Prerequisites

- sets (operations)
- multisets (basics)

Operations on multisets

Standard set operations

The set operations union, intersection and relative complement can be generalized to multisets.

Definition 1. Given two natural numbers m and n with $m \leq n$, let $\max(m, n) = n$ and $\min(m, n) = m$. Then for any two multisets A_M and B_M

```
- the union $A_M \cup B_M$ maps every $a$ to $\max(A_M(a), B_M(a))$,
- the intersection $A_M \cap B_M$ maps every $a$ to $\min(A_M(a), B_M(a))$
- the relative complement $A_M - B_M$ maps every $a$ to $A_M(a) - B_M(a)$ (or $0$ if the v
```

Example 1 Let $A_M := \{a : 3, b : 2, c : 1\}$ and $B_M := \{a : 1, b : 1, c : 2, d : 1\}$. Then

```
- $A_M \cup B_M = B_M \cup A_M = \setof{a:3, b:2, c:2, d:1}$
- $A_M \cap B_M = B_M \cap A_M = \setof{a:1, b:1, c:1}$
- $A_M - B_M = \setof{a:2, b:1, c:0}$
- $B_M - A_M = \setof{c:1, d:1}$
```

```python
from collections import Counter

A = Counter({"a": 3, "b": 2, "c": 1})
B = Counter({"a": 1, "b": 1, "c": 2, "d": 1})

def multiset_operator(A, B, function):
    keys = set(A.keys()).union(set(B.keys()))
    return Counter({key: function(A.get(key,0), B.get(key,0)) for key in keys})

def multiset_union(A, B):
    return multiset_operator(A, B, max)

def multiset_intersection(A, B):
    return multiset_operator(A, B, min)

print("Union of\n{} and\n{} is\n{}\n".format(A, B, multiset_union(A, B)))
print("Intersection of\n{} and\n{} is\n{}\n".format(A, B, multiset_intersection(A, B)))
print("Relative complement of\n{} and\n{} is\n{}\n".format(A, B, A-B))
print("Relative complement of\n{} and\n{} is\n{}\n".format(B, A, B-A))
```

Exercise 1 Fill each gap with a matching multiset or operator.

- $\{a : 3, b : 2, c : 1\} \cup \{a : 5, b : 1, d : 8\} = \_$
- $\{c : 17\} \_ \{a : 5, b : 1, d : 8\} = \{c : 17\}$
- $\{a : 3, b : 3\} \cup \_ = \{a : 5, b : 3, c : 5, d : 6\}$
- $\_ \_ \{a : 5, b : 1, d : 8\} = \{a : 3, b : 1\}$

Special operations for multisets

Since multisets are a generalization of sets, they allow for certain operations that would not make much sense with sets. These are *multiset sum* ($\uplus$) and *scalar multiplication* ($\otimes$).

Definition 2. Let A_M and B_M be two multisets and n a natural number. Then

```
- the multiset sum $A_M \multisum B_M$ maps every $a$ to $A_M(a) + B_M(a)$,
- the scalar multiplication $n \multimult A_M$ maps every $a$ to $n \mult A_M(a)$ (where $
```

Example 2 As in the previous examples, we will look at the multisets $A_M := \{a : 3, b : 2, c : 1\}$ and $B_M := \{a : 1, b : 1, c : 2, d : 1\}$. For those two sets, $A_M \uplus B_M := \{a : 4, b : 3, c : 3, d : 1\}$, which is identical to $B_M \uplus A_M$. Furthermore, $3 \otimes A_M = \{a : 9, b : 6, c : 3\}$ whereas $3 \otimes B_M = \{a : 3, b : 3, c : 6, d : 3\}$.

```python
def scalar_multiplication(A, n):
    return Counter({key: n * val for key, val in A.items()})

print("{} + {} = {}".format(A, B, A+B))
print("{} * {} = {}".format(3, A, scalar_multiplication(A, 3)))
print("{} * {} = {}".format(3, B, scalar_multiplication(B, 3)))
```

Exercise 2 Calculate the final result of the equations below. If the result cannot be uniquely determined without additional assumptions, explain why.

- $3 \otimes ((\{a : 5, b : 3\}_M \uplus \{a : 1, c : 5, d : 6\}_M) - (\{a : 5, b : 3\}_M \cap \{a : 1, c : 5, d : 6\}_M))$
- $\{\text{John}, \text{John}\} \uplus 3 \otimes \{\text{Mary}, \text{Mary}, \text{Mary}, \text{John}\}_M$
- $\{a : 5, c : 1\}_M \cup \{a : 3, b : 3, c : 3, d : 3\}_M \uplus \{a : 1\}_M$