Name	Rayyan Ahmed Siddiqui
UID no.	2023800118
Experiment No.	8
Batch	B3
Course Code	**CS303**

AIM:	Unsupervised Training Algorithms (LVQ)
Program 1	
PROBLEM STATEMENT :	Define fuzzy sets for product **Defect Level (D)** and **Acceptability (A)** over five items: $D = \{0.1/item_1,\ 0.5/item_2,\ 0.8/item_3,\ 0.3/item_4,\ 0.2/item_5\}$ $A = \{0.9/item_1,\ 0.6/item_2,\ 0.2/item_3,\ 0.7/item_4,\ 0.8/item_5\}$ Find: a) $D \cup A$ b) $D \cap A$ c) $\overline{D}$ d) $A - D$ e) $\overline{A \cap D}$ f) $\overline{D} \cup A$
Code	import pandas as pd from tabulate import tabulate import numpy as np # ========== # 🔹 FUZZY SET DEFINITIONS # ========== D = {'item1': 0.1, 'item2': 0.5, 'item3': 0.8, 'item4': 0.3, 'item5': 0.2} A = {'item1': 0.9, 'item2': 0.6, 'item3': 0.2, 'item4': 0.7, 'item5': 0.8} U = list(D.keys()) # Universe of discourse # ========== # 🔹 FUZZY SET OPERATIONS # ========== def fuzzy_union(X, Y):

```
    """Fuzzy Union: μX∪Y(x) = max(μX(x), μY(x))"""
    return {item: round(max(X.get(item, 0), Y.get(item, 0)), 2) for item in U}

def fuzzy_intersection(X, Y):
    """Fuzzy Intersection: μX∩Y(x) = min(μX(x), μY(x))"""
    return {item: round(min(X.get(item, 0), Y.get(item, 0)), 2) for item in U}

def fuzzy_complement(X):
    """Fuzzy Complement: μ¬X(x) = 1 - μX(x)"""
    return {item: round(1 - X.get(item, 0), 2) for item in U}

def fuzzy_difference(X, Y):
    """Fuzzy Difference: μX−Y(x) = min(μX(x), 1 - μY(x))"""
    return fuzzy_intersection(X, fuzzy_complement(Y))


# ==========================================
# 🔹 COMPUTE BASIC OPERATIONS
# ==========================================
D_union_A = fuzzy_union(D, A)
D_intersection_A = fuzzy_intersection(D, A)
D_complement = fuzzy_complement(D)
A_complement = fuzzy_complement(A)
A_difference_D = fuzzy_difference(A, D)
A_intersection_D_complement = fuzzy_intersection(A, D_complement)
D_complement_union_A = fuzzy_union(D_complement, A)
A_intersection_D_complement_complement =
fuzzy_complement(D_intersection_A)  # ¬(A ∩ D)

# ==========================================
# 🔹 TABULAR DISPLAY USING PANDAS + TABULATE
# ==========================================
results = {
    'D': D,
    'A': A,
    'D ∪ A': D_union_A,
    'D ∩ A': D_intersection_A,
    '¬D': D_complement,
    '¬A': A_complement,
    'A - D': A_difference_D,
    'A ∩ ¬D': A_intersection_D_complement,
    '¬(A ∩ D)': A_intersection_D_complement_complement,
    '¬D ∪ A': D_complement_union_A
}

df = pd.DataFrame(results)
df = df[['D', 'A', 'D ∪ A', 'D ∩ A', '¬D', '¬A', 'A - D', 'A ∩ ¬D', '¬(A ∩ D)', '¬D
∪ A']]

print("\n============================")
```

	print(" FUZZY SET OPERATIONS TABLE") print("==============================\n") print(tabulate(df, headers='keys', tablefmt='fancy_grid', showindex=True))
Result	========== FUZZY SET OPERATIONS TABLE ========== ╒═══════╤═══╤═══╤═══════╤═══════╤═══╤═══╤═══════╤═══════╤═══════════╤═══════╕ │ │ D │ A │ D ∪ A │ D ∩ A │ ¬D │ ¬A │ A − D │ A ∩ ¬D │ ¬(A ∩ D) │ ¬D ∪ A │ ╞═══════╪═══╪═══╪═══════╪═══════╪═══╪═══╪═══════╪═══════╪═══════════╪═══════╡ │ item1 │ 0.1 │ 0.9 │ 0.9 │ 0.1 │ 0.9 │ 0.1 │ 0.9 │ 0.9 │ 0.9 │ 0.9 │ ├───────┼───┼───┼───────┼───────┼───┼───┼───────┼───────┼───────────┼───────┤ │ item2 │ 0.5 │ 0.6 │ 0.6 │ 0.5 │ 0.5 │ 0.4 │ 0.5 │ 0.5 │ 0.5 │ 0.6 │ ├───────┼───┼───┼───────┼───────┼───┼───┼───────┼───────┼───────────┼───────┤ │ item3 │ 0.8 │ 0.2 │ 0.8 │ 0.2 │ 0.2 │ 0.8 │ 0.2 │ 0.2 │ 0.8 │ 0.2 │ ├───────┼───┼───┼───────┼───────┼───┼───┼───────┼───────┼───────────┼───────┤ │ item4 │ 0.3 │ 0.7 │ 0.7 │ 0.3 │ 0.7 │ 0.3 │ 0.7 │ 0.7 │ 0.7 │ 0.7 │ ├───────┼───┼───┼───────┼───────┼───┼───┼───────┼───────┼───────────┼───────┤ │ item5 │ 0.2 │ 0.8 │ 0.8 │ 0.2 │ 0.8 │ 0.2 │ 0.8 │ 0.8 │ 0.8 │ 0.8 │ ╘═══════╧═══╧═══╧═══════╧═══════╧═══╧═══╧═══════╧═══════╧═══════════╧═══════╛
Program 2	
PROBLEM STATEMENT :	Air quality index (AQI) is influenced by **pollutant level (P)** and **weather condition (W)**, which further determine **health risk (H)**. Given: $$R_1(P,W) = \begin{bmatrix} 0.9 & 0.4 & 0.2 \\ 0.6 & 0.7 & 0.5 \\ 0.3 & 0.5 & 0.8 \end{bmatrix}, \quad R_2(W,H) = \begin{bmatrix} 0.8 & 0.6 & 0.4 \\ 0.5 & 0.7 & 0.6 \\ 0.3 & 0.5 & 0.9 \end{bmatrix}$$ Compute $R(P,H) = R_1(P,W) \circ R_2(W,H)$ using **max–min composition**. Then, if $P = [0.2, 0.8, 0.6]$, find the fuzzy health risk $H' = P \circ R(P,H)$.
Code	import numpy as np def max_min_composition(R1, R2): """ Computes the max-min composition (R1 o R2) of two fuzzy relations. For matrices R1 (m x n) and R2 (n x p), the resulting matrix R (m x p) has elements R[i, j] = max_k(min(R1[i, k], R2[k, j])). """ # Get the dimensions of the input relations m, n = R1.shape n2, p = R2.shape

```
    # Check for compatibility (number of columns in R1 must equal rows in
R2)
    if n != n2:
        raise ValueError("Incompatible dimensions for max-min composition.
R1 must have the same number of columns as R2 has rows.")

    # Initialize the result matrix R with zeros (m x p)
    R = np.zeros((m, p))

    # Perform the max-min composition
    for i in range(m):  # rows of R1
        for j in range(p):  # columns of R2
            min_values = []
            for k in range(n):  # intermediate index
                # Compute min(R1[i, k], R2[k, j])
                min_val = min(R1[i, k], R2[k, j])
                min_values.append(min_val)

            # Compute max of the minimums
            R[i, j] = max(min_values)

    return R

def fuzzy_set_composition(P, R):
    """
    Computes the composition of a fuzzy set P with a fuzzy relation R (P o
R).

    For fuzzy set P (1 x m) and fuzzy relation R (m x p), the resulting
    fuzzy set H' (1 x p) has elements H'[j] = max_i(min(P[i], R[i, j])).
    """
    # Ensure P is treated as a row vector (1 x m)
    P_vector = np.array(P).flatten()
    m = len(P_vector)
    m2, p = R.shape

    # Check for compatibility (length of P must equal rows in R)
    if m != m2:
        raise ValueError("Incompatible dimensions for fuzzy set composition.
Length of P must equal the number of rows in R.")

    # Initialize the result fuzzy set H'
    H_prime = np.zeros(p)

    # Perform the fuzzy set composition
    for j in range(p):  # columns of R (elements of H')
        min_values = []
        for i in range(m):  # elements of P and rows of R
            # Compute min(P[i], R[i, j])
```

```
            min_val = min(P_vector[i], R[i, j])
            min_values.append(min_val)

        # Compute max of the minimums
        H_prime[j] = max(min_values)

    return H_prime

# --- Given Data ---

# Fuzzy relation R1(P, W) (Pollutant to Weather)
R1_PW = np.array([
    [0.9, 0.4, 0.2],
    [0.6, 0.7, 0.5],
    [0.3, 0.5, 0.8]
])

# Fuzzy relation R2(W, H) (Weather to Health)
R2_WH = np.array([
    [0.8, 0.6, 0.4],
    [0.5, 0.7, 0.6],
    [0.3, 0.5, 0.9]
])

# Fuzzy set P (Pollutant level)
P = [0.2, 0.8, 0.6]


# --- Computations ---

## 1. Compute R(P, H) = R1(P, W) o R2(W, H) using max-min composition
R_PH = max_min_composition(R1_PW, R2_WH)

## 2. Find the fuzzy health risk H' = P o R(P, H)
H_prime = fuzzy_set_composition(P, R_PH)


# --- Output Results ---

print("--- Fuzzy Logic Computation ---")

print("\n1. Fuzzy Relation R1(P, W):")
print(R1_PW)

print("\n2. Fuzzy Relation R2(W, H):")
print(R2_WH)

print("\n--- Result 1: R(P, H) = R1(P, W) o R2(W, H) using max-min
composition ---")
# Display the resulting matrix with 3 decimal places for clean formatting
```

	print(np.round(R_PH, 3))
	
	print("\n--- Given Fuzzy Set P: ---")
	print(P)
	
	print("\n--- Result 2: Fuzzy Health Risk H' = P o R(P, H) ---")
	# Display the resulting vector with 4 decimal places
	print(np.round(H_prime, 4))

Result	`--- Fuzzy Logic Computation ---`
	
	`1. Fuzzy Relation R1(P, W):`
	`[[0.9 0.4 0.2]`
	` [0.6 0.7 0.5]`
	` [0.3 0.5 0.8]]`
	
	`2. Fuzzy Relation R2(W, H):`
	`[[0.8 0.6 0.4]`
	` [0.5 0.7 0.6]`
	` [0.3 0.5 0.9]]`
	
	`--- Result 1: R(P, H) = R1(P, W) o R2(W, H) using max-min composition ---`
	`[[0.8 0.6 0.4]`
	` [0.6 0.7 0.6]`
	` [0.5 0.5 0.8]]`
	
	`--- Given Fuzzy Set P: ---`
	`[0.2, 0.8, 0.6]`
	
	`--- Result 2: Fuzzy Health Risk H' = P o R(P, H) ---`
	`[0.6 0.7 0.6]`

Conclusion	In this experiment, I successfully implemented and tested the basic operations of Fuzzy Sets and the Max–Min Composition of Fuzzy Relations using Python and NumPy. Fuzzy Sets enable partial membership, unlike classical (crisp) sets. Their operations—Union, Intersection, and Complement—are defined using the max, min, and 1 - μ functions, respectively. The Max–Min Composition of fuzzy relations extends the idea of matrix multiplication by substituting conventional arithmetic operators with max and min. It plays a vital role in fuzzy logic–based reasoning, particularly within Fuzzy Inference Systems (FIS).