Assignment #5
Course: CPSC 442 (Python for Data Science)

Please Note: You need to use *try-except* block wherever the need arises for any of the following questions.

1. Create a class called *Shape.* Your task is to find the area and volume of the shapes. Consider the following shapes while designing the code: Sphere and Cylinder.
   The class should have only one constructor.

```
import math
class Test:
    def __init__(self, *args):
        //write code
        return super().__init__()

    def Area(self,val):
        //write code

    def Volume(self,val):
        //write code


def main():
    sphere=Test(2)       #inputs may change
    print('Sphere area:',sphere.Area('Sphere'))
    print('Sphere volume:',sphere.Volume('Sphere'))
    cyclinder=Test(1,2)  #inputs may change
    print('Cyclinder area:',cyclinder.Area('Cyclinder'))
    print('Cyclinder volume:',cyclinder.Volume('Cyclinder'))

if __name__=='__main__':
    main()
```

2. Create a class *Compute.* Your task is to perform operator overloading. Following are the operators that would be overloaded.
   i. +

   The input for the operators would be same sized lists, same sized tuple or variable-sized dictionary.

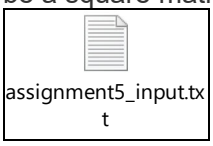
   Following task needs to be completed:

   For **+** operator, if the caller is a list, then you need to add the contents of the two lists and return the same sized list. Similarly, you need to add the contents of the two tuples and return the same sized tuple. For a dictionary, you need to concatenate the two dictionaries. If the key is already present, then you need to merge the contents or values of those keys.

```
Example:

If input is list:

[1,2,3,4] and [3.3,4,5,1.2], then for + operator output would be [4.4,6,8,5.2]
```

```
Same applies for tuples.

If input is dictionary:

{1:’abc’,3:’xyz’,5:’test’} and {2:’test1’,3:’test2’,4:’test3’}, then for + operator
output would be {1:’abc’,3:’xyztest2’,5:’test’, 2:’test1’,4:’test3’}
```

3. You need to create a class for a pathfinder. You would be given an input file with the size of the matrix, start position, destination position, and the obstacle positions. Your task is to output the matrix with the path traced from the start to the destination position. The input file is also attached in Canvas. The number of obstacles would be >= 5. The grid would always be a square matrix.



Note: The system has to be automated. No user inputs must be accepted.

5 <= grid size <= 100

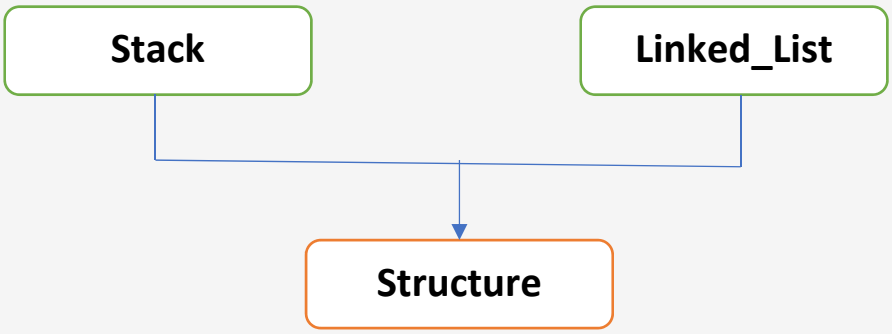

```
Example:

Input file:

7

3 29

5

16

10

34

45


1st line of the input is the grid size.(Square matrix)

2nd line is the start_position(S) end_position(E)

Remaining lines is the obstacle positions(O).
```

```
Initial matrix:
- - S - O - -
- - O - - - -
- O - - - O -
- - - - - - -
E - - - - O -
- - - - - - -
- - O - - - -
Output matrix:
- # S - O - -
# # O - - - -
# O - - - O -
# - - - - - -
E - - - - O -
- - - - - - -
- - O - - - -
# - is the path traced.
The output should be the output matrix or if the destination is not reachable, then r
eturn “Unreachable Destination”.
```

4. Your task is to perform multiple inheritance. You have two classes, *Stack* and *Linked_List* as the base classes. The *Stack* base class would have *push* and *pop* functions. The *Linked_List* would have insert node and delete node functions. The derived class, *Structure* would call one of the two classes to perform the function depending on the type of input data.
   The Stack class would handle numeric data (int , float , and double) only and *Linked_List* class would work on string data.
   Do not use any existing libraries for *Stack* or *Linked_List*. Also, you cannot use push, append, pop functions as well. You have to write functions for that. For Linked_List, you need to create a node and every node would be linked to the next incoming node.
   Please note, the code given is an outline for the system to be created. You may need to make some alterations in the function definitions, etc.

   Hint: You may need to create a Node class to store the element. Take reference from the already created classes.

Architecture:




```
class Stack():
    """description of class"""
    def __init__(self, *args, **kwargs):
        # write code
        return super().__init__(*args, **kwargs)

    def push(self):
        # write code
        pass

    def pop(self):
        # write code
        pass

class LinkedList():
    def __init__(self, *args, **kwargs):
        #write code
        return super().__init__(*args, **kwargs)

    def addElement(self):
        #write code
```

```
        # Hint: You may need to create a Node class to store the element. Take refere
nce from the already created classes.
        pass


    def removeElement(self):
        #write code
        pass


class Structure(Stack,LinkedList):
    def __init__(self, *args, **kwargs):
        return super().__init__(*args, **kwargs)


def main():
    obj=Structure()
    #input from file
    # if input is numeric
    obj.push()

    #if input is string
    obj.addElement()

    #code to pop element
    obj.pop()

    #code to remove element
    obj.removeElement()

    #print the stack
    #print the linked list
```