BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai – 400058-India
Department of Computer Engineering

Name	Rishabh Santosh Shenoy
UID no.	2023300222
Experiment No.	8

AIM:	Fuzzy set operations:
Program 1	
PROBLEM STATEMENT :	7. For the two given fuzzy sets $\underset{\sim}{A} = \left\{\frac{0.1}{0} + \frac{0.2}{1} + \frac{0.4}{2} + \frac{0.6}{3} + \frac{1}{4}\right\}$ $\underset{\sim}{B} = \left\{\frac{1}{0} + \frac{0.5}{1} + \frac{0.7}{2} + \frac{0.3}{3} + \frac{0}{4}\right\}$ find the following: **(a)** $\underset{\sim}{A} \cup \underset{\sim}{B}$; **(b)** $\underset{\sim}{A} \cap \underset{\sim}{B}$; **(c)** $\overline{\underset{\sim}{A}}$; **(d)** $\overline{\underset{\sim}{B}}$; **(e)** $\underset{\sim}{A} \cup \overline{\underset{\sim}{A}}$; **(f)** $\underset{\sim}{A} \cap \overline{\underset{\sim}{A}}$; **(g)** $\underset{\sim}{B} \cup \overline{\underset{\sim}{B}}$; **(h)** $\underset{\sim}{B} \cap \overline{\underset{\sim}{B}}$; **(i)** $\underset{\sim}{A} \cap \overline{\underset{\sim}{B}}$; **(j)** $\underset{\sim}{A} \cup \overline{\underset{\sim}{B}}$; **(k)** $\underset{\sim}{B} \cap \overline{\underset{\sim}{A}}$; **(l)** $\underset{\sim}{B} \cup \overline{\underset{\sim}{A}}$; **(m)** $\overline{\underset{\sim}{A} \cup \underset{\sim}{B}}$; **(n)** $\overline{\underset{\sim}{A}} \cap \overline{\underset{\sim}{B}}$

PROGRAM:	#question 7
	A = [0.1, 0.2, 0.4, 0.6, 1.0]
	B = [1.0, 0.5, 0.7, 0.3, 0.0]
	def fuzzy_union(set1, set2):
	  return [round(max(a, b), 2) for a, b in zip(set1, set2)]
	
	def fuzzy_intersection(set1, set2):
	  return [round(min(a, b), 2) for a, b in zip(set1, set2)]
	
	def fuzzy_complement(set1):
	  return [round(1 - a, 2) for a in set1]
	
	A_union_B = fuzzy_union(A, B)
	A_inter_B = fuzzy_intersection(A, B)

```
A_comp = fuzzy_complement(A)
B_comp = fuzzy_complement(B)
A_union_A = fuzzy_union(A, fuzzy_complement(A))
A_inter_A = fuzzy_intersection(A, fuzzy_complement(A))
B_union_B = fuzzy_union(B, fuzzy_complement(B))
B_inter_B = fuzzy_intersection(B, fuzzy_complement(B))
A_inter_B_comp = fuzzy_intersection(A,fuzzy_complement(B))
A_union_B_comp = fuzzy_union(A,fuzzy_complement(B))
B_union_A = fuzzy_union(B, fuzzy_complement(A))
B_inter_A = fuzzy_intersection(B, fuzzy_complement(A))
A_union_B_comp = fuzzy_complement(fuzzy_union(A, B))
A_inter_B_comp =
fuzzy_intersection(fuzzy_complement(A),fuzzy_complement(B))

print("A =", A)
print("B =", B)
print("\n(a) A ∪ B =", A_union_B)
print("(b) A ∩ B =", A_inter_B)
print("(c) A' =", A_comp)
print("(d) B' =", B_comp)
print("(e) A ∪ A' =", A_union_A)
print("(f) A ∩ A' =", A_inter_A)
print("(g) B ∪ B' =", B_union_B)
print("(h) B ∩ B' =", B_inter_B)
print("(i) A ∩ B' =", A_inter_B_comp)
print("(j) A ∪ B' =", A_union_B_comp)
print("(k) B ∩ A' =", B_inter_A)
print("(l) B ∪ A' =", B_union_A)
print("(m) (A ∪ B)' =", A_union_B_comp)
print("(n) (A' ∩ B') =", A_inter_B_comp)
```

PROGRAM WITH LIBRARY:

```
import numpy as np
import skfuzzy as fuzz

A = np.array([0.1, 0.2, 0.4, 0.6, 1.0])
B = np.array([1.0, 0.5, 0.7, 0.3, 0.0])

A_union_B = fuzz.fuzzy_or(A, B, np.arange(len(A)), np.arange(len(B)))[1]
A_inter_B = fuzz.fuzzy_and(A, B, np.arange(len(A)), np.arange(len(B)))[1]
A_comp = fuzz.fuzzy_not(A)
B_comp = fuzz.fuzzy_not(B)

A_union_A = fuzz.fuzzy_or(A, A_comp, np.arange(len(A)),
np.arange(len(A)))[1]
A_inter_A = fuzz.fuzzy_and(A, A_comp, np.arange(len(A)),
np.arange(len(A)))[1]
B_union_B = fuzz.fuzzy_or(B, B_comp, np.arange(len(B)),
np.arange(len(B)))[1]
B_inter_B = fuzz.fuzzy_and(B, B_comp, np.arange(len(B)),
np.arange(len(B)))[1]

A_inter_B_comp = fuzz.fuzzy_and(A, B_comp, np.arange(len(A)),
np.arange(len(B)))[1]
A_union_B_comp = fuzz.fuzzy_or(A, B_comp, np.arange(len(A)),
np.arange(len(B)))[1]
B_inter_A = fuzz.fuzzy_and(B, A_comp, np.arange(len(B)),
np.arange(len(A)))[1]
B_union_A = fuzz.fuzzy_or(B, A_comp, np.arange(len(B)),
np.arange(len(A)))[1]

A_union_B_comp = fuzz.fuzzy_not(fuzz.fuzzy_or(A, B, np.arange(len(A)),
np.arange(len(B)))[1])
A_inter_B_comp = fuzz.fuzzy_and(A_comp, B_comp, np.arange(len(A)),
np.arange(len(B)))[1]

def r(x): return np.round(x, 2)

print("A =", A)
print("B =", B)
```

```
print("\n(a) A ∪ B =", r(A_union_B))
print("(b) A ∩ B =", r(A_inter_B))
print("(c) A' =", r(A_comp))
print("(d) B' =", r(B_comp))
print("(e) A ∪ A' =", r(A_union_A))
print("(f) A ∩ A' =", r(A_inter_A))
print("(g) B ∪ B' =", r(B_union_B))
print("(h) B ∩ B' =", r(B_inter_B))
print("(i) A ∩ B' =", r(A_inter_B_comp))
print("(j) A ∪ B' =", r(A_union_B_comp))
print("(k) B ∩ A' =", r(B_inter_A))
print("(l) B ∪ A' =", r(B_union_A))
print("(m) (A ∪ B)' =", r(A_union_B_comp))
print("(n) (A' ∩ B') =", r(A_inter_B_comp))
```

RESULT:

```
psipl@psipl-OptiPlex-SFF-7010:~/Desktop/Rishabh_Shenoy$ /usr/bin/python3 /home/psipl/Desktop/Rishabh_Shenoy/Exp08/part1.py
A = [0.1, 0.2, 0.4, 0.6, 1.0]
B = [1.0, 0.5, 0.7, 0.3, 0.0]

(a) A ∪ B = [1.0, 0.5, 0.7, 0.6, 1.0]
(b) A ∩ B = [0.1, 0.2, 0.4, 0.3, 0.0]
(c) A' = [0.9, 0.8, 0.6, 0.4, 0.0]
(d) B' = [0.0, 0.5, 0.3, 0.7, 1.0]
(e) A ∪ A' = [0.9, 0.8, 0.6, 0.6, 1.0]
(f) A ∩ A' = [0.1, 0.2, 0.4, 0.4, 0.0]
(g) B ∪ B' = [1.0, 0.5, 0.7, 0.7, 1.0]
(h) B ∩ B' = [0.0, 0.5, 0.3, 0.3, 0.0]
(i) A ∩ B' = [0.0, 0.5, 0.3, 0.4, 0.0]
(j) A ∪ B' = [0.0, 0.5, 0.3, 0.4, 0.0]
(k) B ∩ A' = [0.9, 0.5, 0.6, 0.3, 0.0]
(l) B ∪ A' = [1.0, 0.8, 0.7, 0.4, 0.0]
(m) (A ∪ B)' = [0.0, 0.5, 0.3, 0.4, 0.0]
(n) (A' ∩ B') = [0.0, 0.5, 0.3, 0.4, 0.0]
```

Program 2

PROBLEM STATEMENT :

9. Consider two fuzzy sets

$$\underset{\sim}{A} = \left\{\frac{0.2}{1} + \frac{0.3}{2} + \frac{0.4}{3} + \frac{0.5}{4}\right\}$$

$$\underset{\sim}{B} = \left\{\frac{0.1}{1} + \frac{0.2}{2} + \frac{0.2}{3} + \frac{1}{4}\right\}$$

Find the algebraic sum, algebraic product, bounded sum and bounded difference of the given fuzzy sets.

PROGRAM WITHOUT

```
#question 9
A = [0.2, 0.3, 0.4, 0.5]
```

LIBRARY:

```
B = [0.1, 0.2, 0.2, 1.0]

def algebraic_sum(A, B):
   return [round((a + b) - (a * b), 2) for a, b in zip(A, B)]

def algebraic_product(A, B):
   return [round(a * b, 2) for a, b in zip(A, B)]

def bounded_sum(A, B):
   return [round(min(1, a + b), 2) for a, b in zip(A, B)]

def bounded_difference(A, B):
   return [round(max(0, a - b), 2) for a, b in zip(A, B)]

alg_sum = algebraic_sum(A, B)
alg_prod = algebraic_product(A, B)
bound_sum = bounded_sum(A, B)
bound_diff = bounded_difference(A, B)

print("A =", A)
print("B =", B)
print("\n(a) Algebraic Sum (A + B - A*B) =", alg_sum)
print("(b) Algebraic Product (A * B) =", alg_prod)
print("(c) Bounded Sum (min(1, A + B)) =", bound_sum)
print("(d) Bounded Difference (max(0, A - B)) =", bound_diff)
```

PROGRAM WITH LIBRARY:

```
#Q9 using library
import numpy as np
import skfuzzy as fuzz

A = np.array([0.2, 0.3, 0.4, 0.5])
B = np.array([0.1, 0.2, 0.2, 1.0])

algebraic_sum = (A + B) - (A * B)
algebraic_product = A * B
bounded_sum = np.minimum(1, A + B)
bounded_difference = np.maximum(0, A - B)
```

```
algebraic_sum = np.round(algebraic_sum, 2)
algebraic_product = np.round(algebraic_product, 2)
bounded_sum = np.round(bounded_sum, 2)
bounded_difference = np.round(bounded_difference, 2)

print("A =", A)
print("B =", B)
print("\n(a) Algebraic Sum =", algebraic_sum)
print("(b) Algebraic Product =", algebraic_product)
print("(c) Bounded Sum =", bounded_sum)
print("(d) Bounded Difference =", bounded_difference)
```

RESULT:

```
psipl@psipl-OptiPlex-SFF-7010:~/Desktop/Rishabh_Shenoy$ /usr/bin/python3 /home/psipl/Desktop/Rishabh_Shenoy/Exp08/part2.py
A = [0.2, 0.3, 0.4, 0.5]
B = [0.1, 0.2, 0.2, 1.0]

(a) Algebraic Sum (A + B - A*B) = [0.28, 0.44, 0.52, 1.0]
(b) Algebraic Product (A * B) = [0.02, 0.06, 0.08, 0.5]
(c) Bounded Sum (min(1, A + B)) = [0.3, 0.5, 0.6, 1]
(d) Bounded Difference (max(0, A - B)) = [0.1, 0.1, 0.2, 0]
```

```
psipl@psipl-OptiPlex-SFF-7010:~/Desktop/Rishabh_Shenoy$ /usr/bin/python3 /home/psipl/Desktop/Rishabh_Shenoy/Exp08/part3.py
A = [0.2 0.3 0.4 0.5]
B = [0.1 0.2 0.2 1. ]

(a) Algebraic Sum = [0.28 0.44 0.52 1.  ]
(b) Algebraic Product = [0.02 0.06 0.08 0.5 ]
(c) Bounded Sum = [0.3 0.5 0.6 1. ]
(d) Bounded Difference = [0.1 0.1 0.2 0. ]
```

Program 3

PROBLEM STATEMENT:

2. Consider the following two fuzzy sets:

$$\underset{\sim}{A} = \left\{\frac{0.3}{x_1} + \frac{0.7}{x_2} + \frac{1}{x_3}\right\}$$

and $$\underset{\sim}{B} = \left\{\frac{0.4}{y_1} + \frac{0.9}{y_2}\right\}$$

Perform the Cartesian product over these given fuzzy sets.

PROGRAM:

```
#unit 11 q2 without library
A = [0.3, 0.7, 1.0]
B = [0.4, 0.9]
```

```
def cartesian_product(A, B):
    R = []
    for a in A:
        row = []
        for b in B:
            row.append(round(min(a, b), 2))
        R.append(row)
    return R

def max_min_composition(R, S):
    T = []
    for i in range(len(R)):
        row = []
        for j in range(len(S[0])):
            vals = [min(R[i][k], S[k][j]) for k in range(len(S))]
            row.append(round(max(vals), 2))
        T.append(row)
    return T

def max_product_composition(R, S):
    T = []
    for i in range(len(R)):
        row = []
        for j in range(len(S[0])):
            vals = [R[i][k] * S[k][j] for k in range(len(S))]
            row.append(round(max(vals), 2))
        T.append(row)
    return T

S = [[1, 0.5, 0.3],
     [0.8, 0.4, 0.7]]

R = cartesian_product(A, B)
max_min_T = max_min_composition(R, S)
max_prod_T = max_product_composition(R, S)

print("A =", A)
print("B =", B)
```

	print("\nFuzzy Cartesian Product R = A × B:")
	for row in R:
	  print(row)
	
	print("\nMax–Min Composition (T = R∘S):")
	for row in max_min_T:
	  print(row)
	
	print("\nMax–Product Composition (T = R×S):")
	for row in max_prod_T:
	  print(row)

PROGRAM WITH LIBRARY:	import numpy as np
	import skfuzzy as fuzz
	
	A = np.array([0.3, 0.7, 1.0])
	B = np.array([0.4, 0.9])
	
	R = np.minimum(A[:, np.newaxis], B[np.newaxis, :])
	
	S = np.array([
	  [1.0, 0.5, 0.3],
	  [0.8, 0.4, 0.7]
	])
	
	T_maxmin = fuzz.relation_min(R, S)
	
	T_maxprod = fuzz.relation_product(R, S)
	
	np.set_printoptions(precision=2, suppress=True)
	print("A =", A)
	print("B =", B)
	print("\nFuzzy Cartesian Product R = A × B:")
	print(R)
	
	#print("\nMax–Min Composition (T = R∘S):")
	#print(T_maxmin)

```
#print("\nMax–Product Composition (T = R×S):")
#print(T_maxprod)
```

RESULT:

```
  [0.64 0.32 0.56]]
psipl@psipl-OptiPlex-SFF-7010:~/Desktop/Rishabh_Shenoy$ /usr/bin/
A = [0.3, 0.7, 1.0]
B = [0.4, 0.9]

Fuzzy Cartesian Product R = A × B:
[0.3, 0.3]
[0.4, 0.7]
[0.4, 0.9]
psipl@psipl-OptiPlex-SFF-7010:~/Desktop/Rishabh_Shenoy$
```

Program 4

PROBLEM STATEMENT:

3. Two fuzzy relations are given by

$$\underset{\sim}{R} = \begin{array}{c} \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{cc} y_1 & y_2 \end{array} \\ \begin{bmatrix} 0.6 & 0.3 \\ 0.2 & 0.9 \end{bmatrix} \end{array}$$

and

$$\underset{\sim}{S} = \begin{array}{c} \\ y_1 \\ y_2 \end{array} \begin{array}{c} \begin{array}{ccc} z_1 & z_2 & z_3 \end{array} \\ \begin{bmatrix} 1 & 0.5 & 0.3 \\ 0.8 & 0.4 & 0.7 \end{bmatrix} \end{array}$$

Obtain fuzzy relation $\underset{\sim}{T}$ as a composition between the fuzzy relations.

PROGRAM:

```
#unit 11 q3
import numpy as np

R = np.array([
    [0.6, 0.3],
    [0.8, 0.4]
])

S = np.array([
    [0.8, 0.4, 0.7],
```

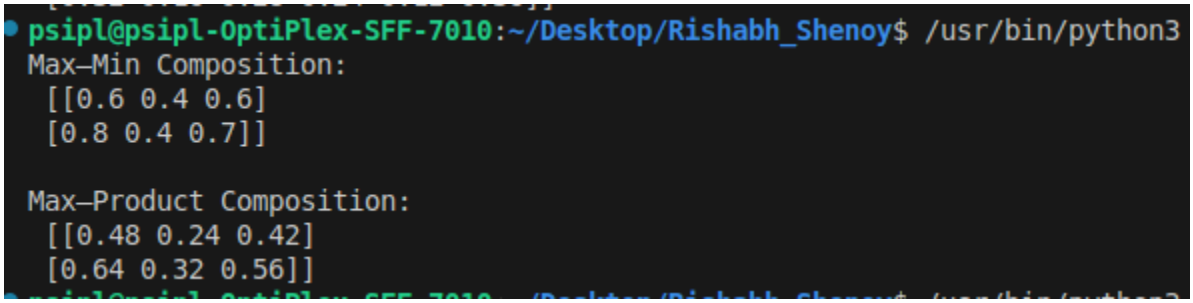
```
    [0.6, 0.3, 0.9]
])

T_maxmin = np.zeros((R.shape[0], S.shape[1]))
for i in range(R.shape[0]):
    for j in range(S.shape[1]):
        T_maxmin[i, j] = np.max(np.minimum(R[i, :], S[:, j]))

T_maxprod = np.zeros((R.shape[0], S.shape[1]))
for i in range(R.shape[0]):
    for j in range(S.shape[1]):
        T_maxprod[i, j] = np.max(R[i, :] * S[:, j])

np.set_printoptions(precision=2, suppress=True)
print("Max–Min Composition:\n", T_maxmin)
print("\nMax–Product Composition:\n", T_maxprod)
```

PROGRAM WITH LIBRARY:

```
import numpy as np
import skfuzzy as fuzz

R = np.array([
    [0.6, 0.3],
    [0.8, 0.4]
])

S = np.array([
    [0.8, 0.4, 0.7],
    [0.6, 0.3, 0.9]
])


T_maxmin = fuzz.relation_min(R, S)

T_maxprod = fuzz.relation_product(R, S)

np.set_printoptions(precision=2, suppress=True)
print("Fuzzy Relation R:\n", R)
print("\nFuzzy Relation S:\n", S)
```

```
print("\nMax–Min Composition (T = R ∘ S):\n", T_maxmin)
print("\nMax–Product Composition (T = R × S):\n", T_maxprod)
```

RESULT:



CONCLUSION:

In this experiment, I have understood how fuzzy relation operations work and how they help in dealing with imprecise or uncertain data. I learned to perform different operations like the fuzzy Cartesian product, max–min composition, and max–product composition, which are used to combine fuzzy sets and analyze their relationships. These operations are very useful in real-life systems such as decision-making, control systems, and pattern recognition, where clear boundaries between values do not exist.

I also understood the difference between logical reasoning using max–min composition and probabilistic reasoning using max–product composition. By implementing the operations both manually and using the scikit-fuzzy library, I gained a deeper understanding of how fuzzy logic can be applied computationally. Overall, this experiment helped me understand how fuzzy logic provides a powerful way to model real-world problems where traditional binary logic fails.