15-110 Refresher Session : Week 10

No Calculators, only Brains !!

1. **Match the following** Let the string `s = "I will work smarter next time."` . Match the outputs of the following code.

`print(s[1::3])`	`I`
`print(s[::-1])`	`1`
`print(len(s))`	`Ii rsrre m`
`print(s.count('r'))`	`.emit txen retrams krow lliw I`
`print(s.count('a'))`	`3`
`print(len(s[1:3]))`	`31`
`print(s[2:0:-1])`	`2`

2. **Act like a Computer**

```
def mystery(S , y , x = 5) :
    newString = ""
    for i in range (x, y) :
        newString = newString + S[i]
    return newString
```

What would the output be for the following function calls ?

(a) `mystery('Europe', 3 , 0) + mystery('seeks', 4 , 2) + mystery('programmers , 6', 6))`

(b) ***Bonus:*** `mystery('Root', 6 , 5)`

3. **Act like a programmer**
Anagrams are words that have the same letters with the same number of occurrences. For example, `was` and `saw` are anagrams. Your task is to define a function `isAnagram(first, second)` which takes two words `first` and `second` and returns `True` if the words are anagrams and `False` otherwise. Your function should be case-insensitive. For example

```
isAnagram("Was" , "saw") returns True
```

4. **Act like a Computer**

```
def mystery(s):
    result = ""
    for i in s.split(","):
        k = "$".join(i.split("e"))
        result = result + k
    return result
```

What does `mystery('Europe, seeks, programmers in, python and, artificial intelligence')` return ?

5. **Act like a Programmer**
Strings can be rotated. HOW ??? A string is said to be rotated once when the last character is placed in front of the string. For example, the word "think" rotated once would be "kthin", the word "think" rotated twice would be "nkthi". Implement a function `rotateStrings(L)` which would take a list of strings and rotate each of the strings with respect to the index. The string at index 0, would be rotated 0 times; the string at index 1 would be rotated 1 time, and so on. Return the rotated strings as a tuple. For example

```
rotateStrings (["I" , "understood", "the", "question", "well"]) returns
                      ("I" , "dunderstoo", "eht" , "ionquest", "well"))
```

Hint : You can use helper functions to rotate the string for you

6. **Act like a Programmer**
Sam is developing his vocabulary. As a part of his learning, he picks up random words, their meanings and tries to use it in a sentence. Today, he is looking at a few words and found that some words are short and some words are long. So he decided to play the game of substrings - a substrings is a string that is a part of another string. The game is played in the following way :

(a) He sets the longest word of the day as the masterword.
(b) He picks a word
(c) He checks if the word is a substring of the masterword. For example, `word` is a substring of `masterword`.
(d) If it is a substring, he writes down the word and position of occurrences.
(e) If it is not a substring, he writes down the word and 0
(f) He repeats the same for every other word and writes them down in lexicographical (alphabetic) order

Your task is to define a function `substringGame(L)` which takes a list L of strings and returns a list of lists with each inner list containing the word and the positions of occurrence. You can assume that there will be only one masterword and all the words are in lower case. For example,

```
substringGame(["yes", "yer" ,"yesyes", "yesyesyesyessaeayesyes", "sey", "ye" ,
"prote", "lie", "bill"]) returns
returns [['bill', 0], ['lie', 0], ['prote', 0], ['sey', 0], ['ye', 0, 3, 6, 9, 16, 19],
['yer', 0], ['yes', 0, 3, 6, 9, 16, 19], ['yesyes', 0, 3, 6, 16]]
```

Hint : You can use helper functions to dissect the problem and solve each part