)
Xp ¢gÅÊ½Þúð!                *(see note)
>>> test_2 = polyAlphaDecipher(test_1, increment)
>>> print(test_2)
Welcome to New York
>>> increment = 5
>>> test_3 = polyAlphaDecipher(test_1, increment)
>>> print(test_3)
WjvrCR¡¸                    *(see note)
```

*(note: your output may be slightly different depending on your operating system. It should be very similar, however.)

Rubric

`multiplyNums`	**10**
correct function header	3
returns result of correct type	2
result is correct	5
`multiplyNums2`	**10**
correct function header	3
returns result of correct type	2
result is correct	5
`pairExists`	**15**
correct function header	3
returns result of correct type	2
result is correct	10
`listSymmetric`	**15**
correct function header	3
returns result of correct type	2
result is correct	10
`replaceStr`	**10**
correct function header	3
returns result of correct type	2
result is correct	5
`polyAlphaCipher`	**25**
correct function header	3
returns result of correct type	2
result is correct	20
`polyAlphaDecipher`	**15**
correct function header	3
returns result of correct type	2
result is correct	10