Question 1:

Create a Python FUNCTION that takes a list as a parameter. The FUNCTION should determine all the unique values in the list and count their occurrences. The output of the FUNCTION should print a row for each unique item using the print statement listed below for each row.

The signature of the FUNCTION should be:

```
def count_unique_values(input_list):
```

The output should have a header using the following statements:

```
print("{:<15}{}".format("Unique Item", "Count"))

print("-" * 25)
```

Format for each row in the table should be:

```
print("{:<15}{}".format(item, count))
```

In [ ]:
```
#function to count the unique values in a list
def count_unique_values(input_list):
    dict = {} #create an empty dictionary

    #loop through the list
    for item in input_list:
        # if the element is  already in the dictionary
        if item in dict:
            # increment the value of the key by 1
            dict[item] = dict[item] + 1
        #if the element is not in the dictionary
        else:
            # add the element to the dictionary and set the value to 1
            dict[item] = 1
    return dict
```

```python
#function to display the result
def display_result(dict):
    print("{:<15}{}".format("Unique Item", "Count"))
    print("-" * 25)
    # loop through the dictionary and print the key and value
    for key, value in dict.items():
        print("{:<15}{}".format(key, value))

dict = count_unique_values([1,2,3,4,5,2,3,4,5,3,4,5,4,5,5]) #call the function
display_result(dict)#call the function

print("-*-" * 12)

dict = count_unique_values(["a",5,2,3,4,5,3,4,5,4,5,5,"a", "z", "$$", "$$$"])#call the  count_unique_values functio
display_result(dict) #call the display_result function
```

```
Unique Item    Count
-------------------------
1              1
2              2
3              3
4              4
5              5
-*--*--*--*--*--*--*--*--*--*--*--*-
Unique Item    Count
-------------------------
a              2
5              5
2              1
3              2
4              3
z              1
$$             1
$$$            1
```

Question 2:

A Caesar cipher is a simple substitution cipher based on the idea of shifting each letter of a word a fixed number (called the "key") of positions in the alphabet. For example, if the "key" value is 2, the word "Sourpuss" would be encoded as "Uqwtrwuu." The original message can be recovered by "reencoding" it using the negative of the "key".

Write a FUNCTION that can encode and decode Caesar ciphers. The input to the FUNCTION will be an integer value of the "key" and a string of plaintext. The output will be an encoded word where each character in the original word is replaced by shifting it "key" characters in the Unicode character set. For example, if ch is a character in the string and "key" is the amount to shift, then the character that replaces ch can be calculated as:

```
chr (ord (ch) + key)
```

The FUNCTION will be called with two arguments, the first will be an integer (either positive or negative) and will be the value of the "key". The second argument will be the string which is to be encoded (if the "key" is a positive value) or decoded (if the "key" is a negative value). The output of the FUNCTION should be the encoded or decoded string.

The FUNCTION signature should be:

```
def main(key, plain)
```

In [ ]:
```python
#function to modify the plain text by adding the key to the ASCII value of each character
def main(key,plain):
    encodedPlain = "" #create an empty string
    for character in plain: #loop through the plain text
        encodedPlain = encodedPlain  +  chr(ord(character) + key) #add the key to the ASCII value of each character
    return encodedPlain #return the encoded plain text

print("{:<15}{}".format("Test", "Result"))
print("-" * 25)
print("{:<15}{}".format("abc",main(1, "abc"))) #call the main function
print("{:<15}{}".format("bcd",main(-1,"bcd"))) #call the main function
print("{:<15}{}".format("cdeAAk",main(5,"cdeAAk"))) #call the main function
```

```
Test           Result
-------------------------
abc            bcd
bcd            abc
cdeAAk         hijFFp
```

Question 3:

One problem with the previous problem is that it does not deal with the case when we "drop off the end" of the alphabet. A true Caesar cipher does the shifting in a circular fashion where the next character after "Z" is "a." and the next character after "z" is "A". Modify the FUNCTION to the previous problem to make it circular. As in the previous function your function should expect two arguments from the call, the first is an integer which is the value for the key and the second is a string which is the message that will be encoded. The function should return just the encoded string. Hint : Make a string containing all the uppercase and lowercase characters of the alphabet and use positions in this string in your code.

```
In [ ]: #create a string of all the uppercase and  lowercase letters
        alphabet = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"

        #function to modify the plain text by adding the key to the ASCII value of each character
        def main(key, plain):
            encodedPlain = "" #create an empty string
            for character in plain:#loop through the plain text
                if character in alphabet:
                    encodedPlain +=  alphabet[alphabet.index(character) + key] #add the key to the ASCII value of each char
            return encodedPlain #return the encodedPlain plain text

        print("{:<15}{}".format("Test", "Result"))
        print("-" * 25)
        print("{:<15}{}".format("abc",main(1, "abc"))) #call the main function
        print("{:<15}{}".format("bcd",main(-3, "abc"))) #call the main function
        print("{:<15}{}".format("cdeAAk",main(25,"xyzAAk"))) #call the main function
```

```
Test           Result
-------------------------
abc            bcd
bcd            XYZ
cdeAAk         WXYZZJ
```