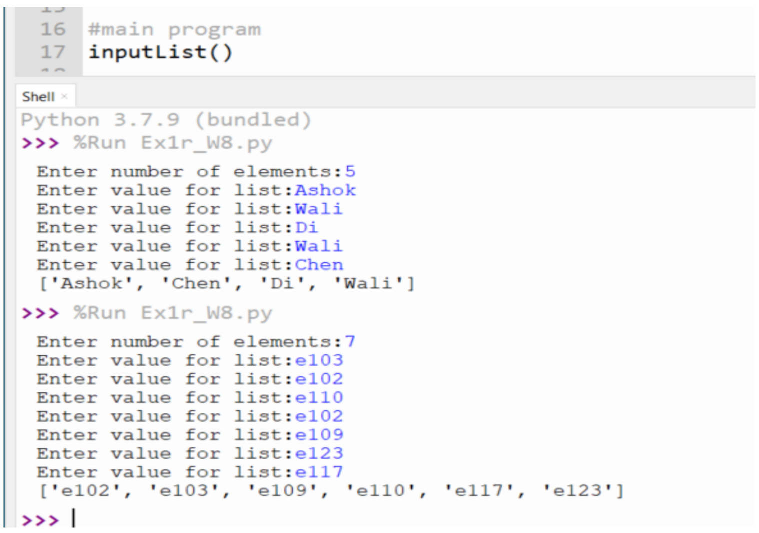
Introduction to Programming with Python
Weekly Programming Assignment – Week 8

All solution files must be submitted at CodeGrade enabled Link for grading.
All solutions must be uploaded on or before 10th December 2021 at 11:59 PM

Exercise 1
Write a program that define subprograms namely:

(i) **inputList():** This prompts the user to enter number of elements (n) that user wants to add into new list. Then the new list should accept values by user input till n times (user input should contain duplicated values). Then the populated list should be passed to another subprogram namely **ListToSet()**.

(ii) **ListToSet():** This accepts a *list* that populated with duplicated values as parameter and remove those duplicates by using set and update again to the passed list. Then print the values of rewritten list in *ascending order*. The test/main program should call only **inputList().** The sample run is here.



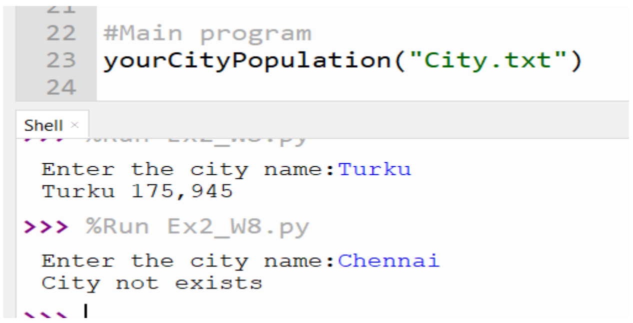
```
#main program
inputList()
```

```
Shell
Python 3.7.9 (bundled)
>>> %Run Ex1r_W8.py

 Enter number of elements:5
 Enter value for list:Ashok
 Enter value for list:Wali
 Enter value for list:Di
 Enter value for list:Wali
 Enter value for list:Chen
 ['Ashok', 'Chen', 'Di', 'Wali']
>>> %Run Ex1r_W8.py

 Enter number of elements:7
 Enter value for list:e103
 Enter value for list:e102
 Enter value for list:e110
 Enter value for list:e102
 Enter value for list:e109
 Enter value for list:e123
 Enter value for list:e117
 ['e102', 'e103', 'e109', 'e110', 'e117', 'e123']
>>> |
```

Exercise 2
The file "**City.txt**" contains city name and its current population. Write a procedure **yourCityPopulation()** that accepts "**City.txt**" only as argument and display the population of specified city based on user input. If the city does not exist, then print appropriate message after the search process is complete. Your code should use dictionary only. Submit both .py and .txt files. [Hint: city as key and population as value]. The sample run is here:



```
#Main program
yourCityPopulation("City.txt")
```

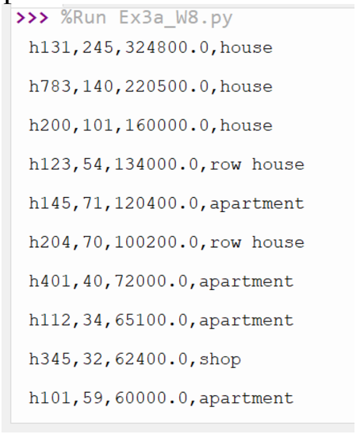
```
Shell

 Enter the city name:Turku
 Turku 175,945
>>> %Run Ex2_W8.py

 Enter the city name:Chennai
 City not exists
```

Exercise 3

"**houseLahti.txt**" contains house information (*house id, size in m^2, sale price* and *type of house*) that are available for sale in Lahti city at present. Write a program that read data from aforenoted file and sort them based on *sale price* (descending order). Then sorted data should be written into new file "**salehouseLahti.txt**". Finally, display the data reading from the file "**salehouseLahti.txt**". Your code must use data structure(s) to execute the aforementioned process.



```
>>> %Run Ex3a_W8.py
  h131,245,324800.0,house

  h783,140,220500.0,house

  h200,101,160000.0,house

  h123,54,134000.0,row house

  h145,71,120400.0,apartment

  h204,70,100200.0,row house

  h401,40,72000.0,apartment

  h112,34,65100.0,apartment

  h345,32,62400.0,shop

  h101,59,60000.0,apartment
```

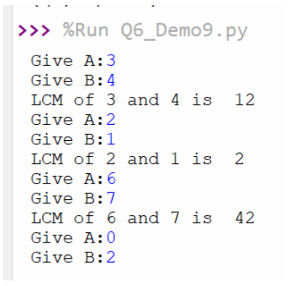
Exercise 4

Write a program that prompts the user to enter two integer values for A and B, and outputs the lowest common multiple (LCM) of these. The LCM of A and B should be taken from dictionary that contains list of LCM. Your code should execute the following when user enter values for A and B

Program gets terminated if either value (A or B) is 0

i. Find the LCM for A and B by looking at the dictionary that stored already (well, when your code run for first time dictionary will not have it, so.). Note, that the order of A and B should not matter; if there is an LCM for numbers A=2 and B=5 in the dictionary, the program then should be able to use that even if numbers are entered as A=5 and B=2.
ii. If the LCM is **not** found in the dictionary, the program then calculates LCM via the function(define on your own) and then save those as LCM at dictionary.
iii. After the LCM is displayed, the program asks for input again for A and B and execute the steps (i) to (iii) until one of the inputs is 0.
iv. Define subprograms to compute LCM, add / search into dictionary and display.

[Hint: a tuple that contain values of A and B is a key: LCM for those A and B is a value. Example: if A is 3 and B is 4 then LCM is 12. The dictionary store these as {(3, 4): 12}]
The sample run is here:

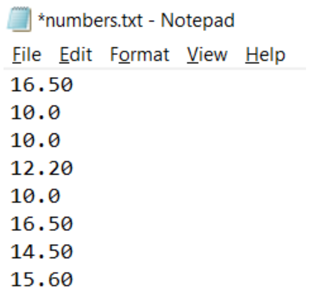


```
>>> %Run Q6_Demo9.py
  Give A:3
  Give B:4
  LCM of 3 and 4 is  12
  Give A:2
  Give B:1
  LCM of 2 and 1 is  2
  Give A:6
  Give B:7
  LCM of 6 and 7 is  42
  Give A:0
  Give B:2
```

Exercise 5

Define a module “**myModule1.py**” that contain the subprograms namely;

(i) **sortData()** which takes any filename that contains data (numbers/names) in single column format and order of sort (integer) then display those in sorted form in vertical order based on user choice (1 - for ascending, and 2 - descending). You can assume that the file contains **one data point** in each row (see example given below).



```
16.50
10.0
10.0
12.20
10.0
16.50
14.50
15.60
```

(ii) Define a function **perfectNumber(n)** which accepts any integer number as argument and returns ***True*** if it is a perfect number. A positive integer is called a perfect number if it is equal to the sum of all of its positive divisors, excluding itself. For example, 6 is the first perfect number because 6 = 3 + 2 + 1. The next is 28 = 14 + 7 + 4 + 2 + 1. However, it should be noted, if the given input **is negative number then it should be converted** as positive number before checking its nature.

Test your module with the main program given here as sample [? -it is incomplete so fix it before using it]. However, it should be noted your module file will be checked with predefined test program (similar to the sample-completed) uploaded in codeGrade. In addition, you should submit “**myModule1.py**” file only [Don’t submit your test/main program] The sample test program is here:

```
#main program
-----?
myModule1.sortData("<your own text file contains values/strings.txt>",<1 or 2>)
if (myModule1.perfectNumber(<any integer number>))) == ?:
    print("Yes it is perfect number")
else:
    print("No it is not")
```

Exercise / task Number	Codegrade link_Moodle for file solution files upload	Points / Marks
1	Exercise1_Week 8	10
2	Exercise2_ Week 8 + City.txt	10
3	Exercise3_Week 8 + houseLahti.txt	15
4	Exercise4_Week 8	20
5	Exercise5_Week 8 (myModule1.py only)	15