	12	30	24	11	21	28	19
	+	+	+	+	+	+	+
35							
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 z_2 bis z_8. (z_1 ist bereits mit dem fixen Wert 35 belegt.)

Aufgabe 2 [18]

Schreiben Sie **eine rekursive** Funktion in C++ mit einem Parameter n (vom Typ int), deren Laufzeitkomplexität **gleichzeitig** die Ordnungen $O(n^3)$, $\Omega(n)$ und $\Theta(n^2)$ hat.
Zeigen Sie mithilfe des Mastertheorems, dass die Laufzeitkomplexität Ihrer Funktion die gewünschten Ordnungen hat.

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. [4] Selection Sort
b. [8] Bucketsort
c. [8] Quicksort

Aufgabe 4 [20]

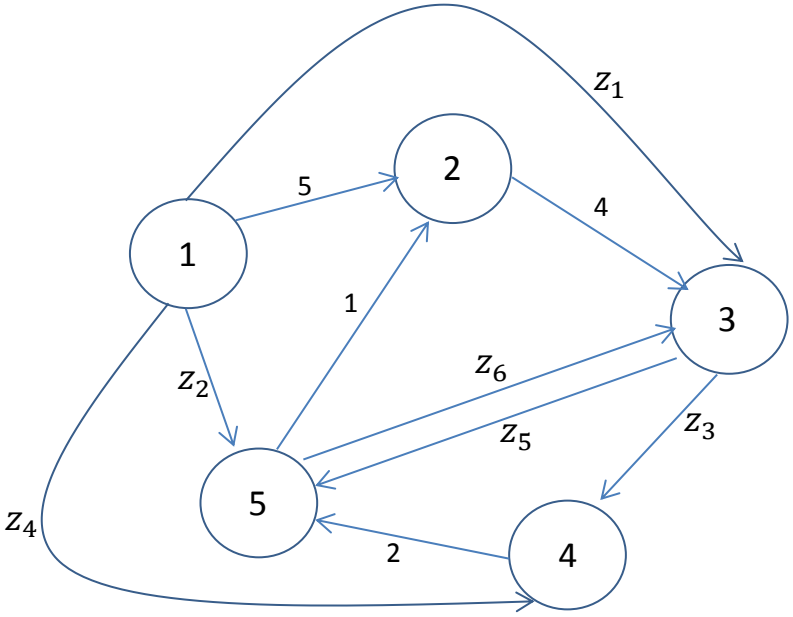
a. [9] Fügen Sie die Werte z_2 bis z_8 aus Aufgabe 1 (in dieser Reihenfolge) in eine zu Beginn leere Hashtabelle der Länge 7 ein. Verwenden Sie als Hashfunktion $h(k) = k\%7$ und double hashing zur Kollisionsbehandlung. Die zweite Hashfunktion ist $g(k) = k\%5 + 1$.
   Skizzieren Sie den Zustand der Hashtabelle nach jedem Einfügeschritt.
   (Anmerkung: Werte können mehrfach in der Tabelle gespeichert werden. Die Tabelle ist statisch, wird also nicht vergrößert!)
b. [1] Löschen Sie den Wert z_3 aus der Tabelle und skizzieren Sie den Zustand der Hashtabelle.
c. [5] Geben Sie den Kollisionspfad (besuchte Indexpositionen) bei einer Suche nach dem Wert z_8 an.
d. [5] Geben Sie den Kollisionspfad (besuchte Indexpositionen) bei einer Suche nach dem Wert 50 an.

Aufgabe 5 [20]

a. [8] Fügen Sie die Werte z_2 bis z_8 aus Aufgabe 1 (in dieser Reihenfolge) in einen zu Beginn leeren Heap ein. Skizzieren Sie den Zustand des Baums nach jedem Einfügeschritt.
   (Anmerkung: Werte können mehrfach im Baum gespeichert werden.)
b. [4] Geben Sie in C++ ähnlicher Notation die Definition einer Datenstruktur für einen Heap an.
c. [4] Geben Sie in C++ ähnlicher Notation eine Definition einer Funktion an, die den Heap breadth first traversiert und alle gespeicherten Werte ausgibt.
d. [4] Bestimmen Sie die Laufzeitkomplexität Ihrer Traversierungsfunktion abhängig von der Anzahl n der im Suchbaum gespeicherten Werte in Θ-Notation.

Aufgabe 6 [20]

Gegeben ist der folgende gerichtete Graph
(die Werte z_1 bis z_6. sind aus Aufgabe 1 zu übernehmen):



a. [3] Geben Sie die Adjazenzmatrix des Graphen an.
b. [3] Skizzieren Sie die Adjazenzliste des Graphen.
c. [10] Bestimmen Sie mit dem Algorithmus von Dijkstra die jeweils kürzesten Wege vom Knoten 1 zu allen anderen Knoten des Graphen.
d. [4] Ist für den Dijkstra-Algorithmus eher die Verwendung einer Adjazenzmatrix oder einer Adjazenzliste vorteilhaft? Begründen Sie Ihre Aussage.