23	18	25	26	24	17	29	2
+	+	+	+	+	+	+	+
z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8

Aufgabe 1 [2]

Fügen Sie in obiger Tabelle in den leeren Kästchen, vor denen das Pluszeichen steht, die Ziffern Ihrer Matrikelnummer ein. Führen Sie die Additionen durch und ermitteln Sie die Zahlen $\boldsymbol{z_1}$ bis $\boldsymbol{z_8}$.

Aufgabe 2 [18]

Gegeben sind folgende Funktionen (a, b und c seien vordefinierte Konstanten):

```
void g(int n, int i) {
  if (!i) return;
  for (int j=0; j<n-1; j+=2)
    g(n,i-1);
}

void f(int n) {
  if (!n) return;
  for (inti=0; i<a; i++)
    f(n/b);
  g(n,c);
}
```

Setzen Sie für die Konstanten $a = z_6\%3 + 7, b = z_7\%3 + 2, c = z_8\%4$ und berechnen Sie die Laufzeitkomplexität der Funktion $\boldsymbol{f}$ in Θ-Notation.

(Hinweis: Erstellen Sie Rekurrenzgleichungen für die Laufzeiten von $\boldsymbol{g}$ bzw. $\boldsymbol{f}$ und lösen Sie diese mittels fortgesetztem Einsetzen bzw. Master Theorem.)

Aufgabe 3 [20]

Die Werte z_1 bis z_8. (aus Aufgabe 1) seien in dieser Reihenfolge von links nach rechts in einem Array gespeichert. Sortieren Sie die Werte aufsteigend mit

a. [8] Quicksort
b. [4] Mergesort
c. [8] Selection Sort

Aufgabe 4 [20]

a. [9] Fügen Sie die Werte $\boldsymbol{z_2}$ bis $\boldsymbol{z_8}$ aus Aufgabe 1 (in dieser Reihenfolge) in eine zu Beginn leere Hashtabelle der Länge 7 ein. Verwenden Sie als Hashfunktion $\boldsymbol{h(k) = k\%7}$ und double hashing zur Kollisionsbehandlung. Die zweite Hashfunktion ist $\boldsymbol{g(k) = k\%3 + 2.}$
   Skizzieren Sie den Zustand der Hashtabelle nach jedem Einfügeschritt.
b. [1] Löschen Sie den Wert z_4 aus der Tabelle und skizzieren Sie den Zustand der Hashtabelle.
c. [5] Geben Sie den Kollisionspfad (besuchte Indexpositionen) bei einer Suche nach dem Wert $\boldsymbol{z_7}$ an.
d. [5] Geben Sie den Kollisionspfad (besuchte Indexpositionen) bei einer Suche nach dem Wert 42 an.

Aufgabe 5 [20]

a. [8] Fügen Sie die Werte z_1 bis z_8 aus Aufgabe 1 (in dieser Reihenfolge) in einen zu Beginn leeren Heap ein.(Werte können im Heap eventuell mehrfach gespeichert sein.) Skizzieren Sie den Zustand des Heaps nach jedem Einfügeschritt.
b. [4] Geben Sie in C++ ähnlicher Notation die Definition einer möglichst effizienten Datenstruktur für einen Heap an.
c. [4] Geben Sie in C++ ähnlicher Notation eine Definition einer Funktion an, die die Tiefe des Heaps in der Baumdarstellung ermittelt und retourniert.
d. [4] Geben Sie in C++ ähnlicher Notation eine Definition einer Funktion an, die ermittelt, ob es sich um einen Min-Heap oder einen Max-Heap handelt. Die Funktion soll true retournieren, wenn es ein Min-Heap ist, false sonst.

Aufgabe 6 [20]

Gegeben ist die folgende Adjazenzmatrix mit Wegekosten für einen gerichteten Graphen
(die Werte $\boldsymbol{z_1}$ bis $\boldsymbol{z_8}$. sind aus Aufgabe 1 zu übernehmen):

$$\begin{pmatrix} 0 & z6 & 0 & 58 & z4 \\ 2 & 0 & z3 & z8 & 0 \\ z_2 & 0 & 4 & 0 & 7 \\ 0 & 1 & 0 & 0 & z7 \\ 42 & 2 & z5 & 0 & 0 \end{pmatrix}$$

a. [2] Skizzieren Sie den gerichteten Graphen.
b. [10] Bestimmen Sie mit dem Algorithmus von Dijkstra die jeweils kürzesten Wege vom Knoten 1 (erste Zeile, erste Spalte der Matrix) zu allen anderen Knoten des Graphen.
c. [8] Bestimmen Sie mit dem Algorithmus von Kruskal einen minimal spannenden Baum des Schattens des Graphen. (Sie erhalten den Schatten des Graphen, indem Sie die Richtungen der Kanten vernachlässigen. Werden dann zwei Knoten durch zwei Kanten verbunden, so werden diese Kanten zu einer zusammengefasst. Anders ausgedrückt: Zwei Knoten x und y im Schatten sind genau dann durch eine ungerichtete Kante verbunden, wenn im ursprünglich gerichteten Graphen zumindest eine der Kanten von x nach y oder von y nach x existiert. Als Gewicht der ungerichteten Kante wählen sie jeweils das Minimum aller durch sie repräsentierten gerichteten Kanten.)