In [1]:
```
# Universe of discourse for temperature
U = [0, 1, 2, 3, 4, 5]
'''
0 = Very Low

1 = Low

2 = Medium

3 = High

4 = Very High

5 = Extreme
'''

# Fuzzy sets A and B on U
# Membership values between 0 and 1
fuzzy_A = {0: 0.3, 1: 0.2, 2: 0.5, 3: 0.7, 4: 1.0, 5: 0.9}
fuzzy_B = {0: 0.1, 1: 0.4, 2: 0.6, 3: 0.8, 4: 0.2, 5: 0.6}
```

In [2]:
```
def union(A, B):
    return {x: max(A[x], B[x]) for x in A}

def intersection(A, B):
    return {x: min(A[x], B[x]) for x in A}

def complement(A):
    return {x: 1 - A[x] for x in A}

def difference(A, B):
    B_complement = complement(B)
    return {x: max(A[x], B_complement[x]) for x in A}

# Perform operations
print("Union:", union(fuzzy_A, fuzzy_B))
print("Intersection:", intersection(fuzzy_A, fuzzy_B))
print("Complement of A:", complement(fuzzy_A))
print("Difference A - B:", difference(fuzzy_A, fuzzy_B))
```

```
Union: {0: 0.3, 1: 0.4, 2: 0.6, 3: 0.8, 4: 1.0, 5: 0.9}
Intersection: {0: 0.1, 1: 0.2, 2: 0.5, 3: 0.7, 4: 0.2, 5: 0.6}
Complement of A: {0: 0.7, 1: 0.8, 2: 0.5, 3: 0.30000000000000004, 4: 0.0,
5: 0.09999999999999998}
Difference A - B: {0: 0.9, 1: 0.6, 2: 0.5, 3: 0.7, 4: 1.0, 5: 0.9}
```

In [3]:
```python
def cartesian_product(A, B):
    return {(a, b): min(A[a], B[b]) for a in A for b in B}

# Cartesian product of A and B
relation_R = cartesian_product(fuzzy_A, fuzzy_B)

# Display a few relation pairs
print("Fuzzy Relation R (A × B):")
for k, v in list(relation_R.items())[:6]:  # show only first 6 for brevity
    print(f"{k}: {v}")
```

```
Fuzzy Relation R (A × B):
(0, 0): 0.1
(0, 1): 0.3
(0, 2): 0.3
(0, 3): 0.3
(0, 4): 0.2
(0, 5): 0.3
```

In [4]:
```
def max_min_composition(R, S, U, V, W):
    T = {}
    for u in U:
        for w in W:
            values = [min(R.get((u, v), 0), S.get((v, w), 0)) for v in V]
            T[(u, w)] = max(values)
    return T

# Create another fuzzy relation S on (V x W)
V = U  # same as U for simplicity
W = [10, 20, 30]
fuzzy_C = {10: 0.3, 20: 0.7, 30: 1.0}

# Relation S: B × C
relation_S = cartesian_product(fuzzy_B, fuzzy_C)

# Max-Min Composition: R (A×B) o S (B×C) => T (A×C)
relation_T = max_min_composition(relation_R, relation_S, U, V, W)

# Display a few composed relation pairs
print("Max-Min Composition T (A × C):")
for k, v in relation_T.items():
    print(f"{k}: {v}")
```

```
Max-Min Composition T (A × C):
(0, 10): 0.3
(0, 20): 0.3
(0, 30): 0.3
(1, 10): 0.2
(1, 20): 0.2
(1, 30): 0.2
(2, 10): 0.3
(2, 20): 0.5
(2, 30): 0.5
(3, 10): 0.3
(3, 20): 0.7
(3, 30): 0.7
(4, 10): 0.3
(4, 20): 0.7
(4, 30): 0.8
(5, 10): 0.3
(5, 20): 0.7
(5, 30): 0.8
```

In [ ]:

In [ ]: