Mock3 Test

This is a mock test for which you will not get a grade. Create the following programs. Save the programs in the files with the names shown in parentheses (**p1.py, p2.py, p3.py, …**). The tasks are a bit more complex so you can prepare yourself better for Test3. During Test3, the tasks will be arranged so that they can be completed within 80 minutes.

**Read this first!!**

Please remember to test every program you've created. To test your program, you can create a main() method and put statements inside it to test a function or a class. See the following example:

# function summing up two numbers  
def f(n1,n2):  
 return n1+n2

def main():  
 # function testing  
 print(f(2,5)==7)  
 print(f(3,0)==3)

if \_\_name\_\_ == "\_\_main\_\_":  
 main()

The main() method will not be checked during program assessment. Only functions or classes you create are graded. Hence, you do not need to remove the main() method from your program. Just make sure the whole program does not contain any errors.

Try to be like a pro and test every program you created. The details on how to create a main() function can be found at: <https://realpython.com/python-main-function/>. **Please do not forget to use the debugger!!!**

(**p1.py**) Define a function f(n) that represents any integer n using sticks. The function returns a string containing as many sticks as indicated by the number. For efficient calculations, the sticks are grouped in groups of five and separated by a minus sign. Example:  
f(-4) 🡪 ""  
f(0) 🡪 ""  
f(5) 🡪 "/////"  
f(7) 🡪 "/////-//"  
f(10) 🡪 "/////-/////"  
f(11) 🡪 "/////-/////-/"

(**p2.py**) Define a function f(arr) that returns true if the given two-dimensional array is a valid multiplication table, or false otherwise. Example:  
f([[1,2,3],[2,4,6],[3,6,9]]) 🡪 True  
f([[1,2],[2,4]]) 🡪 True  
f([[1,2,3,4],[2,4,6,8],[3,6,9,12],[4,8,12,16]]) 🡪 True  
f([[1,2],[3,6]]) 🡪 False  
f([[1,2,3],[2,4,6]]) 🡪 False  
f([[1,2,3],[2,5,6]]) 🡪 False

(**p3.py**) The text t contains information about family members. Define a function f1(t) that extracts the names and ages of family members from the text and returns these data in the form of a dictionary, in alphabetical order. Define a function f2(d) that returns the total age of all family members based on the dictionary data. Example:  
f1("Mark is 24 and Ann is 27") 🡪 {"Ann":27, "Mark":24}  
f1("Peter is 11, Barbara is 7 and their grandfather John is 103!!") 🡪 {"Barbara":7, "John":103, "Peter":11}   
f2(f1("Mark is 24 and Ann is 27")) 🡪 51  
f2(f1("Peter is 11, Barbara is 7 and their grandfather John is 103!!")) 🡪 121

(**p4.py**) The computer system registers all entries into the car park ("in") and exits from the car park ("out"). Define the function f(d), which for the given data will return an array containing the registration numbers of the vehicles that remain in the car park, in alphabetical order. Example:  
cars = [["KR234","in"],["BA123","in"],["GX444","in"],["KR234","out"],  
["BA111","in"],["BA123","out"],["KR234","in"]]  
f(cars) 🡪 ["BA111","GX444","KR234"]

(**p5.py**) The object created from a class C contains an array of integers, passed in when the object is created. Define a text representation of an object that returns an expression that sums up all the numbers in an array, in the order as in the array. Example:  
C([5,12]) 🡪 "5+12=17"  
C([6,0,15]) 🡪 "6+0+15=21"

(**p6.py**) The counter allows you to count any elements. Create a class C to create counters. The initial value of the counter is assigned when the object is created. The class contains the following methods:  
m1() - returns the counter value  
m2() - increases the counter value by 1  
m3() - decreases the counter value by 1  
m4(n) - increases the counter value by n  
Example:  
c=C(5)  
c.m1() 🡪 5  
c.m2()  
c.m1() 🡪 6  
c.m4(-8)  
c.m1() 🡪 -2  
c.m3()  
c.m1() 🡪 -3  
c.m4(10)  
c.m1() 🡪 7

(**p7.py**) A class C contains the following static/class methods:  
m1(n) – returns a number based on n, with the odd digits removed  
m2(n) – returns true when each subsequent digit in a number n is equal to or greater than the previous digit or false otherwise  
m3() – returns a string, with the digits missing from the number n, in ascending order  
Example:  
C.m1(4231564) 🡪 4264  
C.m1(79381) 🡪 8  
C.m2(125579) 🡪 True  
C.m2(4557879) 🡪 False  
C.m3(23557) 🡪 "014689"  
C.m3(12340) 🡪 "56789"

(**p8.py**) A class C contains a dictionary with students and their grades. The dictionary is passed to the object when it is created. Define a method m(n) that returns a list of students whose average grade is not lower than n. Return the list of students as a string, names separated by commas, without spaces, in alphabetical order. Example:  
s = C({"Peter":[4,5,4]}, "Harry":[2,5], "Barbara":[3,3,3,5,5,5]})  
s.m(4) 🡪 "Barbara,Peter"  
s.m(3) 🡪 "Barbara,Harry,Peter"  
s.m(5) 🡪 ""