Aufgabe 8: Hashing

Fügen Sie die folgenden Werte in der gegebenen Reihenfolge in eine Streutabelle der Größe 8 (mit den Indizes 0 bis 7) und der Streufunktion $h(x) = x \mod 8$ ein. Verwenden Sie die jeweils angegebene Hash-Variante bzw. Kollisionsauflösung: 15, 3, 9, 23, 1, 8, 17, 4

(a) **Offenes Hashing**

Zur Kollisionsauflösung wird Verkettung verwendet.

Beispiel

Für die beiden Werte 8 und 16 würde die Lösung wie folgt aussehen:

Bucket	0	1	2	···
Inhalt	8			
	16			

$$
\begin{aligned}
h(15) &= 15 \mod 8 &&= 7\\
h(3) &= 3 \mod 8 &&= 3\\
h(9) &= 9 \mod 8 &&= 1\\
h(23) &= 23 \mod 8 &&= 7\\
h(1) &= 1 \mod 8 &&= 1\\
h(8) &= 8 \mod 8 &&= 0\\
h(17) &= 17 \mod 8 &&= 1\\
h(4) &= 4 \mod 8 &&= 4
\end{aligned}
$$

Bucket	0	1	2	3	4	5	6	7
Inhalt	8	9		3	4			15
		1						23
		17						

(b) **Geschlossenes Hashing**

Zur Kollisionsauflösung wird lineares Sondieren (nur hochzählend) mit Schrittweite $+5$ verwendet.

Treten beim Einfügen Kollisionen auf, dann notieren Sie die Anzahl der Versuche zum Ablegen des Wertes im Subskript (z. B. das Einfügen des Wertes 8 gelingt im 5. Versuch: 8_5).

Beispiel

Für die beiden Werte 8 und 16 würde die Lösung wie folgt aussehen:

Bucket	0	1	2	3	4	5	···
Inhalt	8					16_1	

$h'(x) = x \mod 8$

$h(x,i) = (h'(x) + i \cdot 5) \mod 8$

17 einfügen

Bucket	0	1	2	3	4	5	6	7
Inhalt	8	9		3	23_2		1_2	15

1. Versuch: $h(17,0) = (h'(17) + 0 \cdot 5) \mod 8 = (1+0) \mod 8 = 1 \mod 8 = 1$ (belegt von 9)
2. Versuch: $h(17,1) = (h'(17) + 1 \cdot 5) \mod 8 = (1+5) \mod 8 = 6 \mod 8 = 6$ (belegt von 1)
3. Versuch: $h(17,2) = (h'(17) + 2 \cdot 5) \mod 8 = (1+10) \mod 8 = 11 \mod 8 = 3$ (belegt von 3)
4. Versuch: $h(17,3) = (h'(17) + 3 \cdot 5) \mod 8 = (1+15) \mod 8 = 16 \mod 8 = 0$ (belegt von 8)
5. Versuch: $h(17,4) = (h'(17) + 4 \cdot 5) \mod 8 = (1+20) \mod 8 = 21 \mod 8 = 5$

4 einfügen

Bucket	0	1	2	3	4	5	6	7
Inhalt	8	9		3	23_2	17_5	1_2	15

1. Versuch: $h(4,0) = (h'(4) + 0 \cdot 5) \mod 8 = (4+0) \mod 8 = 4$ (belegt von 23)
2. Versuch: $h(4,1) = (h'(4) + 1 \cdot 5) \mod 8 = (4+5) \mod 8 = 1$ (belegt von 9)
3. Versuch: $h(4,2) = (h'(4) + 2 \cdot 5) \mod 8 = (4+10) \mod 8 = 6$ (belegt von 1)
4. Versuch: $h(4,3) = (h'(4) + 3 \cdot 5) \mod 8 = (4+15) \mod 8 = 3$ (belegt von 3)
5. Versuch: $h(4,4) = (h'(4) + 4 \cdot 5) \mod 8 = (4+20) \mod 8 = 0$ (belegt von 8)
6. Versuch: $h(4,5) = (h'(4) + 5 \cdot 5) \mod 8 = (4+25) \mod 8 = 5$ (belegt von 17)
7. Versuch: $h(4,6) = (h'(4) + 6 \cdot 5) \mod 8 = (4+30) \mod 8 = 2$

Bucket	0	1	2	3	4	5	6	7
Inhalt	8	9	4_7	3	23_2	17_5	1_2	15

(c) Welches Problem tritt auf, wenn zur Kollisionsauflösung lineares Sondieren mit Schrittweite 4 verwendet wird? Warum ist 5 eine bessere Wahl?

Beim linearen Sondieren mit der Schrittweite 4 werden nur zwei verschiedene Buckets erreicht, beispielsweise: 1, 5, 1, 5, etc.

Beim linearen Sondieren mit der Schrittweite 5 werden nacheinander alle möglichen Buckets erreicht, beispielsweise: 1, 6, 3, 0, 5, 2, 7, 4.

Additum Kleines Java-Hilfsprogramm zum Ausrechnen der Sondierungen

```
3  public class Hashing {
4    public static int hashen(int x) {
5      return (x % 8);
6    }
7
```

```
8     public static int sondieren(int x, int i) {
9       int ergebnis = (hashen(x) + i * 5) % 8;
10      System.out.println(String.format("h(%s,%s) = %s", x, i, ergebnis));
11      return ergebnis;
12    }
13
14    public static void main(String[] args) {
15      for (int i = 0; i < 5; i++) {
16        sondieren(17, i);
17      }
18
19      for (int i = 0; i < 7; i++) {
20        sondieren(4, i);
21      }
22    }
23  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/baum/Hashing.java`