23	**17**	**39**	**24**	**32**	**26**	**33**	**28**
+	**+**	**+**	**+**	**+**	**+**	**+**	**+**
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_1 bis z_8.

Aufgabe 2 [18]

a. [9] Schreiben Sie **eine** 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.

b. [9] Fügen Sie in nachfolgender Tabelle Kreuze an den Positionen ein, wo die in der Zeile angeführte Funktion von der in der Spalte angegebenen Ordnung ist.

$f(n)$	$O(n)$	$O(n^3)$	$O(\log n)$	$O(\log^2 n)$	$\Omega(n^2)$	$\Omega(\log n)$	$\Omega(\log^3 n)$	$\Theta(n^2)$	$\Theta(\log n)$	$\Theta(\log^3 n)$
n										
n^2										
n^3										
$\log n$										
$\log^2 n$										
$\log^3 n$										

Anmerkung: $\log^2 n = (\log n)(\log n)$

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] Selection Sort
c. [8] Heap Sort

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.

b. [1] Löschen Sie den Wert z_5 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_3 an.

d. [5] Geben Sie den Kollisionspfad (besuchte Indexpositionen) bei einer Suche nach dem Wert 52 an.

Aufgabe 5 [20]

a. [4] Fügen Sie die Werte z_2 bis z_8 aus Aufgabe 1 (in dieser Reihenfolge) in einen zu Beginn leeren binären Suchbaum ein. Skizzieren Sie den Zustand des Baums nach jedem Einfügeschritt.
(Anmerkung: Werte können mehrfach im Baum gespeichert werden.)

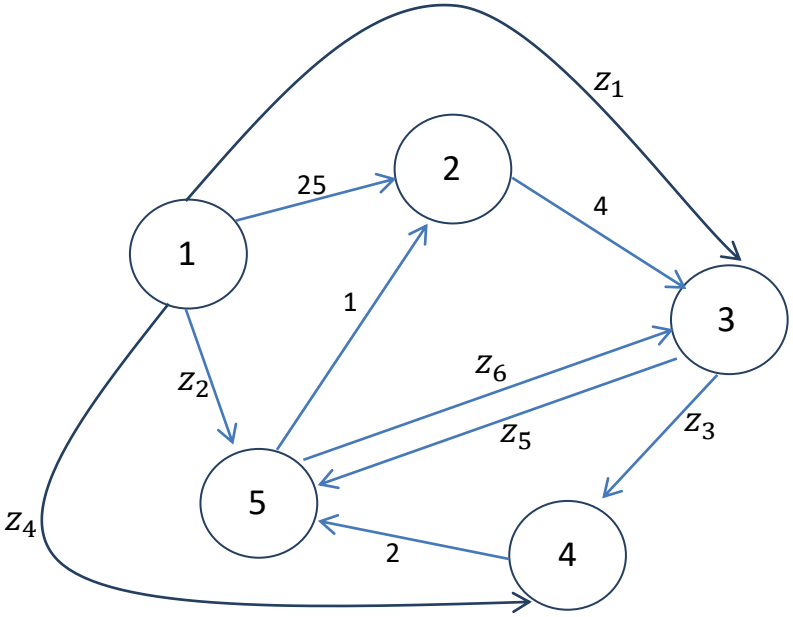
b. [4] Geben Sie in C++ ähnlicher Notation die Definition einer möglichen Datenstruktur für einen binären Suchbaum an.

c. [8] Geben Sie in C++ ähnlicher Notation eine Definition einer Funktion oder Methode an, die das Maximum der im binären Suchbaum gespeicherten Werte ermittelt.

d. [4] Bestimmen Sie die Laufzeitkomplexität Ihrer Funktion abhängig von der Anzahl n der im Suchbaum gespeicherten Werte in O-Notation. Begründen Sie Ihr Ergebnis kurz.

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. [12] 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 oder mehr 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.) Notieren Sie alle Zwischenschritte so genau, dass klar ist, wann welche Kante zum spannenden Baum hinzugefügt wird.

d. [2] Ist der oben dargestellte Graph topologisch sortierbar? Falls ja, geben Sie eine topologische Sortierung an, andernfalls begründen Sie, warum eine solche nicht gefunden werden kann.