Introprog Gedächtnisprotokoll vom 21.02.18

~~geben~~ Gegeben sei das Array [6, 3, 12, 5, 4]

Kreuzen Sie an, was zutrifft:

- Nach einer Ausführung der inneren Schleife von BubbleSort sieht das Array so aus: [3, 6, 12, 4, 5]
- Nach einer Ausführung der inneren Schleife von InsertionSort sieht das Array so aus: [3, 6, 12, 5, 4]
- Nach einer Ausführung der inneren Schleife von Selection Sort sieht das Array so aus [4, 3, 12, 5, 6] ↳ (*) (siehe letzte Seite!)

Fügen Sie die Zahlen 5, 2, 10, 8, 9 in einen Binärbaum ein.

Welche Laufzeit hat der ~~Bin~~ Binärbaum im Worst Case? O( )

Tragen Sie in die folgende Tabelle die Worst-Case-Komplexität ein:

Liste:	
Einfügen am Anfang	
Suchen	
Nachfolger eines Elements bestimmen	
Vorgänger eines Elements bestimmen	

Doppelt verkettete Liste:	
Einfügen am Anfang	
Suchen	
Nachfolger eines Elements bestimmen	

Vorgänger eines Elements bestimmen

Min-Heap

Extrahieren des Minimums

Suchen

```
FirstIsMin (array A, array_len)
  resultat ← 1
  while (i <= len) do
      if A[1] > A[i] resultat ← 0
      i++;
  return resultat.
```

Stellen Sie die Korrektheitsbedingung für FirstIsMin auf, indem Sie aus den unten genannten Möglichkeiten genau eine Quantifizierung und eine Aussage auswählen:

Quantifizierung: ☐ Aussage: ☐

$\forall k \in \{1 \ldots\}$	. . .
< Liste mit Quantifizierungen und Aussagen >	

Stellen Sie die Schleifeninvariante auf, indem Sie aus den unten genannten Möglichkeiten genau eine Quantifizierung und eine Aussage auswählen.

Zeigen Sie, dass die Invariante gilt, und dass sich dadurch die Korrektheitsaussage des Algorithmus ableiten lässt.

Quantifizierung Aussage

☐ ☐

< Liste mit	Quantifizierungen
und	Aussagen >

Fibonacci

```
int fib(int n)
    return(fib(n-1) + fib(n-2))
```

Welche Komplexität hat fib(n)?

☐ $O(n)$ ☐ $O(n^2)$ ☐ $O(2^n)$

Wie könnte man die Funktion schneller machen (Hinweis: Denken Sie iterativ)

Implementieren Sie den Algorithmus in C. Benutzen Sie ein dynamisch alloziertes Hilfsarray.

```
fib(int n) {



}
```

Welche Komplexität hat dieser Algorithmus?

```
int mod(int a, b) {
remainder_new = 0;



return remainder;
}
```

Die Funktion soll a mod b zurückgeben. Implementieren Sie diese Funktion in C. Verwenden Sie dabei weder die modulo-Funktion noch einen Divisionsalgorithmus.

Sei jetzt a=14 und b=3.

Füllen Sie die untere Tabelle aus zum Anfangszustand und nach jeder Iteration. Falls eine Variable noch keinen Wert zugewiesen bekommen hat, schreiben Sie N/A

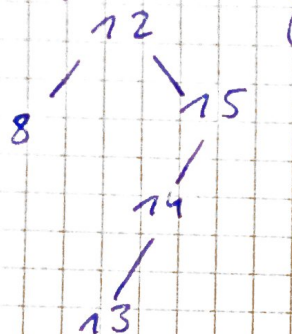
remainder	remainder_new

Implementieren Sie den Algorithmus Treshold, der hier in Pseudocode gegeben ist, in C. Füllen Sie anschließend die unten gegebene Tabelle mit den Variablenbelegungen zum Anfang und nach jeder Iteration aus.

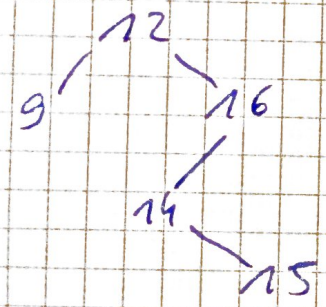
(hier war Pseudocode gegeben)

Rotieren Sie die folgenden Binäre-AVL-Bäume entsprechend der Balance-Funktion. Geben Sie die jeweilige Rotation und den ~~auf~~ entsprechenden Knoten, um den rotiert wird, in der untenstehenden Tabelle an und zeichnen Sie den Baum nach jeder Rotation.

(Werte können hier von denen in der Klausur abweichen)



~~Links/Rechts~~	1. Rotation	2. Rotation
Links/rechts		
Knoten		

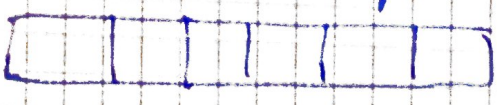


	1. Rotation	2. Rotation
Links/rechts		
Knoten		

Welche Eigenschaft kennzeichnet einen AVL-Baum im Gegensatz zum Binärbaum?

Fügen Sie die folgenden Zahlen entsprechend der Heap-Insert-Prok. aus der Vorlesung in einen min-Heap ein: 7, 12, 5, 4, 1, 2

Geben Sie den Heap im Array-Schreibweise an:



(*) Weitere Anmerkungen

- InsertionSort hat eine Worst-Case Laufzeit von $O(n^2)$
- InsertionSort hat eine Worst-Case Laufzeit von $O(n)$
- QuickSort hat eine Average Laufzeit von $O(n \log n)$
- QuickSort hat eine Average Laufzeit von $O(n^2)$