Function Practice Problems

EASY

1. Write a function in Python that will take a string text as input from the user and returns the list of unique characters concatenated with their ASCII value at the front and back side.

===

Sample Input:

"pythonbook"

Function Calling:

```
function_name("pythonbook")
```

Sample Output:

```
['112p112', '121y121', '116t116', '104h104', '111o111', '110n110', '98b98', '107k107']
```

2. Write a function in Python that will take a string text as input from the user and returns a dictionary having the unique characters as the keys and the list of their both-way indexes (positive and negative index) as the values.

===

Sample Input:

"pythonbook"

Function Calling:

```
function_name("pythonbook")
```

Sample Output:

```
{'p': [0, -10], 'y': [1, -9], 't': [2, -8], 'h': [3, -7], 'o': [4, -6, 7, -3, 8, -2], 'n': [5, -5], 'b': [6, -4], 'k':
[9, -1]}
```

MEDIUM

3. Write a function in Python that will take a space separated string text as input from the user and returns a dictionary having the unique words as the keys and their frequency in the given text as the values in a sorted order(ascending) according to the frequencies.

===

Hints (1):
Frequencies of the words can be easily counted with built-in function count().

Hints (2):
Sorting according to frequencies can be done in two approaches.

- **Approach (1):** Use the built-in sorted() function. Here, for the key, you need to call a custom-made function that will return frequencies for each word. So, the sorted() function will return values sorted according to the frequencies.
- **Approach (2):** Make two lists. One with the keys and other one with the frequencies. For a particular keys, values should be found in the same index of the frequencies list. Then similar to the Assignment07 Task-5. Sort the values of frequencies and in time of swapping, swap BOTH keys & frequencies. You can use any one of the sorting algorithms taught (Bubble sort or Selection sort)

===

Sample Input1:
"go there come and go here and there go care"

Function Call1:
function_name("go there come and go here and there go care")

Sample Output1:
{'come': 1, 'here': 1, 'care': 1, 'there': 2, 'and': 2, 'go': 3}

===

4. Write a function in Python that will take a number string text as input from the user and returns a dictionary having the unique numbers as the keys and the tuple of being the number to be even, odd, prime and perfect as thevalues.

===

Hints (1): Write a function to check whether a number is Perfect or not and RETURN “Perfect” and “Not Perfect” accordingly.

Hints (2): Write a function to check whether a number is Prime or not and RETURN “Prime” and “Not Prime” accordingly.

Hints (3): Write a function to check whether a number is Even or not and RETURN “Even” and “Odd” accordingly.

Hints (4): Call 3 above mentioned functions and store their returned values in a list/tuple.

```
even= even_check()
prime= prime_check()
perfect= perfect_check()
tup_for_digit = (even, prime, perfect)
```

===

Sample Input1:

"2441396"

Function Call1:

function_name("2441396")

Sample Output1:

```
{2: ('even', 'prime', 'not perfect'), 4: ('even', 'not prime', 'not perfect'), 1: ('odd', 'not
prime', 'not perfect'), 3: ('odd', 'prime', 'not perfect'), 9: ('odd', 'not prime', 'not perfect'), 6: ('even', 'not
prime', 'perfect')}
```

===

5. Assume, you have been given two matrixes in list format. Write a function in Python that will calculate the summation of these two matrices. Then RETURN the summation matrix and print it in the function call. [A matrix can only be added to another matrix if the two matrices have the same dimension] **[Avoid using built-in Functions]**

==

Given Matrix1:

matrix_A = [[1,5], [-4,3]]
matrix_B = [[2,-1] , [4,-1]]

Function Call1:

function_name(matrix_A , matrix_B)

Sample Output1:

matrix_sum = [[3,4] , [0,2]]

Explanation1:

Inside matrix_A and matrix_B, each list is a row matrix.
For example, In matrix_A, Row 1 ----> [1, 5]

Row 2----> [4, 3]

In matrix_B, Row 1 ----> [2, -1]

Row 2----> [4, -1]

So, in output, matrix_sum = [[1+2 , 5 -1]
[-4+ 4, 4 -1]]

==

Given Matrix2:

matrix_A = [[1,5, 4] , [-4,3, 3]]
matrix_B = [[2,-1, -3] , [4,-1, -4]]

Function Call2:

function_name(matrix_A , matrix_B)

Sample Output2:

matrix_sum = [[3, 4, 1], [0, 2, -1]]

==

Given Matrix3:

matrix_A = [[1,5, 4] , [-4,3, 3] ,[-4,0, 6]]
matrix_B = [[2,-1, -3] , [4,-1, -4], [2,6, 3]]

Function Call3:

function_name(matrix_A , matrix_B)

Sample Output3:

matrix_sum = [[3, 4, 1], [0, 2, -1], [-2, 6, 9]]

==

Given Matrix4:

matrix_A = [[1,5] , [-4,3], [9,2]]
matrix_B = [[[2,-1] , [4,-1]]]]

Function Call4:

function_name(matrix_A , matrix_B)

Sample Output4:

```
No of rows not same
```

==

Given Matrix5:

matrix_A = [[1,5, 4] , [-4,3] ,[-4,0, 6]]
matrix_B = [[2,-1, -3] , [4,-1, -4], [2,6]]

Function Call5:

function_name(matrix_A , matrix_B)

Sample Output5:

```
No of columns not same
```

==

HARD

6. Assume, you have been given a Matrix in a list format. Write a Python function called print_matrix_list() that will take a list as an argument and print the Matrix in its proper square form.

===

Given Matrix1:
matrix= [[1, 2, 3, 4], [4, 5, 6, 7], [7, 8, 9, 3], [9, 1, 2, 3]]

Function Call1:
print_matrix_list(matrix)

Sample Output1:

```
1 2 3 4
4 5 6 7
7 8 9 3
9 1 2 3
```

===

Given Matrix2:
matrix= [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

Function Call2:
print_matrix_list(matrix)

Sample Output2:

```
1 2 3
4 5 6
7 8 9
```

===

Given Matrix3:
matrix= [[1, 0, 0, 0], [0, 5, 0, 0], [0, 0, 9, 0], [0, 0, 0, 3]]

Function Call3:
print_matrix_list(matrix)

Sample Output3:

```
1 0 0 0
0 5 0 0
0 0 9 0
0 0 0 3
```

===

7. Assume, you have been given a **Square Matrix** in a dictionary format.
Square Matrix: A square matrix is a matrix with the same number of rows and columns.

Now, write function called **convert_to_list(),** which converts the dictionary into a list of lists and returns the list where each list represents a row matrix. Then, print the returned list in the function call. Finally, print the returned matrix in Square format using the **print_matrix_list()** function.

===

Given1:
square_matrix_dict = {1 : [1,2,3,4] , 2 : [4,5,6,7] , 3 : [7,8,9,3] , 4:[9,1,2,3] }

Function Call1:
convert_to_list(square_matrix_dict)

Sample Output1:
[[1, 2, 3, 4], [4, 5, 6, 7], [7, 8, 9, 3], [9, 1, 2, 3]]

===

Given2:
square_matrix_dict = {1 : [1,2,3] , 2 : [4,5,6] , 3 : [7,8,9] }

Function Call2:
convert_to_list(square_matrix_dict)

Sample Output2:
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]

===

8. Assume, you have been given a Square Matrix in a dictionary format. Now, write function called **convert_to_diagonal()** which converts the Square Matrix into a Diagonal Matrix. Then prints both the Square Matrix and Diagonal Matrix in list of list format and their original square format.

[For printing in square format MUST use the print_matrix_list() function.]

Square Matrix: A square matrix is a matrix with the same number of rows and columns.

Diagonal Matrix: Any given square matrix where all the elements are zero except for the elements that are present diagonally is called a diagonal matrix.

==

Hints:

Step1: Inside the convert_to_diagonal(), call convert_to_list() for converting the dictionary into a list of lists.

Step2: Store and print the returned Square- Non-diagonal matrix in a variable.

Step3: Call the print_matrix_list() with the returned Square- Non-diagonal matrix for printing it in the original square format.

Step4: Except for the cases, when row_number & column_no is equal, make all other elements 0. By doing this, you will get your Diagonal matrix.

Step5: Call the print_matrix_list() with the Diagonal matrix made in step4 matrix for printing it in the original square format.

==

Given1:

```
square_matrix_dict = {1 : [1,2,3,4] , 2 : [4,5,6,7] , 3 : [7,8,9,3] , 4:[9,1,2,3] }
```

Function Call1:

```
convert_to_diagonal(square_matrix_dict)
```

Sample Output1:

```
Non-diagonal matrix:
In list of list format: [[1, 2, 3, 4], [4, 5, 6, 7], [7, 8, 9, 3], [9, 1, 2, 3]]
in original square format:
1 2 3 4
4 5 6 7
7 8 9 3
9 1 2 3
================
Diagonal matrix:
In list of list format: [[1, 0, 0, 0], [0, 5, 0, 0], [0, 0, 9, 0], [0, 0, 0, 3]]
in original square format:
1 0 0 0
0 5 0 0
0 0 9 0
0 0 0 3
```

==

Given2:

```
square_matrix_dict = {1 : [1,2,3] , 2 : [4,5,6] , 3 : [7,8,9]  }
```

Function Call2:

```
convert_to_diagonal(square_matrix_dict)
```

Sample Output2:

```
Non-Diagonal matrix:
In list of list format: [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
in original square format:
1 2 3
4 5 6
7 8 9
================
Diagonal matrix:
In list of list format: [[1, 0, 0], [0, 5, 0], [0, 0, 9]]
in original square format:
1 0 0
0 5 0
0 0 9
```