Aufgabe 7.1

(1) Type

MAP[K,V]

(2) Functions

mt_map: MAP[K,V]

insert: K × V × MAP[K,V] → MAP[K,V]

update: K × V × MAP[K,V] → MAP[K,V]

is_in_dom: K × MAP[K,V] → Bool

lookup: K × MAP[K,V] ↛ V

lookup_opt: K × MAP[K,V] → Option[V]

delete: K × MAP[K,V] → MAP[K,V]

union: MAP[K,V] × MAP[K,V] → MAP[K,V]

size: MAP[K,V] → Nat

(3) Preconditions: ∀ k : K; m : MAP[K,V] •

pre(lookup(k, m)) ⇔ is_in_dom(k, m)

(4) Axioms

∀ k_1, k_2 : k; v_1, v_2 : v; m, m' : MAP[K,V] •

$k_1 = k_2$ ⇒ insert(k_2, v_2, insert(k_1,v_1,m)) = insert(k_2, v_2, m)

k_1 != k_2 ⇒ inser(k_2, v_2, insert(k_1, v_1, m)) = insert(k_2, v_2, insert(k_1, v_1, m))

$k_1 = k_2$ ⇒ update(k_2, v_2, insert(k_1, v_1, m)) = insert(k_1, v_2, m)

update(k_1, v_1, mt_map) = mt_map

k_1 != k_2 ⇒ update(k_2, v_2, insert(k_1, v_1, m)) = insert(k_1, v_2, insert(k_1, v_1, m))

is_in_dom(k_1, mt_map) = false

$k_1 = k_2$ ⇒ is_in_dom(k_2, insert(k_1, v_1, m)) = true

k_1 != k_2 ⇒ is_in_dom(k_2, insert(k_1, v_1, m)) = is_in_dom(k_2, m)

$k_1 = k_2$ ⇒ lookup(k_2, insert(k_1, v_1, m)) = v_1

k_1 != k_2 ⇒ lookup(k_2, insert(k_1, v_1, m)) = lookup(k_2, m)

lookup_opt(k_1, mt_map) = none

k_1 != k_2 ⇒ lookup_opt(k_2, insert(k_1, v_1, m)) = lookup_opt(k_2, m)

$k_1 = k_2$ ⇒ lookup_opt(k_2, insert(k_1, v_1, m)) = some(v_1)

$k_1 = k_2$ ⇒ delete(k_2, insert(k_1, v_1, m)) = m

k_1 != k_2 ⇒ delete(k_2, insert(k_1, v_1, m)) = insert(k_1, v_1, delete(k_2, m))

union(m, mt_map) = m

union(mt_map, m) = m

union(insert(k_1, v_1, m), m') = insert(k_1, v_1, union(m, m'))

size(mt_map) = zero

size(insert(k_1, v_1, m)) = succ(size(m))

Aufgabe 7.2

```
// Map.java
/**
 * Created by liangchun on 05.06.17.
 */
public interface Map<K, V> {
    void insert(K x, V y);
    void update(K x, V y);
    boolean is_in_dom(K x);
    V lookup(K x);
    MyOption<V> lookup_opt(K x);
    void delete(K x);
    void union(Map<K, V> x);
    int size();
}
```

```java
// MyMap.java
import java.util.HashMap;
/**
 * Created by liangchun on 05.06.17.
 */
public class MyMap<K, V> implements Map<K, V> {
    public HashMap<K, V> map;
    MyMap() {
        this.map = new HashMap<>();
    }
    public void insert(K x, V y) {
        if (this.map.get(x) == null)
            this.map.put(x, y);
        else
            this.update(x, y);
    }
    public void update(K x, V y) {
        if (this.map.get(x) != null)
            this.map.put(x, y);
    }
    public V lookup(K x) {
        return this.map.get(x);
    }
    public void delete(K x) {
        if (this.map.get(x) != null)
            this.map.remove(x);
    }
    public void union(Map<K, V> x) {
        MyMap<K, V> temp = (MyMap<K, V>) x;
        temp.map.putAll(this.map);
        this.map = temp.map;
    }
    public boolean is_in_dom(K x) {
        return (this.map.containsKey(x)) ? true : false;
    }
    public int size() {
        return this.map.size();
    }
    public MyOption<V> lookup_opt(K x) {
        if (this.map.containsKey(x))
            return new MyOptionSome<>(this.map.get(x));
        return new MyOptionNone<>();
    }
}
```

```java
/**
 * Created by liangchun on 05.06.17.
 */
public class Main {
    public static void main(String args[]) {
        MyMap<String, Integer> x = new MyMap<>();

        // insert
        x.insert("s", 2);
        System.out.println("Lookup s in x: " + x.lookup("s"));

        // delete
        x.insert("delete", 123);
        x.delete("delete");
        System.out.println("Deleted key-value pair 'delete' from map");

        // lookup
        System.out.println("Lookup 'delete' in x: " + x.lookup("delete"));

        // is_in_dom
        System.out.println("Check if delete is_in_dom in x: " + x.is_in_dom("delete"));

        // update
        x.update("s", 999);
        System.out.println("Update s to 999: " + x.map);

        // union
        MyMap<String, Integer> y = new MyMap<>();
        y.insert("Key", 24);
        y.union(x);
        System.out.println("Union map of y with x: " + y.map);

        // size
        System.out.println("Size of x: " + x.size());
    }
}
```