Q: Implement fuzzy membership functions as discussed in the class. Your program should contain functions for R-function, L-function, triangular-function, and trapezoidal function. The program should ask required arguments for above functions and should generate fuzzy sets.

Solution:

Code available at:
https://colab.research.google.com/drive/1pLsICoPWl1xjBNVOzkTznbIGVDL0aIZe?usp=sharing

```
class FuzzySet:
    def __init__(self):
        self.elements = []
        self.memberships = []

    def add_element(self, element):
            self.elements.append(element)

    def r_function(self, alpha_value, beta_value, set1):
            mem = []
            for element in set1.elements:

                if element <= alpha_value:
                    mem.append(0)
                elif element >= beta_value:
                    mem.append(1)
                else:
                    mem_value = (float(element) - float(alpha_value)) / (float(beta_value) - float(alpha_value))
                    mem.append(round(mem_value, 2))
            for element, membership in zip(set1.elements, mem):
              print(f"{element} - {membership}")

    def l_function(self, alpha_value, beta_value, set1):
        mem = []
        for element in set1.elements:

            if element <= alpha_value:
                mem.append(1)
            elif element >= beta_value:
                mem.append(0)
            else:
                mem_value = (float(beta_value) - float(element)) / (float(beta_value) - float(alpha_value))
                mem.append(round(mem_value, 2))

        for element, membership in zip(set1.elements, mem):
          print(f"{element} - {membership}")


    def t_function(self, alpha_value, beta_value, gamma_value, set1):
        mem = []
        for element in set1.elements:

            if element < alpha_value or element > gamma_value:
                mem.append(0)

            elif alpha_value <= element <= beta_value:
                mem_value = (float(element) - float(alpha_value)) / (float(beta_value) - float(alpha_value))
                mem.append(round(mem_value, 2))
            else:
                mem_value = (float(gamma_value) - float(element)) / (float(gamma_value) - float(beta_value))
                mem.append(round(mem_value, 2))

        for element, membership in zip(set1.elements, mem):
          print(f"{element} - {membership}")



    def tra_function(self, alpha_value, beta_value, gamma_value, sigma_value, set1):
        mem = []
        for element in set1.elements:

            if element < alpha_value or element > sigma_value:
                mem.append(0)
            elif alpha_value <= element <= beta_value:
                mem_value = (float(element) - float(alpha_value)) / (float(beta_value) - float(alpha_value))
                mem.append(round(mem_value, 2))

            elif beta_value <= element <= gamma_value:
                mem.append(1)
            else:
                mem_value =(float(sigma_value) - float(element)) / (float(sigma_value) - float(gamma_value))
                mem.append(round(mem_value, 2))
        for element, membership in zip(set1.elements, mem):
          print(f"{element} - {membership}")

    def print_set(self):
        for i in range(len(self.elements)):
            print(self.elements[i])

set1 = FuzzySet()

n = int(input("Enter the number of elements in set: "))

for i in range(n):
    element = input(f"Enter element {i+1} in set: ")
    set1.add_element(element)



print("\nSet:")
set1.print_set()



rFunction = FuzzySet()

alpha_value = input(f"\nEnter the alpha value : ")
beta_value = input(f"\nEnter the beta value : ")
gamma_value = input(f"\nEnter the gamma value : ")
sigma_value = input(f"\nEnter the sigma value : ")

print("\nR-Function:")
```

```
rFunction.r_function(alpha_value, beta_value, set1)


lFunction = FuzzySet()
print("\nL-Function:")
lFunction.l_function(alpha_value, beta_value, set1)


tFunction = FuzzySet()
print("\nTriangular-Function:")
tFunction.t_function(alpha_value, beta_value, gamma_value, set1)


traFunction = FuzzySet()
print("\nTrapezoid-Function:")
traFunction.tra_function(alpha_value, beta_value, gamma_value,sigma_value, set1)
```

Output

```
Enter the number of elements in set: 7
Enter element 1 in set: 20
Enter element 2 in set: 30
Enter element 3 in set: 40
Enter element 4 in set: 50
Enter element 5 in set: 60
Enter element 6 in set: 70
Enter element 7 in set: 80

Set:
20
30
40
50
60
70
80

Enter the alpha value : 40

Enter the beta value : 70

Enter the gamma value : 70

Enter the sigma value : 70

R-Function:
```

https://colab.research.google.com/drive/1pLsICoPWl1xjBNVOzkTznbIGVDL0aIZe?usp=sharing

20 - 0

30 - 0

40 - 0

50 - 0.33

60 - 0.67

70 - 1

80 - 1

L-Function:

20 - 1

30 - 1

40 - 1

50 - 0.67

60 - 0.33

70 - 0

80 - 0

Triangular-Function:

20 - 0

30 - 0

40 - 0.0

50 - 0.33

60 - 0.67

70 - 1.0

80 - 0

Trapezoid-Function:

20 - 0

30 - 0

40 - 0.0

50 - 0.33

60 - 0.67

70 - 1.0

80 - 0

https://colab.research.google.com/drive/1pLsICoPWl1xjBNVOzkTznbIGVDL0aIZe?usp=sharing