**Q:** Implement crisp sets as discussed in the class. Your program should ask user to input the set elements and corresponding membership values. The program should accept the valid membership values only.  For the set, use appropriate data structure to store fuzzy element with its corresponding membership value. Also write functions to implement Union, Intersection, Complement and Subset operations over the sets.

Solution:

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

class CrispSet:

def \_\_init\_\_(self):

self.elements = []

self.memberships = []

def add\_element(self, element, membership):

if membership < 0 or membership > 1:

print("Invalid membership value. Membership value should be between 0 and 1.")

again\_membership = float(input(f"Enter membership value of {element} AGAIN! (between 0 and 1): "))

self.elements.append(element)

self.memberships.append(again\_membership)

else:

self.elements.append(element)

self.memberships.append(membership)

def union(self, other\_set):

union\_set = CrispSet()

for i in range(len(self.elements)):

union\_set.add\_element(self.elements[i], self.memberships[i])

for i in range(len(other\_set.elements)):

if other\_set.elements[i] not in union\_set.elements:

union\_set.add\_element(other\_set.elements[i], other\_set.memberships[i])

else:

index = union\_set.elements.index(other\_set.elements[i])

if other\_set.memberships[i] > union\_set.memberships[index]:

union\_set.memberships[index] = other\_set.memberships[i]

return union\_set

def intersection(self, other\_set):

intersection\_set = CrispSet()

for i in range(len(self.elements)):

if self.elements[i] in other\_set.elements:

index = other\_set.elements.index(self.elements[i])

if self.memberships[i] < other\_set.memberships[index]:

intersection\_set.add\_element(self.elements[i], self.memberships[i])

else:

intersection\_set.add\_element(self.elements[i], other\_set.memberships[index])

return intersection\_set

def complement(self):

complement\_set = CrispSet()

for i in range(len(self.elements)):

diff = 1-self.memberships[i]

if(diff < 0 ):

newMem = max(0, diff)

elif(diff > 1):

newMem = min(1, diff)

else:

newMem = diff

complement\_set.add\_element(self.elements[i], newMem)

# complement\_set.add\_element(self.elements[i], 1 - self.memberships[i])

return complement\_set

def subset(self, other\_set):

for i in range(len(other\_set.elements)):

if other\_set.elements[i] not in self.elements or self.memberships[self.elements.index(other\_set.elements[i])] < other\_set.memberships[i]:

return False

return True

def print\_set(self):

for i in range(len(self.elements)):

print(self.elements[i], self.memberships[i])

set1 = CrispSet()

n = int(input("Enter the number of elements in set 1: "))

for i in range(n):

element = input(f"Enter element {i+1} in set 1: ")

membership = float(input(f"Enter membership value of {element} in set 1 (between 0 and 1): "))

set1.add\_element(element, membership)

set2 = CrispSet()

n = int(input("\nEnter the number of elements in set 2: "))

for i in range(n):

element = input(f"Enter element {i+1} in set 2: ")

membership = float(input(f"Enter membership value of {element} in set 2 (between 0 and 1): "))

set2.add\_element(element, membership)

union\_set = set1.union(set2)

intersection\_set = set1.intersection(set2)

complement\_set = set1.complement()

print("\nSet 1:")

set1.print\_set()

print("\nSet 2:")

set2.print\_set()

print("\nUnion Set:")

union\_set.print\_set()

print("\nIntersection Set:")

intersection\_set.print\_set()

print("\nComplement of Set 1:")

complement\_set.print\_set()

print("\nIs Set 2 a subset of Set 1?", set1.subset(set2))

print("Is Set 1 a subset of Set 2?", set2.subset(set1))

**OUTPUT**

Enter the number of elements in set 1: 5

Enter element 1 in set 1: 2

Enter membership value of 2 in set 1 (between 0 and 1): 0.5

Enter element 2 in set 1: 3

Enter membership value of 3 in set 1 (between 0 and 1): 0.4

Enter element 3 in set 1: 4

Enter membership value of 4 in set 1 (between 0 and 1): 0.3

Enter element 4 in set 1: 6

Enter membership value of 6 in set 1 (between 0 and 1): 0.9

Enter element 5 in set 1: 9

Enter membership value of 9 in set 1 (between 0 and 1): 1

Enter the number of elements in set 2: 4

Enter element 1 in set 2: 1

Enter membership value of 1 in set 2 (between 0 and 1): 0.6

Enter element 2 in set 2: 2

Enter membership value of 2 in set 2 (between 0 and 1): 0.2

Enter element 3 in set 2: 3

Enter membership value of 3 in set 2 (between 0 and 1): 0.02

Enter element 4 in set 2: 10

Enter membership value of 10 in set 2 (between 0 and 1): 1

Set 1:

2 0.5

3 0.4

4 0.3

6 0.9

9 1.0

Set 2:

1 0.6

2 0.2

3 0.02

10 1.0

Union Set:

2 0.5

3 0.4

4 0.3

6 0.9

9 1.0

1 0.6

10 1.0

Intersection Set:

2 0.2

3 0.02

Complement of Set 1:

2 0.5

3 0.6

4 0.7

6 0.09999999999999998

9 0.0

Is Set 2 a subset of Set 1? False

Is Set 1 a subset of Set 2? False