7️⃣

# Blatt 07

## Aufgabe 1

Hashfunktion : $h(x) = ((x \ll 2)\ xor\ (x + 13))$

Kompressionsfunktion : $c(x) = x \mod 10$

Füge folgende Werte in eine leere Hashtabelle ein:

$$\boxed{3, 2, 62, 0, 11, 9}$$

| Index | Value |
|---|---|
| 0 | valu=9, (36 xor 22) = 50 mod 10 = **0** |
| 1 | |
| 2 | value=11, (44 xor 24) = 52 mod 10 = **2** |
| 3 | value=0, (0 xor 13) = 13 mod 10 = **3** |
| 4 | |
| 5 | |
| 6 | |
| 7 | value=2, (8 xor 15) = 7 mod 10 = **7** |
| 8 | value=3, (12 xor 16) = 28 mod 10 = **8** |
| 9 | value=62, (248 xor 75) = 179 mod 10 = **9** |

## Aufgabe 2

1. Die Funktion $a(x)$ ist nicht geeignet, da z.B die Werte $x = \{0, 1, 2, 3, 4\}$ alle auf den Indize $0$ eingefügt werden, wenn wir für $N = 5$ nehmen. Zu viele Kollisionenm, weshalb diese Funktion sehr ungünstig ist.
2. Auch hier ist das Problem gleich, da zu viele Kollisionen stattfinden und dies ein Zeichen von einer schelchten Hashfunktion ist. Nehmen wir für $x = \{6, 15, 24, 33, 42, 51, ...\}$ für jedes dieser $x$ würde 6 herauskommen.

## Aufgabe 3

-

## Aufgabe 4

```
public static List<WordCountPair> wordsSortedByDistinctCharacters(String string) {
        // the suggested solution uses the WordCountPair class, which is provided to y
ou
        // You do not have to use this class, if you are unhappy with this approach
        List<WordCountPair> wordUniqueCharsCountPairs = new ArrayList<>();
        HashMap<String, Integer> hashMap = new HashMap<>();

        String updatedString = string.replaceAll("[^a-zA-Z0-9 ]", "");
        String[] splittedString = updatedString.split(" ");

        for (int i = 0; i < splittedString.length; i++) {
            hashMap.put(splittedString[i], (int)splittedString[i].chars().distinct().c
ount());
            WordCountPair countPairs = new WordCountPair(splittedString[i], hashMap.ge
t(splittedString[i]));
            wordUniqueCharsCountPairs.add(countPairs);
        }

        Collections.sort(wordUniqueCharsCountPairs);

        /*
        //function without Map
        for (int i = 0; i < splittedString.length; i++) {
            long distinctChars = splittedString[i].chars().distinct().count();
            String word = splittedString[i];
            wordUniqueCharsCountPairs.add(new WordCountPair(word,(int)distinctChars));
        }
        Collections.sort(wordUniqueCharsCountPairs);
         */
        return wordUniqueCharsCountPairs;
    }
```