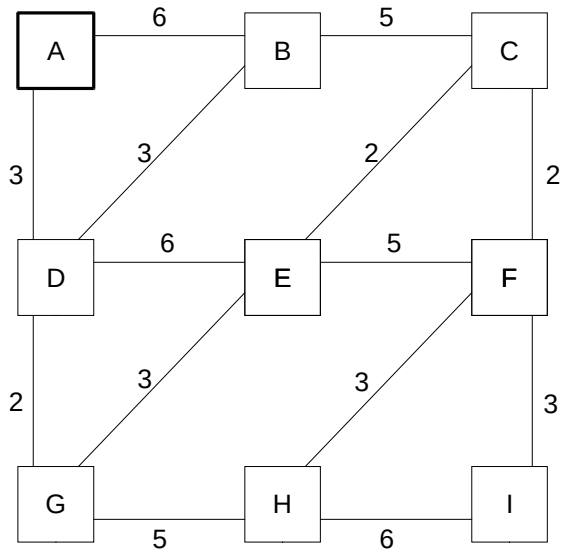
Vorname Nachname Matrikel-Nr

Diese Klausur ist mein **letzter Prüfungsversuch** (bitte ankreuzen): Ja ○ Nein ○

Ihre Lösung für Aufgabe 1 können Sie direkt auf dieses bedruckte Blatt schreiben. Schreiben Sie jede andere Lösung auf die *Vorderseite* eines Ihrer Lösungsblätter (und lassen Sie die Rückseiten Ihrer Lösungsblätter **leer**).

Aufgabe 1 (20 Punkte): Betrachten Sie den folgenden Graphen:



Teilaufgabe 1.1. Berechnen Sie nach dem Dijkstra-Algorithmus die kürzesten Wege vom Knoten **A** zu allen anderen Knoten, indem Sie die folgende Tabelle ausfüllen (wie im SU und den Ü behandelt):

besucht	**Knoten**											**Min**
	A											
	B											
	C											
	D											
	E											
	F											
	G											
	H											
	I											

Teilaufgabe 1.2.
Über welche Knoten führt der kürzeste Weg von **A** nach **I**?
Und wie lang ist dieser Weg?

Aufgabe 2 (20 Punkte): Schreiben Sie eine **rekursive** Methode ("Schleifen dürfen hier nicht rein!") entsprechend der folgenden Spezifikation:

```
1      static boolean ist2erPotenz(long n) {
2         // Verlaesst sich darauf, dass n positiv (d.h. groesser als 0) ist.
3         // Liefert true, wenn n eine (positive) 2-er-Potenz ist,
4         // und liefert sonst false.
5         // Positive 2-er-Potenzen sind die Zahlen 1, 2, 4, 8, 16, ... .
6         // Als 2-er-Zahlen (Binärzahlen) dargestellt sehen diese Zahlen
7         // so aus: 1, 10, 100, 1000, 10000, ...
8         ...
9      }
```

Tipp: Wir haben behandelt, wie man die einzelnen Ziffern einer b-er-Zahl (z.B. einer 2-er-Zahl) berechnen kann.

Aufgabe 3 (20 Punkte): Betrachten Sie die folgenden Variablenvereinbarungen:

```
1         String[]   r02 = {"A", null, ""};
2         String[][] r01 = {r02, null, r02, {}};
```

3.1. Stellen Sie die Variablen `r02` und `r01` als Bojen dar.

3.2. Welche *Referenz* hat (in Ihrer Bojendarstellung) die Variable `r01`?

3.3. Welchen *Wert* hat (in Ihrer Bojendarstellung) die Variable `r02`?

3.4. Welchen *Wert* hat (in Ihrer Bojendarstellung) die Variable `r02[2]`?

Aufgabe 4 (20 Punkte):

4.1. Wie wurde im SU der Begriff **Schritt** definiert (bei der Analyse von Algorithmen)?

4.2. Was muss gelten, damit ein binärer Baum **sortiert** ist?

4.3. *Ungefähr* wie viele *Knoten* kann ein binärer Baum
mit 54 Ebenen (oder: mit der Tiefe 54) höchstens enthalten?
Geben Sie als Antwort eine zweistellige Dezimalzahl und ein deutsches Zahlwort an
(wie z.B. *12 Tausend* oder *37 Trillionen* oder so ähnlich).

4.4. *Genau* wie viele *Ebenen* (oder: welche Tiefe) hat ein binärer Baum
mit 5 Trillionen Knoten *mindestens*?
Geben Sie als Antwort keine Formel oder andere Rechenaufgabe an,
sondern eine Dezimalzahl (wie z.B. *75* oder *123* oder so ähnlich).

4.5. Was wissen Sie über die Werte der Literale
`0.1` (vom Typ `double`) und
`0.1F` (vom Typ `float`)?
Versuchen Sie, diese Frage mit 2 oder 3 kurzen, einfachen Sätzen zu beantworten.

Aufgabe 5 (20 Punkte): Im Projekt 1 sollten Sie eine Klasse namens `LongSpeicher10` entwickeln, die etwa wie folgt aussieht:

```
1   class LongSpeicher10 implements LongSpeicher {
2      // ---------------------------------------------------------------------
3      private long[] speicher;
4      private int    nfi = 0; // naechster freier Index
5
6      public LongSpeicher10(int groesse) {
7         speicher = new long[groesse];
8      }
9      // ---------------------------------------------------------------------
10     private int index(long n) {
11        // Liefert den kleinsten Index i fuer den gilt: speicher[i] == n
12        // oder -1, wenn n nicht im speicher ist.
13        ...
14     }
15     // ---------------------------------------------------------------------
16     ...
17     // ---------------------------------------------------------------------
18     public boolean loesche(long n) {
19        // Loescht ein Vorkommen von n in diesem Speicher, und liefert true.
20        // Liefert false falls n nicht in diesem Speicher vorkommt.
21        ...
22     }
23     // ---------------------------------------------------------------------
24     ...
25     // ---------------------------------------------------------------------
26  } // class LongSpeicher10
```

Wie sollte die Methode `loesche` aussehen? Geben Sie die Methode *vollständig* an, einschließlich der ersten Zeile (nur den Anfangskommentar dürfen Sie als bekannt voraussetzen und weglassen).

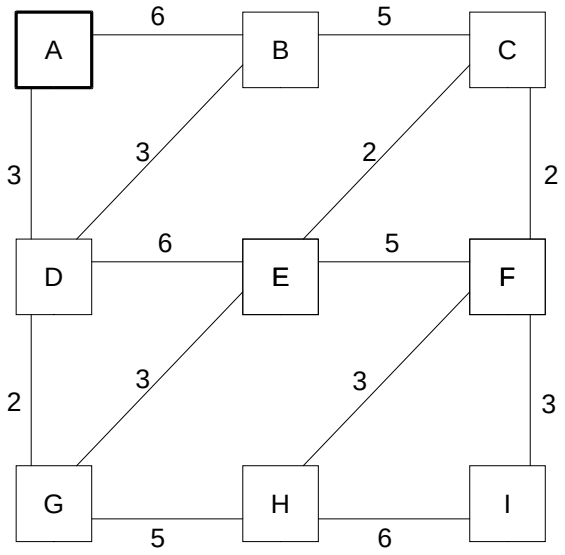
Beurteilung der Klausur:

Punkte↓	
Aufgabe 1:	Note:
Aufgabe 2:	Datum: 24.07.2015
Aufgabe 3:	
Aufgabe 4:	
Aufgabe 5:	
Summe:	

Korrigierte Beurteilung der Klausur:

Punkte↓	
Aufgabe 1:	Note:
Aufgabe 2:	Datum:
Aufgabe 3:	
Aufgabe 4:	
Aufgabe 5:	
Summe:	

Lösung 1 (20 Punkte): Betrachten Sie den folgenden Graphen:



1.1. Berechnen Sie nach dem Dijkstra-Algorithmus die kürzesten Wege vom Knoten A zu allen anderen Knoten, indem Sie die folgende Tabelle ausfüllen (wie im SU und den Ü behandelt):

besucht	**Knoten**		**A 0**	**D 3**	**G 5**	**B 6**	**E 8**	**C 10**	**H 10**	**F 12**	**I 15**	**Min**
1	A	0										0
4	B	inf	6	6								6
6 (7)	C	inf				11	10					10
2	D	inf	3									3
5	E	inf		9	8							8
8	F	inf					13	12	13			12
3	G	inf		5								5
7 (6)	H	inf			10							10
9	I	inf							16	15		15

Bei einer alternativen Lösung wird C *nach* H (statt C *vor* H) besucht.

1.2. Über welche Knoten führt der kürzeste Weg von A nach H?

A - 3- D -2 - G - 3 - E - 2 - C - 2 - F - 3 - I, Länge 3+2+3+2+2+3 gleich 15

Lösung 2 (20 Punkte):

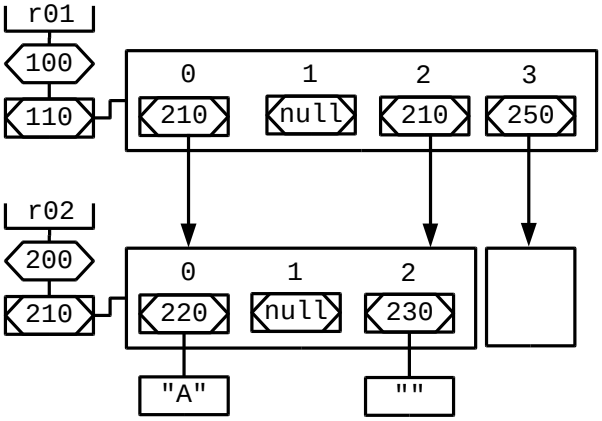
```
1       static boolean ist2erPotenz(long n) {
2          // Verlaesst sich darauf, dass n positiv (d.h. groesser als 0) ist.
3          // Liefert true, wenn n eine (positive) 2-er-Potenz ist,
4          // und liefert sonst false.
5          // Positive 2-er-Potenzen sind die Zahlen 1, 2, 4, 8, 16, ... .
6          // Als 2-er-Zahlen (Binärzahlen) dargestellt sehen diese Zahl
7          // so aus: 1, 10, 100, 1000, 10000, ...
8
9          if (n==1) return true;
10         return n%2==0 && ist2erPotenz(n/2);
11      }
```

Lösung 3 (20 Punkte):

Betrachten Sie die folgenden Variablenvereinbarungen:

```
1         String[]   r02 = {"A", null, ""};
2         String[][] r01 = {r02, null, r02, {}};
```

3.1. Stellen Sie die Variablen `r02` und `r01` als Bojen dar.



3.2. Welche *Referenz* hat (in Ihrer Bojendarstellung) die Variable `r01`? **100**

3.3. Welchen *Wert* hat (in Ihrer Bojendarstellung) die Variable `r02`? **210**

3.4. Welchen Wert hat (in Ihrer Bojendarstellung) die Variable `r02[2]`? **230**

Lösung 4 (20 Punkte)

4.1. Wie wurde im SU der Begriff **Schritt** definiert (bei der Analyse von Algorithmen)?

Ein Schritt ist eine Befehlsfolge, von der es plausibel ist anzunehmen, dass ihre Ausführungszeit immer etwa gleich groß ist oder zumindest eine bestimmte feste Zeit nicht überschreitet.
Insbesondere darf die Ausführungszeit für einen Schritt nicht von der Problemgröße N abhängen.

4.2. Was muss gelten, damit ein **binärer Baum** sortiert ist?

Ein binärer Baum ist sortiert wenn jeder Knoten einen Schlüssel enthält und
für jeden Knoten k gilt: Der Schlüssel von k ist
- größer als alle Schlüssel im linken Unterbaum von k und
- kleiner als alle Schlüssel im rechten Unterbaum von k

4.3. *Ungefähr* wie viele *Knoten* kann ein binärer Baum
mit 54 Ebenen (oder: mit der Tiefe 54) höchstens enthalten?
Geben Sie als Antwort eine zweistellige Dezimalzahl und ein deutsches Zahlwort an
(wie z.B. *12 Tausend* oder *37 Trillionen* etc.). **Ungefähr 16 Billiarden Knoten**

4.4. *Genau* wie viele *Ebenen* (oder: welche Tiefe) hat ein binärer Baum
mit 5 Trillionen Knoten mindestens?
Geben Sie als Antwort keine Formel oder andere Rechenaufgabe an, sondern eine Dezimalzahl (wie z.B. *75* oder *123* etc.). **Mindestens 63 Ebenen**

4.5. Was wissen Sie über die Werte der Literale
`0.1` (vom Typ `double`) und
`0.1F` (vom Typ `float`)?
Der Wert des Literals `0.1` ist (ein bisschen) größer als die Zahl 0.1 (ein Zehntel).
Der Wert des Literals `0.1F` ist (ein bisschen) größer als der Wert des Literals `0.1`.

Lösung 5 (20 Punkte):

Wie sollte die Methode `loesche` aussehen? Geben Sie die Methode vollständig an, einschließlich der ersten Zeile (nur den Anfangskommentar dürfen Sie als bekannt voraussetzen und weglassen).

```
1       public boolean loesche(long n) {
2          // Loescht ein Vorkommen von n in diesem Speicher, und liefert true.
3          // Liefert false falls n nicht in diesem Speicher vorkommt.
4
5          int i = index(n);
6          if (i == -1) return false;
7
8          speicher[i] = speicher[--nfi];
9          return true;
10      }
```