ENGR 101 – Introduction to Programming

Study Questions - Week 11

Q1- Write a function, it takes a string input and finds which letter exists how many times in this given string. {Hint:"use dictionary"}

E.g. if the input is "kara murat kim"

Output : {"k":2, "a":3, "r":2, "m":2, "u":1, "i":1," ":2}

Q2- Write a new function which takes the Output dictionary(given above) as an input. And create a new dictionary which maps from frequencies to letters.

E.g. new_dict1 = {1: ['u', 't', 'i'], 2: [' ', 'r', 'k', 'm'], 3: ['a']}

Q3- You are a teacher and you made 2 exams to your class. The results of exams are given at below as a dicitonary. Please write a function, which finds and returns the average of the first exam grades.

```
exam_dict = {"john":{"exam1":40, "exam2":70}, "murat":{"exam1":20, "exam2":30},
"faruk":{"exam1":35, "exam2":55}, "sow":{"exam1":80, "exam2":90}}
```

Q4- Modify the function which you wrote for Q3, so it can find total average . (Total average= All grades belong to exam1 and exam2 /2x(student number).)

Q5- You have an exam result list called exam_list. ([student_name, (exam1_grade, exam2_grade)])

```
exam_list = ["john", (40, 70), "murat", (20, 30), "faruk", (35, 55), "sow", (80, 90)]
```

Write a function, which finds and returns the average of the first exam grades.

Q6- Modify the function which you wrote for Q5, so it can find total average .

Q7- Read the code below and tell the output.

```
def blabla(x):
    for i,j,k in x:
      print i,j,k

blabla([("batman", "beats", "superman"), ("Sixth", "of",
"November"), ("Since", "18", "years"), ("You are", "the", "best")])
```

Q8- Modify the function in Q7, to get the same output, but this time you are not allowed to use multiple variables for for loop. (not allowed : for i,j,k in x, allowed: for i in x)

Q9- Write a function which converts,

This:

{'lemon': 1, 'watermelon': 5, 'apple': 1, 'banana': 2}

To this:

[('watermelon', 5), ('lemon', 1), ('apple', 1), ('banana', 2)]

Q10- Convert back this [('watermelon', 5), ('lemon', 1), ('apple', 1), ('banana', 2)]

To this

{'lemon': 1, 'watermelon': 5, 'apple': 1, 'banana': 2}

Q11- In this exercise, you are going to compress the strings that have long sequences of equal characters. Your program should first prompt the user to enter a string str, and return compresed form of str. For example, if the input is "**aaaabbaaabbbbbbcccd**",the output should be 4a2b3a5b3c1d. (Warning: For counting the occurrence of each letter, you must use a dictionary).

Q12- Write a function that takes two dictionaries (d1 and d2, where keys and values are integers) and returns a new dictionary which contains the union of items in d1 and d2. Union of two dictionaries d1 and d2 results in a new dictionary d3 such that:

- If a key is included only in d1, then this key and corresponding value from d1 should be included in d3.
- If a key is included only in d2, then this key and corresponding value from d2 should be included in d3.
- If a key is included in both d1 and d2, then d3 should include this key and larger of the corresponding values from d1 and d2.

Example: If d1={1:2,3:4,5:6} and d2={1:3,6:5,3:8} your output should be something like {1:3,5:6,6:5,3:8}. The order of elements in the output dictionary is not important.

Q13- Write a function, generalize(), that takes a category (which may be one of "airport", "city", or "status") and a dictionary (where the key is a tuple containing airport name, city name, and flight type (either "domestic" or "international") and the value is an integer representing the number of flights per month. The function should return a new dictionary where keys are categories and values are number of total flights.

Example: If the input is sample_dictionary (defined below) and "city", the output should be something like {"Istanbul":135000, Izmir:20000}.

sample_dictionary={(“Ataturk”, “Istanbul”,”International”):30000, (“Ataturk”, “Istanbul”,”Domestic”):65000, “SabihaGokcen”, “Istanbul”,”International”):10000, “SabihaGokcen”, “Istanbul”,”Domestic”):30000, (“Menderes”, “İzmir”,”International”):5000, (“Menderes”,“İzmir”,”Domestic”):15000}

Q14- Create a function named **charMap** that accepts a string as a parameter. Your function should create a dictionary that has each unique character in the string as a key. The value associated with the key will be a tuple that contains all of the (integer) positions (indices) in the string where that character occurs. Your function should return this dictionary after creating it. Example output:

```
>>> result = charMap("Hello World")

>>> print result

{' ': (5,), 'e': (1,), 'd': (10,), 'H': (0,), 'l': (2, 3, 9), 'o': (4, 7),
'r': (8,), 'W': (6,)}
```

Q15- Write a function called getTopelements() that takes a string as an input and returns the most common 3 letters in that string. If the characters have same frequency, it should return in alfabetical order.

For example:

```
>>>getTopelement(“programming”)

m
a
g
```

Q16- Write a Python program that reads in lines of input from the user until you enter same input more than three times.

Q17- Write a program that first ask you enter a student name and then his/her grade. Your program should assign student to the grade and all students with the same grade will be assigned to same grade.

• For example:

```
Enter a student: Ayse

Enter grade:50

Enter a student: Kamuran

Enter a grade: 70

Enter a student: Raziye

Enter grade: 50
```

• And output should be {50:[Ayse, Raziye], 70:[Kamuran]}

Q18- What will be the output of the following code?

```
arr={}
arr[1]=1
arr['1']=2
arr[1] += 1
sum = 0
for k in arr:
    sum += arr[k]
 print (sum
```

Q19- Implement a function make_simple that takes a tüple which may be nested, and returns a new tuple that has no nested tuples.

For example:

```
>>> x = (1, (2, (3,), 4), 5)
>>> make_simple(x)
(1, 2, 3, 4, 5)
>>> y = (1, 2, 3, 4)
>>> make_simple(y)
(1, 2, 3, 4)
```

Q20- Implement a function reversed_tuple() that takes a tuple and reverses it. If the tuple has elements that are themselves tuples, those elements will be reversed too.

For example:

```
>>> tup = (1, (2, (3,), 4), 5)
>>> reversed_tuple(tup)
(5, (4, (3,) 2), 1)
>>> y = (1, 2, 3, 4)
>>> reversed_tuple(y)
(4, 3, 2, 1)
```