Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Klausur
Algorithmen und Datenstrukturen
SS 2020 – 18.02.2021

Hinweise:

- Die Bearbeitungszeit beträgt **60 Minuten!**
- Zum Bestehen der Klausur sind 50 Punkte erforderlich.
- Erlaubte Hilfsmittel: keine
- Notieren sie Ihre Lösungen auf eigenem Papier. **Geben Sie bei jeder Lösung Aufgabennummer und Teilaufgabe an.** Versehen Sie jedes Blatt mit Ihren Namen, Matrikelnummer und Studiengang.
- Bitte schreiben Sie deutlich mit einem schwarzen Stift!
- Alle vorgegebenen und zu erstellenden Programmtexte beziehen sich auf die Programmiersprache Java.

Viel Erfolg!

Aufgabe	**Maximalpunkte**	**Erreichte Punkte**
1 (Komplexität, Rekursion)	17	
2 (Listen, Stack)	20	
3 (Binärbaum, Heap)	30	
4 (Sortieren)	18	
5 (Graphen)	20	
Summe	105	

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Aufgabe 1 (Komplexität, Rekursion) — *17 Punkte*

Betrachten Sie folgende Funktion:

```
static int func(int n)
{
  tuwas();

  if (n > 0)
    return func(n-1) + func(n-1);
  else
    return 1;
}
```

a) Berechnen Sie die Anzahl der Aufrufe von `tuwas()` für `n=3`. Geben Sie auch den Rechenweg an.

b) Bestimmen Sie die Anzahl der Aufrufe von `tuwas()` als Funktion von `n`. Geben Sie auch den Rechenweg an.

c) Bestimmen Sie die asymptotische Zeitkomplexität (O-Notation) von `func` unter der Annahme, dass die asymptotische Zeitkomplexität von `tuwas()` O(1) ist

d) Bestimmen Sie die Rekursionstiefe für `n=3`.

e) Bestimmen Sie die Rekursionstiefe in Abhängigkeit von `n`.

f) Berechnen Sie den Funktionswert `func(3)`. Geben Sie auch den Rechenweg an.

g) Welche Funktion berechnet `func` für $n \geq 0$? Geben Sie eine Formel an.

NOTIEREN SIE DIE LÖSUNG AUF IHREM LÖSUNGSBLATT!

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Aufgabe 2 (Listen, Stack) *20 Punkte*

Ihr Dozent ist gerade dabei, sich eine Modelleisenbahn zuzulegen. Derzeit lagern diverse Schachteln auf dem Schreibtisch:

stapel[0] [1] [2][3] [4] [5]

Dies wird durch den folgenden Java-Code nachgebildet. Als Datenstruktur kommen Stacks auf Basis verketteter Listen zum Einsatz. Zunächst definieren wir eine Klasse `Schachtel` und eine Hilfsklasse `Link` analog zur Vorlesung:

```
public class Schachtel {
    …
}

public class Link {
    Schachtel schachtel;
    Link naechster;

    public Link(Schachtel schachtel, Link naechster) {
        this.schachtel = schachtel;
        this.naechster = naechster;
    }
}
```

Jeder Stapel ist ein Stack auf Basis einer einfach-verketteten Liste mit kontrollierten Zugriffspunkten. Auf dem Schreibtisch befinden sich mehrere Stapel (siehe Bild). Das oberste Element der Stapel wird jeweils durch ein Element des Arrays `stapel` referenziert:

```
public class Schreibtisch
{
    Link stapel[];

    public Schreibtisch(int anzahlStapel) {
        stapel = new Link[anzahlStapel];     // Im Beispielbild: 6
    }

    public Schachtel pop(int nummerStapel){
        // Aufgabe a
    }
```

```
    public void push(int nummerStapel, Schachtel schachtel){
        // Aufgabe b
    }

    public void umlegen(int vonStapel, int aufStapel) {
        // Aufgabe c
    }
}
```

a) Implementieren Sie die Methode `pop(int nummerStapel)`, die vom Stapel `nummerStapel` die oberste `Schachtel` entfernt und an den Aufrufer zurückgibt. **Hinweis:** durch die `assert`-Anweisungen ist sichergestellt, dass `nummerStapel` immer einen gültigen Wert hat und mindestens eine `Schachtel` auf dem Stapel vorhanden ist.

```
public Schachtel pop(int nummerStapel)
{
    assert(nummerStapel >= 0 && nummerStapel < stapel.length);
    assert(stapel[nummerStapel] != null);

    NOTIEREN SIE DIE LÖSUNG AUF IHREM LÖSUNGSBLATT!
}
```

b) Implementieren Sie die Methode `push(int nummerStapel, Schachtel schachtel)`, die eine `Schachtel` auf den Stapel `nummerStapel` legt. **Hinweis:** durch die `assert`-Anweisungen ist sichergestellt, dass `nummerStapel` immer einen gültigen Wert hat.

```
public void push(int nummerStapel, Schachtel schachtel)
{
    assert(nummerStapel >= 0 && nummerStapel < stapel.length);

    NOTIEREN SIE DIE LÖSUNG AUF IHREM LÖSUNGSBLATT!
}
```

c) Implementieren Sie die Methode `umlegen(int vonStapel, int aufStapel)`, die eine `Schachtel` vom Stapel `vonStapel` auf den Stapel `aufStapel` legt. **Hinweis:** durch die `assert`-Anweisungen ist sichergestellt, dass `vonStapel` und `aufStapel` immer einen gültigen Wert haben und mindestens eine `Schachtel` auf dem Stapel `vonStapel` vorhanden ist. Sie dürfen die Methoden aus den Aufgabenteilen a) und b) benutzen.

```
public void umlegen(int vonStapel, int aufStapel)
{
    assert(vonStapel >= 0 && vonStapel < stapel.length);
    assert(aufStapel >= 0 && aufStapel < stapel.length);
    assert(stapel[vonStapel] != null);

    NOTIEREN SIE DIE LÖSUNG AUF IHREM LÖSUNGSBLATT!
}
```

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Aufgabe 3 (Binärbaum, Heap) *30 Punkte*

Betrachten Sie die folgende teilweise Implementierung eines Heaps als Binärbaum:

```
public class HKnoten
{
   public int schluessel;
   public HKnoten[] kinder = new HKnoten[2];
}

public class Heap
{
   private HKnoten wurzel;

   // Methoden zum Einfügen und Entfernen
   // …

   public int groessterSchluessel() {
      // Aufgabenteil a)
   }

   public boolean istHeap() {
      return istHeap(wurzel);
   }

   private boolean istHeapKnoten(HKnoten knoten){
      // Aufgabenteil b)
   }

   private boolean istHeap(HKnoten knoten) {
      // Aufgabenteil c)
   }
}
```

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

a) Vervollständigen Sie die Methode `groessterSchluessel`.

Hinweise:
- Sie können davon ausgehen, dass die Wurzel existiert.
- Sie können davon ausgehen, dass der Binärbaum ein Heap ist.

```
public int groessterSchluessel()
{
    assert(wurzel != null);
    assert(istHeap());

    NOTIEREN SIE DIE LÖSUNG AUF IHREM LÖSUNGSBLATT!

}
```

b) Vervollständigen Sie die Methode `istHeapKnoten`, die prüft, ob `knoten` und seine beiden Kinder die Heap-Eigenschaft für Schlüssel erfüllen.

```
private boolean istHeapKnoten(HKnoten knoten)
{

    NOTIEREN SIE DIE LÖSUNG AUF IHREM LÖSUNGSBLATT!

}
```

c) Vervollständigen Sie die Methode `istHeap`, die rekursiv überprüft, ob alle Knoten im Teilbaum mit der Wurzel `knoten` die Heap-Eigenschaft der Schlüssel erfüllen.

Hinweis:
Nutzen Sie die Methode `istHeapKnoten` aus b).

```
private boolean istHeap(HKnoten knoten)
{

    NOTIEREN SIE DIE LÖSUNG AUF IHREM LÖSUNGSBLATT!

}
```

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Aufgabe 4 (Sortieren) *18 Punkte*

In der folgenden Abbildung ist die Implementierung einer Variante eines bekannten Sortieralgorithmus angegeben.

```
public static void sortiere(int[] feld)
{
  for (int a = 1; a < feld.length; a++)
  {
    int temp = feld[a];
    int b = a - 1;

    while ((b >= 0) && (feld[b] < temp))
    {
      feld[b+1] = feld[b];
      b--;
    }

    feld[b+1] = temp;
  }
}
```

a) Wie lautet der Namen des Verfahrens?

b) Ist der Algorithmus stabil? Geben Sie eine Begründung in einem Satz an.

c) Sei `n` die Anzahl der zu sortierenden Elemente. Geben Sie die asymptotische Anzahl an Vergleichen abhängig von `n` im besten (best case) und im schlechtesten Fall (worst case) in O-Notation an.

NOTIEREN SIE DIE LÖSUNG AUF IHREM LÖSUNGSBLATT!

d) Wenden Sie die Implementierung aus Aufgabenteil a) auf das unten angegebene Feld an. Beachten Sie die Sortierrichtung und geben Sie die Reihenfolge der Schlüssel nach jedem Durchlauf der äußeren Schleife an. Benutzen Sie einen senkrechten Strich **|**, um Teilfelder voneinander abzutrennen.

2	8	4	9	7	3

NOTIEREN SIE DIE LÖSUNG AUF IHREM LÖSUNGSBLATT!

Name, Vorname, Matrikelnummer	Studiengang	Sitzplatznummer

Aufgabe 5 (Graphen) — *20 Punkte*

Folgender Graph G ist durch seine Adjazenzmatrix gegeben:

		Nach				
		A	**B**	**C**	**D**	**E**	**F**
Von	**A**		1	7			
	B	1			6		2
	C	7			5		
	D		6	5		4	3
	E				4		
	F		2		3		

a) Zeichnen Sie den gegebenen Graphen. Nutzen Sie alle Informationen aus der Adjazenzmatrix.

b) Geben Sie alle Zyklen des Graphen an.

c) Zeichnen Sie den minimalen Spannbaum von G.

ZEICHNEN BZW. NOTIEREN SIE DIE LÖSUNG AUF IHREM LÖSUNGSBLATT!