Name, Vorname, Matrikelnummer Studiengang

Klausur
Algorithmen und Datenstrukturen
WS 2022/23 – 08.02.2023

Hinweise:

- Die Bearbeitungszeit beträgt **90 Minuten!**
- Zum Bestehen der Klausur sind 50 Punkte erforderlich.
- Erlaubte Hilfsmittel: keine
- Alle vorgegebenen und zu erstellenden Programmtexte beziehen sich auf die Programmiersprache Java.
- Bevor Sie mit der Bearbeitung der Aufgaben beginnen, **müssen** Sie auf allen Blättern Ihren Namen, Ihre Matrikelnummer und Ihren Studiengang eintragen.
- Der Klausurtext enthält ausreichend Platz zur Lösung der Aufgaben. Sollten Sie trotzdem zusätzliches Papier benötigen, wenden Sie sich an die Klausuraufsicht. Die Nutzung eigenen Papiers ist nicht gestattet.
- Sollte Ihre Lösung nicht unmittelbar unter oder neben der Aufgabenstellung stehen, machen Sie bitte einen entsprechenden Hinweis. Streichen Sie diejenigen Teile der von Ihnen geschriebenen Texte deutlich durch, die nicht in die Bewertung eingehen sollen.
- Bitte schreiben Sie deutlich.

Viel Erfolg!

Aufgabe	**Maximalpunkte**	**Erreichte Punkte**
1 (Wissen)	16	
2 (Komplexität, Rekursion)	14	
3 (Listen)	23	
4 (Binärbäume)	22	
5 (Suchen)	12	
6 (Graphen)	18	
Summe	105	

Name, Vorname, Matrikelnummer Studiengang

Aufgabe 1 (Wissen) *16 Punkte*

a) Nennen Sie drei charakteristische Eigenschaften eines Algorithmus‘ (entsprechend der Definition aus der Vorlesung).

b) Notieren Sie zu jeder Aussage, ob sie richtig oder falsch ist.

1. $(n-1)\cdot(n+1) = O(n)$
2. $n\cdot\log(n) + n^2 = O(n^2)$
3. $n^2 + 2^n = O(n^3)$

c) Ergänzen Sie in der Tabelle die Zeitkomplexität der angegebenen Operationen auf Listen mit n Objekten in O-Notation. Die ArrayList habe noch mindestens ein freies Element.

	ArrayList	LinkedList
Anhängen ans Ende		
Zugriff auf i-tes Element		

d) Erläutern Sie die prinzipielle Vorgehensweise eines Divide-and-Conquer-Algorithmus‘.

e) Nennen Sie zwei Sortieralgorithmen, die nach dem Prinzip Divide-and-Conquer arbeiten.

f) Nennen Sie zwei Sortieralgorithmen, deren Zeitkomplexität im schlechtesten Fall O(n log n) ist.

Name, Vorname, Matrikelnummer Studiengang

Aufgabe 2 (Komplexität, Rekursion) *14 Punkte*

Bestimmen Sie für die nachstehenden Prozeduren folgende Informationen:

```
static void proz1(int n)
{
    for (int a=n; a>0; a--)
        for (int b=0; b<a; b++)
            tuwas();

}
```

```
static void proz2(int n)
{
    if (n > 1)
    {
       proz2(n-1);
       proz2(n-1);
    }
    else
       tuwas();
}
```

- Bestimmen Sie die Anzahl der Aufrufe von tuwas() für n=4.
- Bestimmen Sie die Anzahl der Aufrufe von tuwas() als Funktion von n.
- Bestimmen Sie die asymptotische Zeitkomplexität in O-Notation unter der Annahme, dass die asymptotische Zeitkomplexität von tuwas() O(1) ist.

a) Notieren Sie die Lösung in der folgenden Tabelle.

Methode	**Anzahl Aufrufe für n=4**	**Anzahl Aufrufe als Funktion von n**	**O-Notation**
`proz1`			
`proz2`			

b) Bestimmen Sie die Rekursionstiefe von proz2 für `n=4`.

c) Bestimmen Sie die Rekursionstiefe von proz2 in Abhängigkeit von `n`.

Name, Vorname, Matrikelnummer Studiengang

Aufgabe 3 (Listen) *23 Punkte*

Betrachten Sie die folgende teilweise Implementierung einer Prioritätsliste für Aktionen als einfach verkettete Liste:

```
public class Link
{
    public String aktion;
    public int prioritaet;
    public Link naechster;

    public Link(String aktion, int prioritaet, Link naechster){
        this.aktion = aktion;
        this.prioritaet = prioritaet;
        this.naechster = naechster;
    }
}

public class Prioritaetsliste
{
    private Link anfang;

    public Prioritaetsliste(){
        anfang = null; // Leere Liste
    }
    …
    public void einfuegen(String aktion, int prioritaet){
        assert(aktion!=null);
        // Aufgabenteil a)
    }

    private Link findeHoechstePrioritaet()
        // Aufgabenteil b)
    }

    public String entferneAktionMitHoechsterPrioritaet(){
        // Aufgabenteil c)
    }
}
```

---

a) Vervollständigen Sie die Methode `einfuegen`, welche die Aktion am Anfang der verketteten Liste einfügt. **Hinweis**: Sie können davon ausgehen, dass der Parameter `aktion` ungleich `null` ist.

```
public void einfuegen(String aktion, int prioritaet)
{
    assert(aktion!=null);









}
```

b) Vervollständigen Sie die Methode `findeHoechstePrioritaet`, welche eine Referenz auf das Element mit der höchsten Priorität zurückgeben soll. Ist die Liste leer, soll `null` zurückgegeben werden. Sind mehrere Elemente mit der höchsten Priorität vorhanden, so soll die Referenz auf das erste entsprechende Element zurückgegeben werden.

```
private Link findeHoechstePrioritaet()
{




}
```

c) Vervollständigen Sie die Methode `entferneAktionMitHoechsterPrioritaet`, welche die Aktion mit höchster Priorität aus der Liste entfernt und als Ergebnis zurückgibt. Ist die Liste leer, soll `null` zurückgegeben werden.
Hinweis: Nutzen Sie die Methode aus Teilaufgabe b).

```
public String entferneAktionMitHoechsterPrioritaet()
{




}
```

Name, Vorname, Matrikelnummer Studiengang

Aufgabe 4 (Binärbäume) *22 Punkte*

Betrachten Sie die folgende teilweise Implementierung eines Binärbaums:

```
public class Knoten
{
   public int schluessel;
   public Knoten linkesKind;
   public Knoten rechtesKind;
}

public class Binaerbaum
{
   private Knoten wurzel;

   private int minimum(Knoten knoten)
   {
      assert(knoten != null)
      // Aufgabenteil a)
   }

   private int maximum(Knoten knoten)
   {
      assert(knoten != null)
      // Aufgabenteil b)
   }

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

   private boolean istSuchbaum(Knoten k)
   {
      // Aufgabenteil c)
   }


}
```

Wichtiger Hinweis: Der Binärbaum muss nicht zwingend ein Suchbaum sein!

Name, Vorname, Matrikelnummer Studiengang

a) Vervollständigen Sie die Methode `minimum`, die rekursiv den kleinsten Schlüssel im Teilbaum mit der Wurzel `knoten` bestimmen soll.

```
private int minimum(Knoten knoten)
{
   assert(knoten != null);









}
```

b) Wie muss der Code aus Teilaufgabe a) geändert werden, um den größten Schlüssel im Teilbaum mit der Wurzel `knoten` zu bestimmen?

c) Vervollständigen Sie die Methode `istSuchbaum`, die rekursiv überprüft, ob der Baum mit der Wurzel `knoten` die Suchbaumeigenschaft besitzt.
Hinweise:

- Ein leerer Baum erfüllt die Suchbaumeigenschaft.
- Nutzen Sie die Methoden aus a) und b).

```
private boolean istSuchbaum(Knoten knoten)
{









}
```

Name, Vorname, Matrikelnummer Studiengang

Aufgabe 5 (Suchen) *12 Punkte*

a) Führen Sie auf dem folgenden Feld eine binäre Suche nach dem Schlüssel 17 durch. Geben Sie für jeden Schritt den Indexbereich der Suche und den Index des Vergleichselements an.

0	1	2	3	4	5	6	7	8
2	5	6	9	10	12	15	16	18

b) Tragen Sie nacheinander die folgenden Schlüssel in der angegebenen Reihenfolge in die Hash-Tabelle ein: 4, 13, 26, 8, 24 und 2. Die Auflösung von Kollisionen soll durch Lineares Sondieren mit der Schrittweite c=3 erfolgen. Geben Sie für jeden Schlüssel die Berechnung des Index in die Tabelle an.

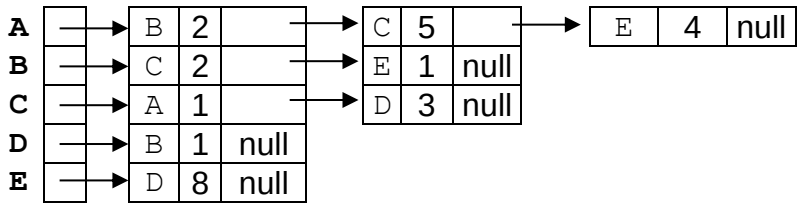
Als Hash-Funktion soll h(k) = k mod 11 verwendet werden.

0	1	2	3	4	5	6	7	8	9	10

Name, Vorname, Matrikelnummer Studiengang

Aufgabe 6 (Graphen) *18 Punkte*

Folgender gewichteter Graph ist durch seine Adjazenzlistenstruktur gegeben:



a) Zeichnen Sie den Graphen.

b) Führen Sie auf dem obigen Graphen den Algorithmus von Dijkstra zur Bestimmung minimaler Wege mit dem Startknoten A aus. Ergänzen Sie dazu die folgende Tabelle.

	A length/pred	B length/pred	C length/pred	D length/pred	E length/pred

c) Bestimmen Sie auf Basis der Tabelle aus Aufgabenteil b) die kürzesten Wege von Knoten A zu den Knoten D und E. Erläutern Sie kurz, wie Sie dabei vorgehen.

Name, Vorname, Matrikelnummer Studiengang

Zusatzblatt