CSE474 - Lab Task 1 (a)

1. Solve the following problems from the Hans Peter book ("1. H.P.L. - A Primer on Scientific Programming with Python"):
    a. Exercise 5.9
    b. Exercise 5.13
    c. Exercise 5.28, 5.29, 5.30, 5.31

CSE474 - Lab Task 1 (b)

1. Create a class named *Disaster* with the following attributes:

date_time, location, casualties, injuries, financial_loss, impact_factor, list_of_casualties, list_of_injuries

datet_time should be a datetime object, as defined in the datetime library (you can import it in Python). *location* should be a character string, *casualties* and *injuries* will be integers, and *financial_loss* will be the incurred loss in BDT.

Impact factor will be calculated by $\log_e(injuries+1)$ + sqrt(*casualties*) + $(1.12)^{financial\_loss/100}$

list_of_casualties and *list_of_injuries* will be lists of dictionaries. The dictionaries will have three keys, namely, *name, age,* and *NID_no*. Initially, these lists will be empty, and *casualties* and *injuries* will be set to 0.

2. Create four child classes *Earthquake, Flood, Cyclone,* and *Draught* of the *Disaster* class. Each of them will have one or two additional attributes. *Eathquake* will have a *source* (a string containing a GPS location) and a *seismic scale* (a floating-point number) attribute, *Flood* will have a *water level* (a floating-point number) attribute, *Cyclone* will have a s*ource* (a string containing a GPS location) and a *water level* (a floating-point number) attribute, and *Draught* will have a *list_of_affected_crops* (a list of strings) attribute.
3. Write `__init__` and `__str__` functions for the parent *Disaster* class. Initialize with *date_time* and *location*. If *date_time* is not given in the constructor, set the current date and time. If *location* is missing, assign "Bangladesh". Both these functions should be inherited by the child classes. The function __str__ should print all attributes of an instance.
4. Write `_update_injuries` and `_update_casualties` functions in the parent class (*Disaster*). They will take in either an integer or a list or both as parameters. If only an integer **n** is given as the parameter, add it to the corresponding *casualties* or *no_of_injuries*, and append **n** dictionary items in the list with {"name" : "unknown", "age" : "unknown", "NID_no" : "unknown"}. If a list is given as a parameter, append it to the corresponding list, and update the number of casualties/injuries. If both a number and a list are given as parameters, do both.
5. Write a `_merge` function. If `_merge(event1, event2)` is called, their types, *date_time,* and *location* will be compared. If the type matches, their *location* attributes match, and their *dates* match (not *time*), create a new object by merging their details. The *location* should remain constant.

Merge the lists of injuries/casualties and update their numbers accordingly. Finally, delete the objects event1 and event2, and return the new object.

Submit within 19 February, Saturday, 1:59 PM.
Your code should be well-commented.
The programming language should be Python3.