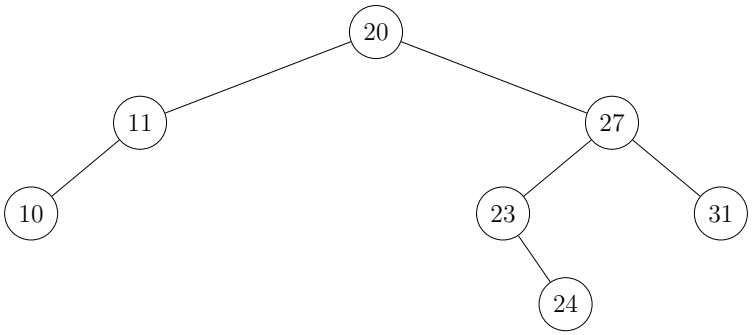
Vorname	Name	Matr.-Nr.

Aufgabe 1 Bäume (3+1+6=10 Punkte)

a) Gegeben sei der folgende binäre Suchbaum:



(i) Geben Sie den binären Suchbaum an, der durch *Rechtsrotation auf den Wurzelknoten* des oben gegebenen Baumes entsteht:

(ii) Geben Sie den binären Suchbaum an, der durch *Linksrotation auf den Wurzelknoten* des oben gegebenen Baumes (*nicht* des resultierenden Baums aus *(i)*) entsteht:

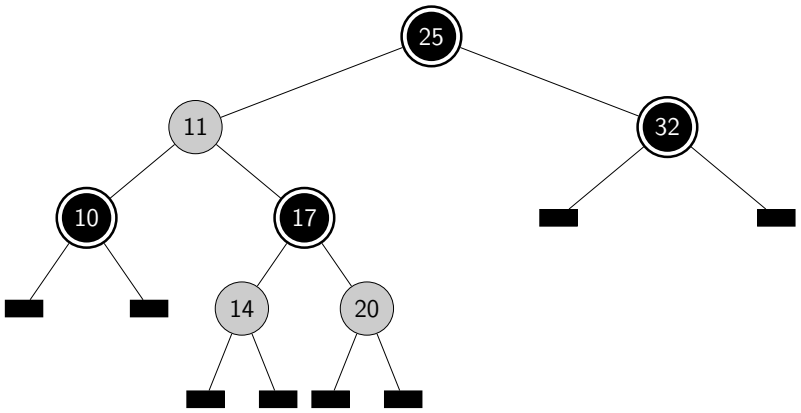
Vorname	Name	Matr.-Nr.

b) Bleibt ein Rot-Schwarz-Baum nach einer Linksrotation um einen inneren Knoten in jedem Fall ein Rot-Schwarz-Baum? Begründen Sie Ihre Antwort.

Vorname	Name	Matr.-Nr.

c) Fügen Sie den Wert 18 in den folgenden Rot-Schwarz-Baum ein. Zeichnen Sie den Baum nach dem Einfügen, vor und nach jeder Rotation, sowie das Endergebnis.

In Ihrer Lösung müssen Sie externe Blätter nicht einzeichnen.



Vorname	Name	Matr.-Nr.

Aufgabe 2 Hashing (2+4+4 = 10 Punkte)

Gegeben sei die folgende Folge von Schlüsselwerten:

5, 19, 22, 14, 17, 32, 30, 43, 1,

sowie die Hashfunktion:

$$h_1(k) = k \bmod 13$$

a) Geben Sie die Hashtabelle an, die entsteht, wenn die gegebenen Schlüssel, unter Verwendung der Hashfunktion h_1, sukzessive in eine zu Beginn leere *Hashtabelle mit Verkettung (geschlossene Adressierung)* der Länge 13 einfügt werden.

0	1	2	3	4	5	6	7	8	9	10	11	12

Vorname	Name	Matr.-Nr.

5

b) Geben Sie die Hashtabelle an, die entsteht, wenn die gegebenen Schlüssel, unter Verwendung von *quadratischer Sondierung* mit $c_1 = c_2 = 2$, sukzessive in eine zu Beginn leere *Hashtabelle mit offener Adressierung* der Länge 13 einfügt werden.

Nutzen Sie für jeden einzufügenden Schlüssel eine Zeile der folgenden Tabelle und markieren Sie fehlgeschlagene Sondierungspositonen mit einem $\times$, sowie die erfolgreiche Sondierungsposition mit dem entsprechenden Schlüssel. Eingefügte Schlüssel müssen in den folgenden Zeilen nicht wiederholt werden.

Schlüssel: 5, 19, 22, 14, 17, 32, 30, 43, 1

0	1	2	3	4	5	6	7	8	9	10	11	12

Vorname	Name	Matr.-Nr.

6

c) Geben Sie die Hashtabelle an, die entsteht, wenn die gegebenen Schlüssel, unter Verwendung von doppeltem Hashing mit h_1 und $h_2(k) = 1 + (k \bmod 5)$, sukzessive in eine zu Beginn leere *Hashtabelle mit offener Adressierung* der Länge 13 einfügt werden.

Nutzen Sie für jeden einzufügenden Schlüssel eine Zeile der folgenden Tabelle und markieren Sie fehlgeschlagene Sondierungspositonen mit einem $\times$, sowie die erfolgreiche Sondierungsposition mit dem entsprechenden Schlüssel. Eingefügte Schlüssel müssen in den folgenden Zeilen nicht wiederholt werden.

Schlüssel: 5, 19, 22, 14, 17, 32, 30, 43, 1

0	1	2	3	4	5	6	7	8	9	10	11	12

Vorname	Name	Matr.-Nr.

Aufgabe 3 Sortieren (4+2+4=10 Punkte)

Quicksort benötigt einen Partitionierungsalgorithmus um die zu sortierende Eingabe schrittweise aufzuteilen. In der Vorlesung haben wir bereits einen Partionierungsalgorithmus kennengelernt. Im Folgenden finden Sie einen weiteren – den von Hoare ursprünglich vorgeschlagenen – Partitionierungsalgorithmus sowie den dazugehörigen Quicksortalgorithmus:

```
1  void QuickSort(int A[], int left, int right){
2
3      if(left < right){
4          int p = Hoare-Partitionierung(A, left, right);
5          QuickSort(A, left, p);
6          QuickSort(A, p+1, right);
7      }
8  }
```

```
1  int Hoare-Partitionierung(int A[], int l, int r){
2      int pivot = A[l];
3      l--;
4      r++;
5
6      while (true){
7
8          r--;
9          while(A[r] > pivot) r--;
10
11         l++;
12         while(A[l] < pivot) l++;
13
14         if (l >= r)
15             return r;
16
17         swap(A[l], A[r]);
18
19         // Ausgabe
20     }
21 }
```

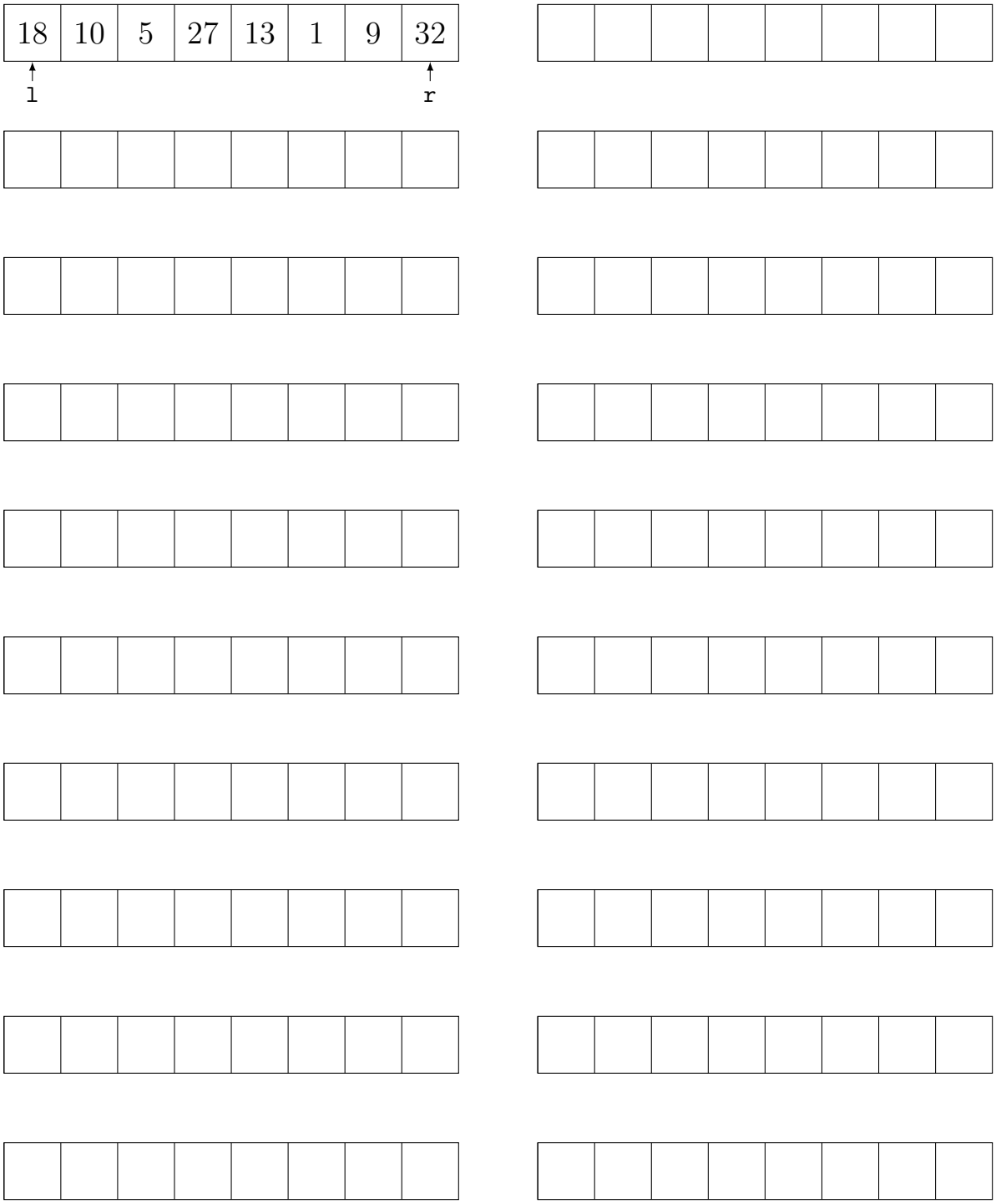
a) Sortieren Sie die folgende Eingabe entsprechend dem oben angegebenen Quicksortalgorithmus beim Aufruf von `QuickSort(A, 0, 7)`. Geben Sie jeweils beim Erreichen der Zeile 19 (*// Ausgabe*) den Inhalt des Arrays sowie die Positionen von `l` und `r` an:

A:	18	10	5	27	13	1	9	32

Auf der folgenden Seite finden Sie obigen Hoare-Partitionierungsalgorithmus in kompakterer Darstellung, eine Kopie des Eingabearrays sowie mehrere Vorlagen für weitere Arrays, die Sie für Ihre Lösung nutzen können. Sie dürfen beliebig viele Zwischenschritte angeben, sollten dann aber die Arrays, die der gewünschten *Ausgabe* entsprechen, besonders kennzeichnen.

Vorname	Name	Matr.-Nr.


```
1  int Hoare-Partitionierung(int A[], int l, int r){
2      int pivot = A[l]; l--; r++;
3      while (true){
4          r--; while(A[r] > pivot) r--;
5          l++; while(A[l] < pivot) l++;
6          if (l >= r) return r;
7          swap(A[l], A[r]);
8          // Ausgabe
9  }   }
```



b) Ist Quicksort mit Hoare-Partitionierung stabil? Begründen Sie Ihre Antwort.

Vorname	Name	Matr.-Nr.

Aufgabe 4 Rekursionsgleichungen (1+2+4+3=10 Punkte)

Betrachten Sie das folgende Programm:

```
1  public static double calcu(double n){
2      if (n <= 1)
3          return 0;
4      double number = calcu(n/4)*calcu(n/4)*calcu(n/4);
5      for (int i = 0 ; i < n; i++){
6          for (int j = 0; j*j < n; j++){
7              number = number + 1;
8          }
9      }
10     return number;
11 }
```

a) Geben Sie eine Rekursionsgleichung für die *asymptotische* Laufzeit des obigen Programms an.

Gehen Sie davon aus, dass die Grundrechenarten `+`, `-`, `*`, `/`, sowie Zuweisungen `=` und Vergleiche `<=` in konstanter Zeit $O(1)$ ausgeführt werden.

b) Lösen Sie die Rekursionsgleichung aus *a)* mit Hilfe des Mastertheorems.

Vorname	Name	Matr.-Nr.

c) Bestimmen Sie die Lösung von $T(n) = 2T(\frac{n}{2}) + n \log n$, $T(1) = 1$ mit $n = 2^k, k \in \mathbb{N}$ anhand des entsprechenden Rekursionsbaums. Das Ergebnis sollte frei von Summenzeichen sein. Sie brauchen Ihre Lösung *nicht* zu beweisen.

d) Versuchen Sie $T(n)$ mit Hilfe des Mastertheorems zu lösen. Begründen Sie Ihre Antwort.

Vorname	Name	Matr.-Nr.

Aufgabe 5 Algorithmen und Laufzeit (5+5=10 Punkte)

a) Gegeben sei ein Integer-Array der Länge n mit beliebigen Werten. Finden Sie einen Algorithmus, der die Länge der längsten zusammenhängenden Folge von Nullen innerhalb des Arrays in Linearzeit ($O(n)$) bestimmt.

Beispiel: Die Länge der längsten zusammenhängenden Folge von Nullen in dem Array $[0, 0, 12, -1, 4, 0, 0, 0, 5, 0]$ ist drei.

Vorname	Name	Matr.-Nr.

b) Gegeben sei der folgende Algorithmus `divInter(int n, int k)`. Bestimmen Sie die exakte Laufzeit $W(n, k)$ für den Aufruf `divInter (n,k)` in Abhängigkeit von n und k.

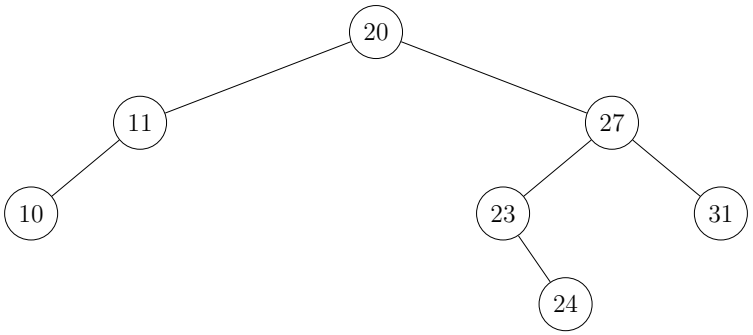
```
1  (int, int) divInter (int n, int k){
2      int q=0;
3      int r=n;
4      int m=2*k;
5      while (r>=m){
6          q=q+1;
7          r=r-m;
8      }
9      q=q*2;
10     return (q,r);
11 }
```

Gehen Sie davon aus, dass die Grundrechenarten `+`, `-`, `*`, `/`, sowie Zuweisungen `=` und Vergleiche `>=` in exakt einer Zeiteinheit ausgeführt werden. Die Kosten der weiteren Anweisungen sollen ignoriert werden.

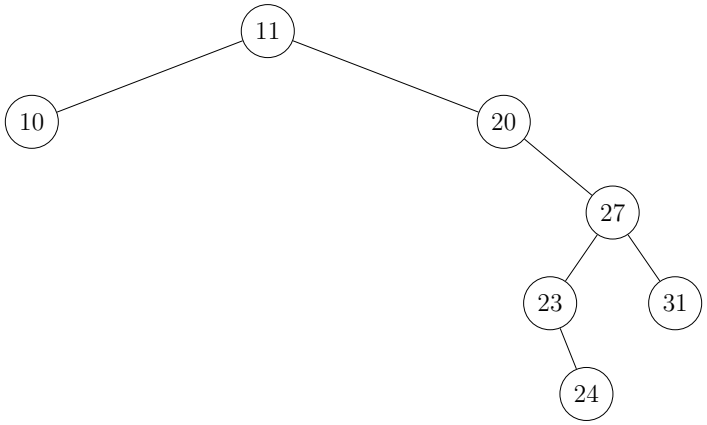
Vorname	Name	Matr.-Nr.

Aufgabe 1 Bäume (3+1+6=10 Punkte)

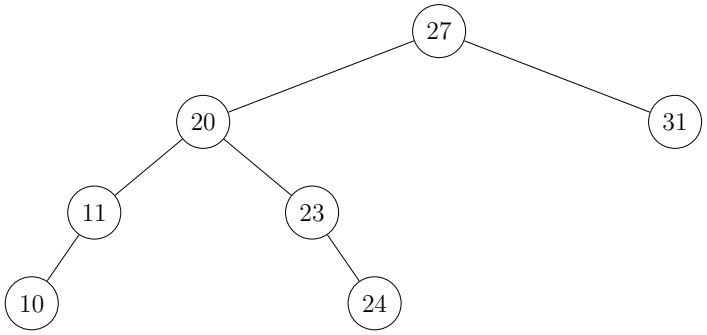
a) Gegeben sei der folgende binäre Suchbaum:



(i) Geben Sie den binären Suchbaum an, der durch *Rechtsrotation auf den Wurzelknoten* des oben gegebenen Baumes entsteht:
Lösung:



(ii) Geben Sie den binären Suchbaum an, der durch *Linksrotation auf den Wurzelknoten* des oben gegebenen Baumes (*nicht* des resultierenden Baums aus *(i)*) entsteht:
Lösung:

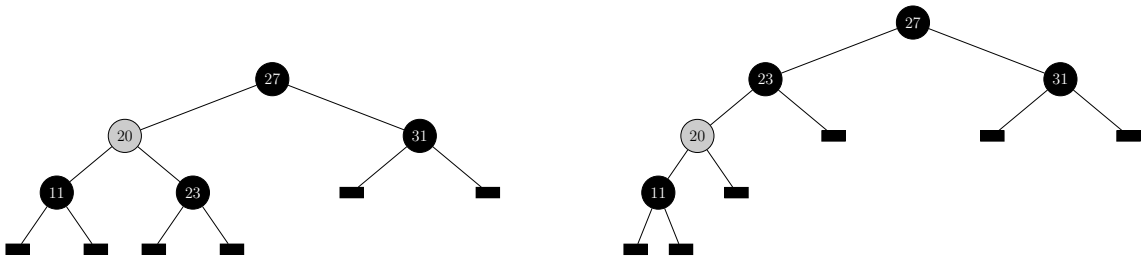


Vorname	Name	Matr.-Nr.

b) Ist ein Rot-Schwarz-Baum nach einer Linksrotation, um einen inneren Knoten, immer noch ein Rot-Schwarz-Baum? Begründen Sie Ihre Antwort.

Lösung: Nein.

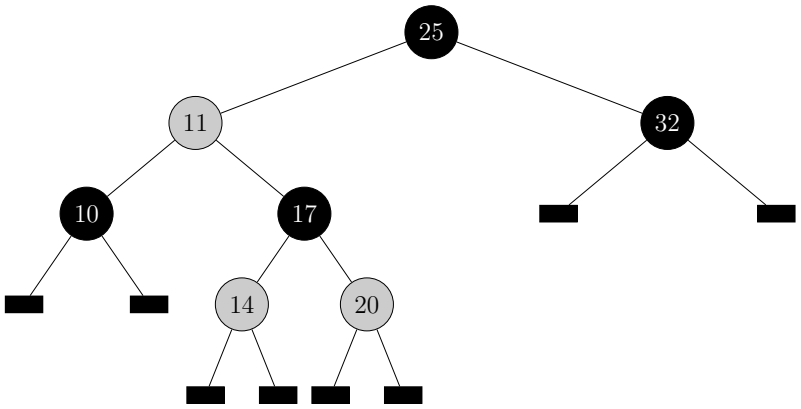
Gegenbeispiel: Linksrotation um den roten Knoten 20 führt zu einem Baum mit unterschiedlicher Schwarzhöhe:



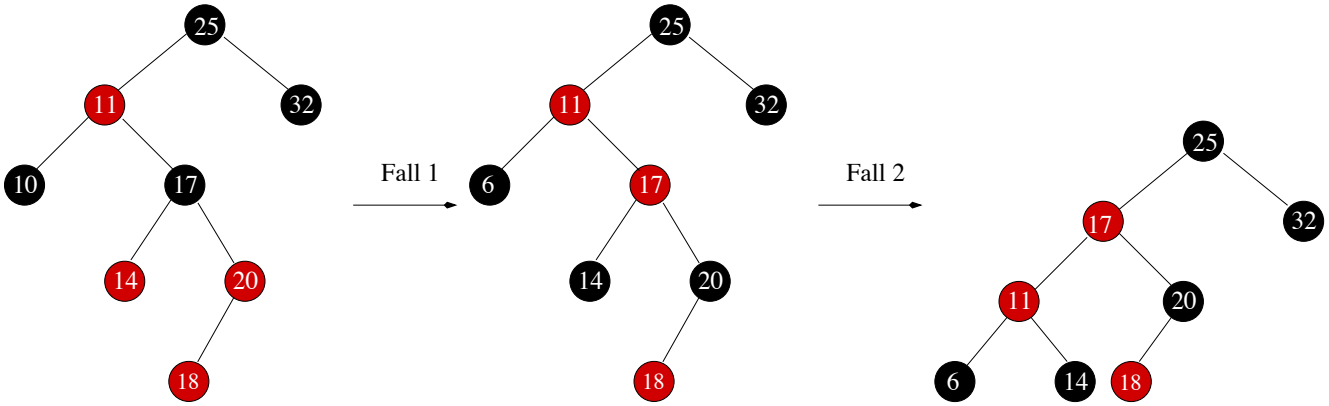
Vorname	Name	Matr.-Nr.

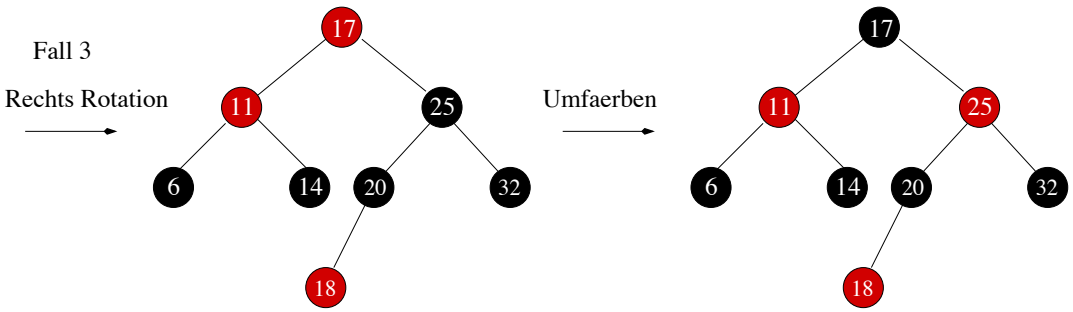
c) Fügen Sie den Wert 18 in den folgenden Rot-Schwarz-Baum ein. Zeichnen Sie den Baum nach dem Einfügen, sowie vor und nach jeder Rotation.

In Ihre Lösung müssen Sie externen Blätter nicht einzeichnen.



Lösung:





Vorname	Name	Matr.-Nr.

Aufgabe 2 Hashing (2+4+4 = 10 Punkte)

Gegeben sei die folgende Folge von Schlüsselwerten:

5, 19, 22, 14, 17, 32, 30, 43, 1,

sowie die Hashfunktion:

$$h_1(k) = k \bmod 13$$

a) Geben Sie die Hashtabelle an, die entsteht, wenn die gegebenen Schlüssel, unter Verwendung der Hashfunktion h_1, sukzessive in eine zu Beginn leere *Hashtabelle mit Verkettung (geschlossener Adressierung)*, der Länge 13 einfügt werden.

Lösung:

0	1	2	3	4	5	6	7	8	9	10	11	12
	14 1			17 30 43	5	19 32			22			

Vorname	Name	Matr.-Nr.

5

b) Geben Sie die Hashtabelle an, die entsteht, wenn die gegebenen Schlüssel, unter Verwendung von *quadratischer Sondierung* mit $c_1 = c_2 = 2$, sukzessive in eine zu Beginn leere *Hashtabelle mit offener Adressierung* der Länge 13 einfügt werden.

Nutzen Sie für jeden einzufügenden Schlüssel eine Zeile der folgenden Tabelle und markieren Sie fehlgeschlagene Sondierungspositonen mit einem $\times$, sowie die erfolgreiche Sondierungsposition mit dem entsprechenden Schlüssel. Eingefügte Schlüssel müssen in den folgenden Zeilen nicht wiederholt werden.

Schlüssel: 5, 19, 22, 14, 17, 32, 30, 43, 1

Lösung: das heißt: $h(k, i) = (h_1(k) + 2i + 2i^2) \bmod 13$

0	1	2	3	4	5	6	7	8	9	10	11	12
					5							
						19						
									22			
	14											
				17								
						$\times$				32		
				$\times$				30				
			43	$\times$				$\times$				
1	$\times$				$\times$							

Vorname	Name	Matr.-Nr.

6

c) Geben Sie die Hashtabelle an, die entsteht, wenn die gegebenen Schlüssel, unter Verwendung von doppeltem Hashing mit h_1 und $h_2(k) = 1 + (k \bmod 5)$, sukzessive in eine zu Beginn leere *Hashtabelle mit offener Adressierung* der Länge 13 einfügt werden.

Nutzen Sie für jeden einzufügenden Schlüssel eine Zeile der folgenden Tabelle und markieren Sie fehlgeschlagene Sondierungspositonen mit einem $\times$, sowie die erfolgreiche Sondierungsposition mit dem entsprechenden Schlüssel. Eingefügte Schlüssel müssen in den folgenden Zeilen nicht wiederholt werden.

Schlüssel: 5, 19, 22, 14, 17, 32, 30, 43, 1

Lösung: das heißt: $h(k, i) = (h_1(k) + i \cdot h_2(k)) \bmod 13$

0	1	2	3	4	5	6	7	8	9	10	11	12
					5							
						19						
									22			
	14											
				17								
						$\times$			$\times$			32
				$\times$	$\times$	$\times$	30					
				$\times$				43				
	$\times$		1									

Vorname	Name	Matr.-Nr.

Aufgabe 3 Sortieren (4+2+4=10 Punkte)

Quicksort benötigt einen Partitionierungsalgorithmus um die zu sortierenden Eingabe schrittweise aufzuteilen. In der Vorlesung haben wir bereits einen Partionierungsalgorithmus kennengelernt. Im Folgenden finden Sie einen weiteren - den von Hoare ursprünglich vorgeschlagenen - Partionierungsalgorithmus sowie den dazugehörigen Quicksort Algorithmus:

```
1  void QuickSort(int A[], int left, int right){
2
3      if(left < right){
4          int p = Hoare-Partitionierung(A, left, right);
5          QuickSort(A, left, p);
6          QuickSort(A, p+1, right);
7      }
8  }
```

```
1  int Hoare-Partitionierung(int A[], int l, int r){
2      int pivot = A[l];
3      l--;
4      r++;
5
6      while (true){
7
8          r--;
9          while(A[r] > pivot) r--;
10
11         l++;
12         while(A[l] < pivot) l++;
13
14         if (l >= r)
15             return r;
16
17         swap(A[l], A[r]);
18
19         // Ausgabe
20     }
21 }
```

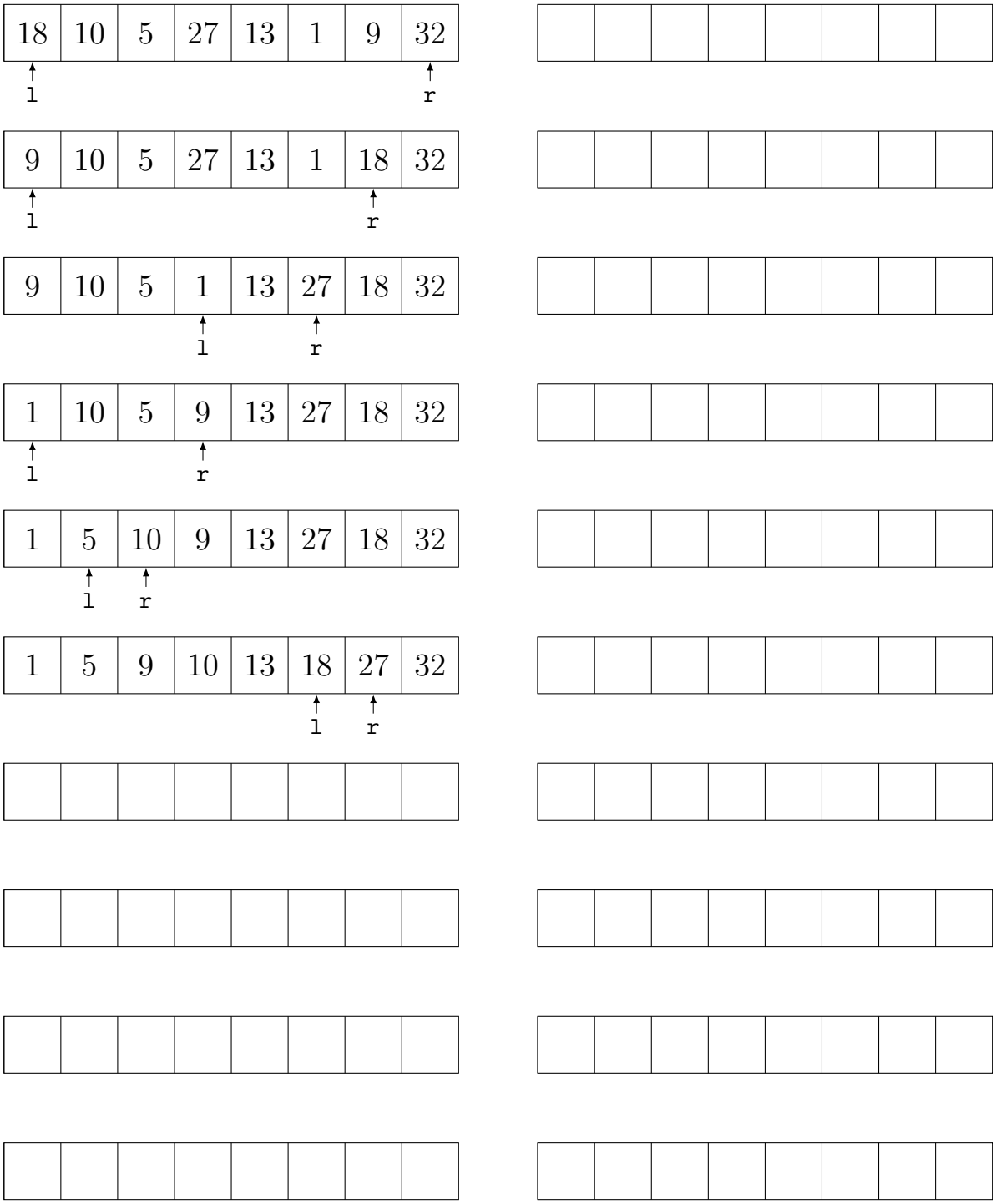
a) Sortieren Sie die folgende Eingabe entsprechend dem oben angegebenen Quicksort-Algorithmus. Geben Sie, für jedes Durchlaufen der Zeile 19 `// Ausgabe`, den Inhalt des Arrays sowie die Positionen von `l` und `r` an:

18	10	5	27	13	1	9	32

Auf der folgenden Seite finden Sie den identischen Hoare-Partionierungs Algorithmus in kompakterer Darstellung, eine Kopie des Eingabearrays sowie mehrere Vorlagen für weiter Arrays, die Sie für Ihre Lösung nutzen können. Sie dürfen beliebig viele Zwischenschritte angeben, sollten dann aber die Arrays, die der gewünschten *Ausgabe* entsprechen, besonders markieren.

Vorname	Name	Matr.-Nr.


```
1  int Hoare-Partitionierung(int A[], int l, int r){
2      int pivot = A[l]; l--; r++;
3      while (true){
4          r--; while(A[r] > pivot) r--;
5          l++; while(A[l] < pivot) l++;
6          if (l >= r) return r;
7          swap(A[l], A[r]);
8          // Ausgabe
9  }   }
```



Vorname	Name	Matr.-Nr.

b) Ist Quicksort mit Hoare-Partitionierung stabil? Begründen Sie Ihre Antwort.

Lösung: Nein! Betrachten wir zum Beispiel das folgende, minimale Gegenbeispiel:

$$[1, 1'] \text{ wird sortiert zu: } [1', 1]$$

c) Beweisen Sie induktiv, dass ein ternärer Baum (jeder Knoten besitzt bis zu drei Kinder) mit Höhe h maximal $\frac{3^{h+1}-1}{2}$ Knoten besitzt.

Lösung:

Behauptung: Ein ternärer Baum hat maximal 3^k Knoten auf Ebene k. Somit hat ein ternärer Baum mit Höhe h maximal $\sum_{k=0}^{h} 3^k = \frac{3^{h+1}-1}{2}$.

Man kann per Induktion zeigen, dass ein ternärer Baum hat maximal 3^k Knoten in Ebene k hat:

•Induktionsanfang: $k = 0$

$$2^0 = 1$$

•Induktionsvorraussetzung: Die Aussage gilt für k.

• Für Ebene $k + 1$:

Wir erzeugen einen maximalen Baum der Höhe $k+1$, indem wir an einen neuen Wurzelknoten drei maximale Bäume der Höhe k anhängen. Die Anzahl der Knoten in diesem Baum ist:

$$\begin{aligned} & 1 + 3 \cdot \frac{3^{h+1} - 1}{2} \\ = \quad & \frac{3^{h+2} - 3}{2} + 1 \\ = \quad & \frac{3^{h+2}}{2} - \frac{3}{2} + 1 \\ = \quad & \frac{3^{h+2}}{2} - \frac{1}{2} \\ = \quad & \frac{3^{h+2} - 1}{2} \end{aligned}$$