

Name:

Vorname:

Matrikelnummer:

Studiengang:

Klausur: Programmierung WS09/10

Hinweise:

- Tragen Sie Ihren Namen und Vornamen sowie Ihre Matrikelnummer in die Kopfzeile jedes Blattes ein.
- Es sind keine Hilfsmittel zugelassen.
- Die Bearbeitungszeit beträgt 110 Minuten plus 10 Minuten Einlesezeit.
- Es sind keine eigenen leeren Blätter zugelassen. Der Platz unter jeder Aufgabe sollte zur Bearbeitung ausreichen. Benötigen Sie dennoch weiteres Papier, so können Sie dies bei der Aufsicht anfordern. Die zusätzlichen Blätter werden sofort hinten an die Klausur getackert.
- Die Klausur muss mit einem (nicht roten) Stift geschrieben werden. Lösung, die mit Bleistift geschrieben wurden, können nicht anerkannt werden.
- Die Benutzung eines Handys (auch zum ablesen der Uhrzeit) wird als Täuschungsversuch gewertet.

Wir wünschen Ihnen viel Erfolg!

Aufgabe:	max. Punkte:	erreicht:
Graphen und Arrays	42	
Baum und Rekursion	36	
Erweiterung von Klassen	22	
Summe	100	

Name:

Vorname:

Matrikelnummer:

Aufgabe 1: Graphen und Arrays [42 Punkte]

Ein gerichteter Graph sei (ähnlich wie in der Vorlesung) durch die folgenden Klassen definiert.

Beachten Sie die zusätzlichen int-Variablen `Anzahl`, `Index` und `Zahl`.

```
class Graph { Knoten Kopf, Fuss; // Liste der Knoten.
              int    Anzahl;      // Anzahl der Knoten im Graphen.
              ...
            }

class Knoten { String Bez;        // Bezeichnung des Knotens.
              int    Index;       // Index des Knotens = Position in der Liste der Knoten.
              Knoten Nf;         // Verweis auf den Nachfolger in der Liste der Knoten.
              Kante  Kopf, Fuss; // Liste der Kanten, die von dem Knoten ausgehen.
              ...
            }

class Kante { int    Zahl;        // Wert der Kante.
              Kante  Nf;         // Verweis auf den Nachfolger in der Liste der Kanten.
              Knoten Kante;      // Verweis auf den Zielknoten der Kante („Pfeilspitze“).
              ...
            }
```

Wir gehen davon aus, dass bereits ein korrekt aufgebauter Graph existiert, in dessen Kanten die Variable `Zahl` jeweils mit einer natürlichen Zahl belegt ist.

a) Methode in der Klasse Graph: **void indiziere()**

Programmieren Sie den Rumpf dieser Methode, die den Knoten entsprechend ihrer Reihenfolge in der Liste der Knoten einen fortlaufenden Index zuordnen soll (der erste Knoten in der Liste: 0, der zweite Knoten: 1, der dritte Knoten: 2, usw.). Der Index ist in die Variable `Index` einzutragen.

Außerdem ist die Anzahl der Knoten in die Variable `Anzahl` des Graphen einzutragen. [5 Punkte]

Name:

Vorname:

Matrikelnummer:

b) Methode in der Klasse Graph: `int[][] Matrix()`

Programmieren Sie den Rumpf dieser Methode, die den Graphen durch eine zweidimensionale int-Matrix repräsentieren soll. Die Zeilen- und die Spaltenanzahl der Matrix sind gleich der Knotenanzahl des Graphen. Wenn eine Kante von einem Knoten mit Index i zu dem Knoten mit Index j führt, wird in das Element `[i][j]` der Matrix die Zahl dieser Kante eingetragen. Andernfalls wird eine 0 eingetragen.

[12 Punkte]

Name:

Vorname:

Matrikelnummer:

Hinweis für die folgenden Teilaufgaben: Sie können (aber müssen nicht) die Methode `Matrix()` aufrufen und dann mit der Matrix-Darstellung des Graphen arbeiten.

c) Methode in der Klasse Graph: `int Summe(Knoten k)`

Programmieren Sie den Rumpf dieser Methode, die für den durch `k` gegebenen Knoten die Summe der Zahlen in allen Kanten (jeweils Variable `Zahl`) liefert, die von `k` ausgehen oder die in `k` enden (d.h. „Pfeilanzfang“ oder „Pfeilspitze“ der betreffenden Kanten ist mit dem Knoten `k` verbunden).

[10 Punkte]

Name:

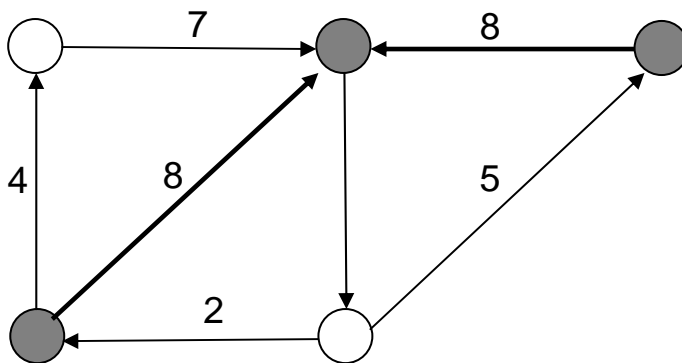
Vorname:

Matrikelnummer:

d) Methode in der Klasse Graph: `int AnzMax()`

Programmieren Sie den Rumpf dieser Methode, welche die Anzahl der Knoten liefert, von denen eine Kante ausgeht oder in die eine Kante mündet, deren Zahl maximal ist (d.h. „Pfeilanzfang“ oder „Pfeilspitze“ einer solchen Kante ist mit dem Knoten verbunden). Die Zahl einer Kante ist maximal, wenn die Zahl keiner anderen Kante im Graphen größer ist.

Hinweis: Es kann eine oder mehrere Kanten im Graphen geben, deren Zahl maximal ist. Wenn es mehrere solche Kanten gibt, dann sind deren Zahlen gleich. Der folgende Beispielgraph zeigt zwei Kanten mit maximalen Zahlen (jeweils 8). Sie sind mit drei Knoten verbunden (grau hervorgehoben), so dass `AnzMax()` den Wert 3 liefern muss. [15 Punkte]



Name:

Vorname:

Matrikelnummer:

Aufgabe 2: Baum und Rekursion [36 Punkte]

Ein Binärbaum sei (wie in der Vorlesung) durch die folgenden Klassen definiert.

```
class Baum { Knoten wurzel;           // Verweis auf den Wurzel-Knoten.
    ...
}

class Knoten { int Zahl;               // Wert des Knotens.
    Knoten links, rechts;             // Linker und rechter Nachfolger-Knoten im Baum.
    ...
}
```

Wir gehen davon aus, dass bereits ein korrekt aufgebauter Baum existiert, in dessen Knoten die Variable `Zahl` jeweils mit einem Wert belegt ist.

Von einer Sortierung der Zahlen kann nicht ausgegangen werden.

a) Methode in der Klasse Baum: **Knoten maxi ()**

Programmieren Sie den Rumpf dieser Methode, die einen Verweis auf den Knoten des Baums liefern soll, dessen `Zahl` die größte Zahl im Baum ist. Wenn es mehrere Knoten gibt, die die größte Zahl enthalten, dann soll ein Verweis auf irgendeinen dieser Knoten geliefert werden. Wenn der Baum leer ist, soll die Methode `null` liefern.

`maxi ()` soll eine rekursive Methode `Knoten maxi (Knoten k)` aufrufen, die Sie ebenfalls programmieren sollen. `maxi (k)` liefert die größte Zahl in dem durch `k` gegebenen Teilbaum.

[14 Punkte]

Name:

Vorname:

Matrikelnummer:

b) Methode in der Klasse Baum: **Knoten mi ni ()**

Markieren Sie in Ihrer Lösung zu Teilaufgabe a) die Programmstellen mit \otimes , die man verändern müsste, um die Methoden `mi ni ()` sowie `mi ni (Knoten k)` zu erhalten, die einen Verweis auf einen Knoten mit der kleinsten Zahl im Baum bzw. Teilbaum liefern.

Sie müssen diese Methoden nicht programmieren.

[1 Punkt]

Name:

Vorname:

Matrikelnummer:

c) Gegeben seien die folgenden Methoden in der Klasse Baum:

```
bool ean s()
```

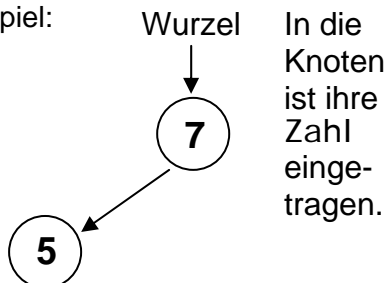
```
{ return s(Wurzel);
}
```

```
bool ean s(Knoten k)
```

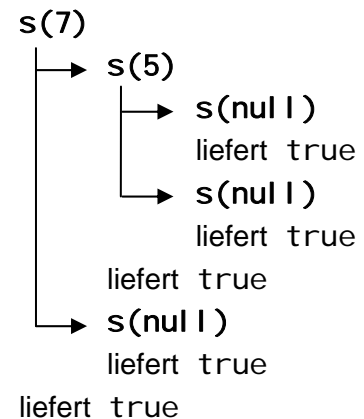
```
{ if (k != null)
    return s(k.links) && s(k.rechts)
    && (k.links == null || maxi(k.links).Zahl <= k.Zahl)
    && (k.rechts == null || mini(k.rechts).Zahl >= k.Zahl);
else
    return true;
}
```

Geben Sie für die beiden folgenden Bäume c1) und c2) jeweils alle Aufrufe der rekursiven Methode `s(Knoten k)` mit dem aktuellen Parameter und dem booleschen Rückgabewert an. Bezeichnen Sie dabei Verweise auf Knoten vereinfachend mit der `Zahl` des betreffenden Knotens bzw. mit `null`. Es empfiehlt sich, die Rekursionstiefe durch entsprechende Einrückung der Aufrufe auszudrücken (wie in der Vorlesung und in den eingebundenen Übungen und wie in dem folgenden Beispiel dargestellt). Die Aufrufe von `maxi(...)` und `mini(...)` müssen nicht dargestellt werden. [9 Punkte]

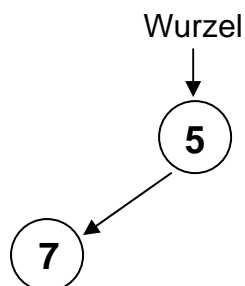
Beispiel:



Vorgegebene Lösung des Beispiels



c1)



Hier die Lösung eintragen:

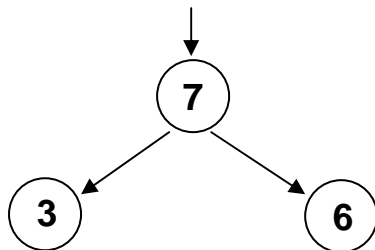
Name:

Vorname:

Matrikelnummer:

c2)

Wurzel



Hier die Lösung eintragen:

Name:

Vorname:

Matrikelnummer:

Zur Lösung der folgenden Teilaufgaben genügen jeweils ca. 1 bis 3 Sätze.

d) Beschreiben Sie die Funktion `s()` verbal: Unter welcher Bedingung liefert sie `true` ? [3 Punkte]

e) Ist die Rekursion der Funktion `s()` linear ? Ist sie schlicht ? [2 Punkte]

f) Erläutern Sie, warum die Funktion `s()` in großen Bäumen zu einer sehr großen Anzahl von Aufrufen der Funktionen `maxi(...)` und `mini(...)` führt, die unnötig hoch ist. [3 Punkte]

g) Geben Sie einen Ansatz an, wie sich die in Teilaufgabe f) beschriebene unnötig hohe Anzahl von Aufrufen der Funktionen `maxi(...)` und `mini(...)` reduzieren ließe, ohne die Funktion von `s()` zu verändern (Die Funktion `s(Knoten k)` spielt dabei keine Rolle).

Es genügt eine verbale Beschreibung der Idee. Sie muss nicht programmiert werden.

[4 Punkte]

Name:

Vorname:

Matrikelnummer:

Aufgabe 3: Erweiterung von Klassen [22 Punkte]

Zur Beantwortung der Fragen in den Teilaufgaben a) bis d) genügen jeweils ca. 1 bis 3 Sätze.

a) Was versteht man unter dem Überschreiben von Methoden ? [2 Punkte]

b) Was versteht man unter dem Überladen von Methoden ? [2 Punkte]

c) Was versteht man unter Widening ? [2 Punkte]

d) Was versteht man unter einer abstrakten Methode ? [2 Punkte]

Name:

Vorname:

Matrikelnummer:

Gegeben seien die folgenden Klassen- und Variablenvereinbarungen:

```
class A
{ int x = 4;
  A(int x) { this.x = x;
            }
  int m(int x) { return x + this.x;
               }
}

class B extends A
{ int x = 5;
  B() { super(6); this.x = x;
        }
  int m(float x) { return (int) x + super.x;
                  }
  int m(int c) { return x - c;
               }
}

B b = new B();

A a = b;
```

- e) Angenommen die zuvor angegebenen Zuweisungen seien ausgeführt worden. Welche Werte liefern dann die folgenden fünf Ausdrücke ? [10 Punkte]

Hier die Lösung eintragen:

a. x

b. x

a. m(3)

b. m(3.0f)

b. m(3)

Name:

Vorname:

Matrikelnummer:

- f) Angenommen in einer Methode ist nur der Verweis a (vom Datentyp A) verfügbar, nicht aber der Verweis b. Schreiben Sie ein (kurzes) Programmstück, das Folgendes erreicht:
Die Variable x der Unterklasse B wird auf 0 gesetzt, wenn das Objekt, auf das a verweist, mit B erweitert wurde. Andernfalls wird die Variable x der Oberklasse auf 0 gesetzt). [4 Punkte]