1. (3 Punkte) Gegeben sei die Hashfunktion $f(s) =$ (Summe der Unicode-Werte der Zeichen von s)%11. Die Unicode-Werte sind: A = 65, E = 69, M = 77, O = 79, T = 84, V = 86.
   (a) Berechne die Hashwerte der Strings EVA, AVE und TOM.
   (b) Welche Strings haben bei dieser Hashfunktion immer denselben Hashwert? Wie kann man die Hashfunktion verbessern?

Lösung:

(a) EVA und AVE: $69 + 86 + 65 = 220$ und $220\%11 = 0$, beide haben den Hashwert 0.
TOM: $84 + 79 + 77 = 240$ und $240\%11 = 9$, der Hashwert ist 9.

(b) Strings mit denselben Zeichen in anderer Reihenfolge, denn bei der Summe spielt die Reihenfolge keine Rolle. Man multipliziert den bisherigen Wert vor jedem neuen Zeichen mit einer Zahl, z.B. $h = (h \cdot 17 +$ Unicode-Wert)%11. Dann zählt auch die Stelle, an der ein Zeichen steht.

2. (3 Punkte) Gegeben sei die Hashfunktion $f(x) = x\%10$.
   (a) Füge die Zahlenfolge `25 13 35 8 45 18 3` mit offenem Hashing in eine Hashtabelle mit den Plätzen 0 bis 9 ein. Zeichne die Tabelle.
   (b) Welche Zahlen muss man bei der Suche nach der 55 ansehen?

Lösung:

(a) Auf jedem Platz steht eine Liste, leere Plätze sind hier weggelassen:
3: $13 \to 3$
5: $25 \to 35 \to 45$
8: $8 \to 18$

(b) Nur die Liste auf Platz 5, also 25, 35 und 45. Danach steht fest, dass die 55 nicht in der Tabelle ist.

3. (2 Punkte) Gegeben sei die Hashfunktion $f(x) = x\%10$. Füge die Zahlenfolge `12 18 22 23 42 19 38` in die Hashtabelle ein. Benutze dabei lineares Sondieren.

f(x)	0	1	2	3	4	5	6	7	8	9
x										

Lösung:

f(x)	0	1	2	3	4	5	6	7	8	9
x	38		12	22	23	42			18	19

4. (3 Punkte) Gegeben sei die Hashfunktion $f(x) = x\%10$. Die folgende Hashtabelle wurde mit linearem Sondieren erzeugt.

f(x)	0	1	2	3	4	5	6	7	8	9
x	67	19	28					47	57	9

   (a) Welche Plätze werden bei der Suche nach der 28 angesehen?

   (b) Welche Plätze werden bei der Suche nach der 38 angesehen?

   (c) Warum darf man bei der Suche nach der 38 aufhören, bevor man alle Plätze angesehen hat?

Lösung:

(a) 8, 9, 0, 1, 2. Auf Platz 2 steht die 28.

(b) 8, 9, 0, 1, 2, 3. Platz 3 ist leer, die 38 ist also nicht in der Tabelle.

(c) Wäre die 38 eingefügt worden, dann spätestens auf dem ersten freien Platz, den man von Platz 8 aus erreicht, also auf Platz 3.

5. (3 Punkte) Gegeben sei die Hashfunktion $f(x) = x\%10$. Die folgende Hashtabelle wurde mit linearem Sondieren erzeugt.

f(x)	0	1	2	3	4	5	6	7	8	9
x			12	22	32	5				

   (a) Die 22 wird gelöscht, indem ihr Platz einfach geleert wird. Was passiert danach bei der Suche nach der 32?

   (b) Wie macht man es richtig?

Lösung:

(a) Die Suche beginnt auf Platz 2 (dort steht die 12) und trifft dann auf den leeren Platz 3. Dort hört sie auf. Die 32 auf Platz 4 wird nicht gefunden, obwohl sie in der Tabelle steht.

(b) Man markiert den Platz als „gelöscht“. Bei der Suche geht man über gelöschte Plätze hinweg und sondiert weiter. Beim Einfügen darf ein gelöschter Platz wieder belegt werden.

6. (4 Punkte) Eine Hashtabelle ist als Liste gespeichert, freie Plätze enthalten `None`. Die folgende Funktion fügt mit der Hashfunktion $f(x) = x\%N$ und linearem Sondieren ein.

```
def einfuegen(tabelle, x):
    N = len(tabelle)
    for i in range(N):
        p = (x % N + i) % N
        if tabelle[p] is None:
            tabelle[p] = x
            return
```

Schreibe eine Funktion `suchen(tabelle, x)`, die den Platz von `x` zurückgibt oder `None`, wenn `x` nicht in der Tabelle steht.

Lösung:

```
def suchen(tabelle, x):
    N = len(tabelle)
    for i in range(N):
        p = (x % N + i) % N
        if tabelle[p] is None:
            return None
        if tabelle[p] == x:
            return p
    return None
```

7. (3 Punkte) Gegeben sei die Hashfunktion $f(x) = x\%10$. Füge die Zahlenfolge `12 18 22 23 42 19 38` in die Hashtabelle ein. Benutze dabei quadratisches Sondieren.

f(x)	0	1	2	3	4	5	6	7	8	9
x										

Lösung:

f(x)	0	1	2	3	4	5	6	7	8	9
x			12	22	23		42	38	18	19

8. (3 Punkte) Gegeben sei die Hashfunktion $f(x) = x\%10$. Beim quadratischen Sondieren werden von $f(x)$ aus die Plätze $f(x)+0,\ f(x)+1,\ f(x)+4,\ f(x)+9,\ f(x)+16, \ldots$ untersucht, jeweils modulo 10.

   (a) Welche Plätze werden für eine Zahl mit $f(x) = 0$ überhaupt untersucht?

   (b) Gib sechs Zahlen an, nach deren Einfügen die Zahl 30 keinen Platz findet, obwohl noch Plätze frei sind.

Lösung:

(a) Die Quadratzahlen $0, 1, 4, 9, 16, 25, 36, 49, 64, 81$ haben modulo 10 die Reste $0, 1, 4, 9, 6, 5, 6, 9, 4, 1$. Untersucht werden also nur die Plätze 0, 1, 4, 5, 6 und 9, nie die Plätze 2, 3, 7 und 8.

(b) z.B. 10, 11, 14, 15, 16, 19. Sie belegen die Plätze 0, 1, 4, 5, 6 und 9. Die 30 findet keinen Platz, obwohl die Plätze 2, 3, 7 und 8 frei sind.

9. (3 Punkte) Gegeben sei die Hashfunktion $f(x) = x\%10$. Die folgende Hashtabelle wurde mit quadratischem Sondieren erzeugt. Schreibe eine mögliche Reihenfolge des Einfügens auf.

f(x)	0	1	2	3	4	5	6	7	8	9
x	36	20	62	22			42	37	12	

Lösung: Mögliche Reihenfolge: 62, 22, 42, 37, 36, 20, 12

Die 36 kommt erst auf Platz 0, wenn 42 und 37 schon da sind. Die 20 kommt erst nach der 36 auf Platz 1. Die 12 kommt erst auf Platz 8, wenn die Plätze 2, 3, 6 und 1 belegt sind, also nach 62, 22, 42 und 20.

10. (3 Punkte) Gegeben sei die Hashfunktion $f(x) = x\%10$. Die folgende Hashtabelle wurde entweder mit linearem oder mit quadratischem Sondieren erzeugt. Welches Verfahren war es? Begründe und schreibe eine mögliche Reihenfolge des Einfügens auf.

f(x)	0	1	2	3	4	5	6	7	8	9
x	31	11	22		15	41	26			35

Lösung: Quadratisches Sondieren. Beim linearen Sondieren wäre die 15 ab Platz 5 spätestens auf dem freien Platz 7 gelandet und nicht auf Platz 4.

Mögliche Reihenfolge: 11, 22, 41, 31, 26, 35, 15

Die 15 findet die Plätze 5, 6 und $5 + 4 = 9$ belegt und kommt auf $5 + 9 = 14$, also Platz 4.

11. (4 Punkte) Gegeben seien die beiden Hashfunktionen $f_1(x) = x\%10$ und $f_2(x) = (2x + 3)\%10$. Füge die Zahlenfolge `12 18 22 23 42 19 38` in die Hashtabelle ein. Benutze als Sondierungsverfahren Double Hashing.

f(x)	0	1	2	3	4	5	6	7	8	9
x										

Lösung:

f(x)	0	1	2	3	4	5	6	7	8	9
x	19		12	23			42	38	18	22

12. (3 Punkte) Gegeben seien die Hashfunktionen $f_1(x) = x\%10$ und $f_2(x) = (2x+3)\%10$ für Double Hashing.
    (a) Welche Werte kann f_2 annehmen?
    (b) In der Hashtabelle sind die Plätze 1 und 6 belegt, alle anderen sind frei. Die 21 soll eingefügt werden. Was passiert?
    (c) Wie sollte man die Größe N der Hashtabelle wählen, damit das nicht passieren kann?

Lösung:
(a) $2x+3$ ist immer ungerade, also kann f_2 nur die Werte 1, 3, 5, 7 und 9 annehmen.
(b) $f_1(21) = 1$ ist belegt. Die Schrittweite ist $f_2(21) = 45\%10 = 5$. Von Platz 1 aus erreicht man nur die Plätze 6, 1, 6, 1, ... Beide sind belegt, die 21 findet keinen Platz, obwohl acht Plätze frei sind.
(c) N sollte eine Primzahl sein. Dann besucht jede Schrittweite von 1 bis $N-1$ alle Plätze. (Die zweite Hashfunktion darf dabei nie 0 liefern.)

13. (3 Punkte) Eine Hashtabelle mit geschlossenem Hashing hat $N = 1000$ Plätze, davon sind 900 belegt.
    (a) Berechne den Belegungsgrad α.
    (b) Bei Double Hashing braucht man im Mittel ungefähr so viele Sondierungsschritte:

$$\text{erfolglose Suche: } \frac{1}{1-\alpha} \qquad \text{erfolgreiche Suche: } -\frac{\ln(1-\alpha)}{\alpha}$$

    Berechne beide Werte.
    (c) Was sollte man tun, wenn noch mehr Elemente dazukommen?

Lösung:
(a) $\alpha = \frac{900}{1000} = 0.9$
(b) erfolglos: $\frac{1}{1-0.9} = 10$ Schritte, erfolgreich: $-\frac{\ln(0.1)}{0.9} \approx 2.56$ Schritte
(c) Eine größere Hashtabelle anlegen, z.B. ungefähr doppelt so groß, und alle Elemente neu einfügen. Je voller die Tabelle wird, desto stärker steigt die Zahl der Sondierungsschritte.

14. (2 Punkte) In Python kann ein Tupel Schlüssel eines Dictionarys sein, eine Liste nicht:

```
d = {}
d[(1, 2)] = 'a'      # erlaubt
d[[1, 2]] = 'b'      # TypeError: unhashable type: 'list'
```

Erkläre mit dem Hashing, warum Listen nicht als Schlüssel erlaubt sind.

Lösung: Ein Dictionary ist eine Hashtabelle. Ein Schlüssel wird an dem Platz gespeichert, den sein Hashwert angibt. Eine Liste ist veränderbar. Würde sie nach dem Einfügen verändert, hätte sie einen anderen Hashwert und stünde am falschen Platz. Man würde sie dann nicht mehr finden. Tupel sind unveränderbar, ihr Hashwert bleibt gleich.