

Prüfung Algorithmen und Datenstrukturen I

Datum	:	21.07.2015, 10:30 Uhr
Bearbeitungszeit	:	90 Minuten
Prüfer	:	Prof. Dr. Oliver Braun
Hilfsmittel	:	Keine
Erreichbare Punkte	:	90

Name: _____

Vorname: _____

Matrikelnummer: _____ Studiengruppe: _____

Hörsaal: _____ Platz Nr.: _____

Unterschrift: _____

Bitte kontrollieren Sie, ob Sie eine vollständige Angabe mit 6 Aufgaben auf 8 Seiten erhalten haben.

Aufgabe	1	2	3	4	5	6	Summe
max. Punkte	15	15	15	15	15	15	90

Anmerkungen:

- Nutzen Sie einen dokumentenechten Stift für alles was bewertet werden soll. Auch bei Skizzen ist die Verwendung eines **Bleistifts nicht** zulässig.
- Schreiben Sie die Lösungen in die dafür vorgesehenen Kästchen bzw. direkt zur Aufgabe. Sollte Ihnen der Platz dabei nicht reichen, benutzen Sie die Rückseite **und vermerken Sie das bei der entsprechenden Aufgabe!**

Aufgabe 1 (15 Punkte)

Gegeben sei folgende C++-Klasse für einen binären Suchbaum:

```
class BinTree {  
private:  
    int val;  
    BinTree *left = nullptr, *right = nullptr;  
public:  
    bool search(const int) const;  
    vector<int> *inorder() const;  
};
```

(a) Geben Sie eine Implementierung für `search` an:

(7)

(b) Geben Sie eine Implementierung für `inorder` (symmetrische Reihenfolge) an:

(8)

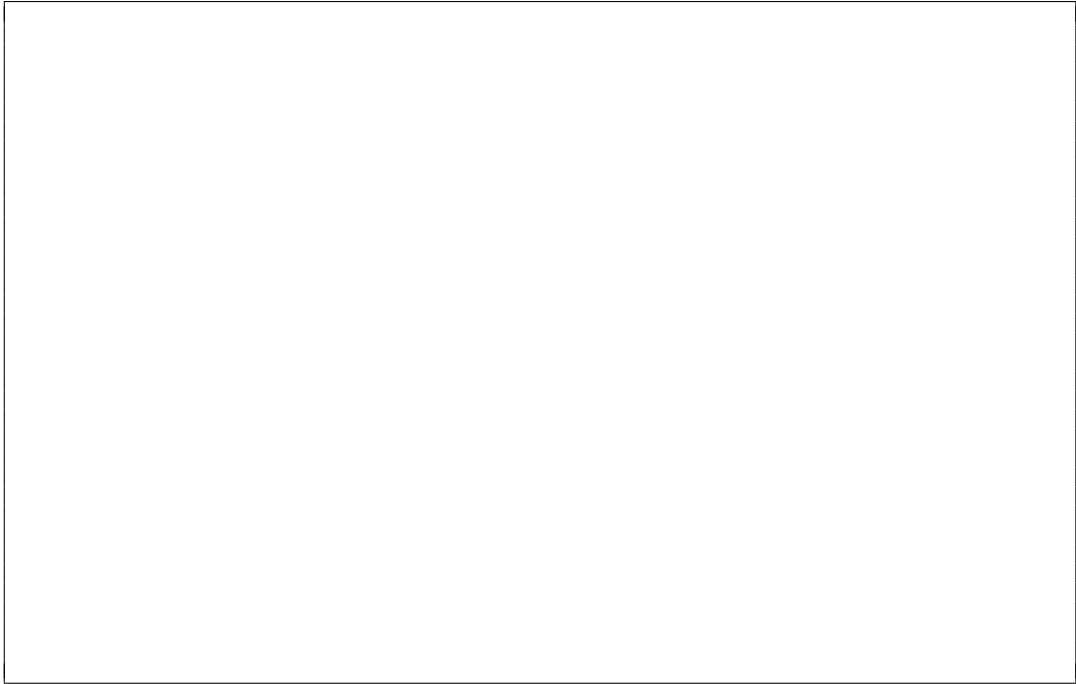
Anmerkung: Sie können an einen Vektor `*v1` den Inhalt von `*v2` folgendermaßen anhängen: `v1->insert(v1->end(), v2->begin(), v2->end())`

Aufgabe 2 (15 Punkte)

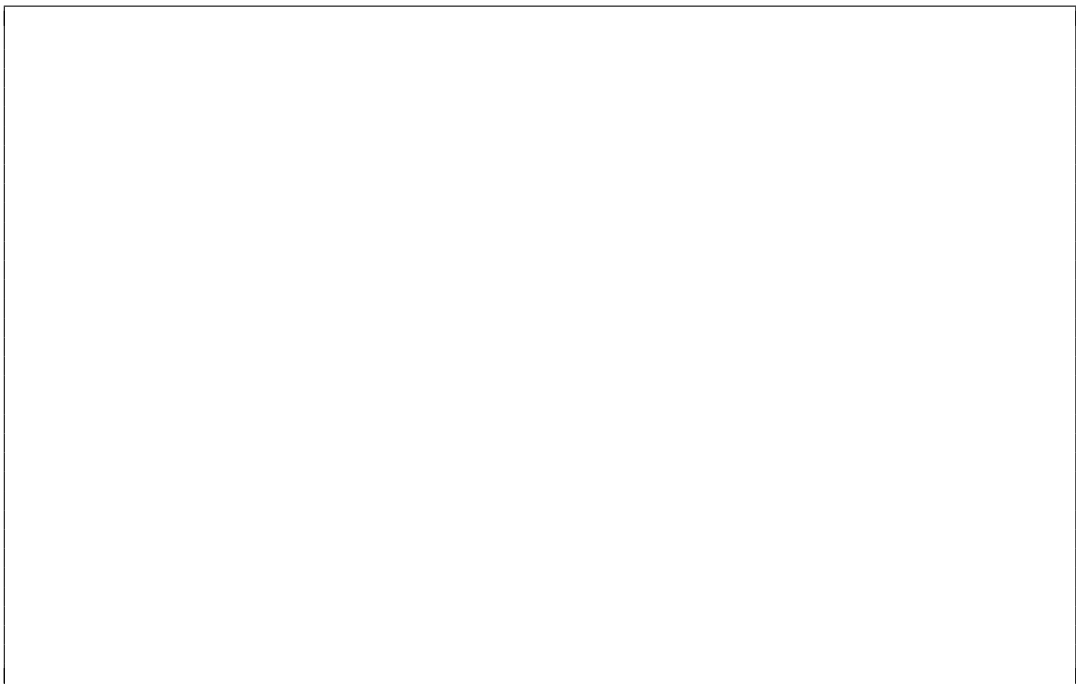
Gegeben sei die Folge

3, 1, 5, 6, 8, 7, 2, 4

- (a) Sortieren Sie die Folge mit dem reinen 2-Wege-Mergesort. Kennzeichnen Sie die Teillisten und geben Sie alle Zwischenergebnisse an. (7)



- (b) Sortieren Sie die Folge mit dem natürlichen 2-Wege-Mergesort. Kennzeichnen Sie die Teillisten und geben Sie alle Zwischenergebnisse an. (8)

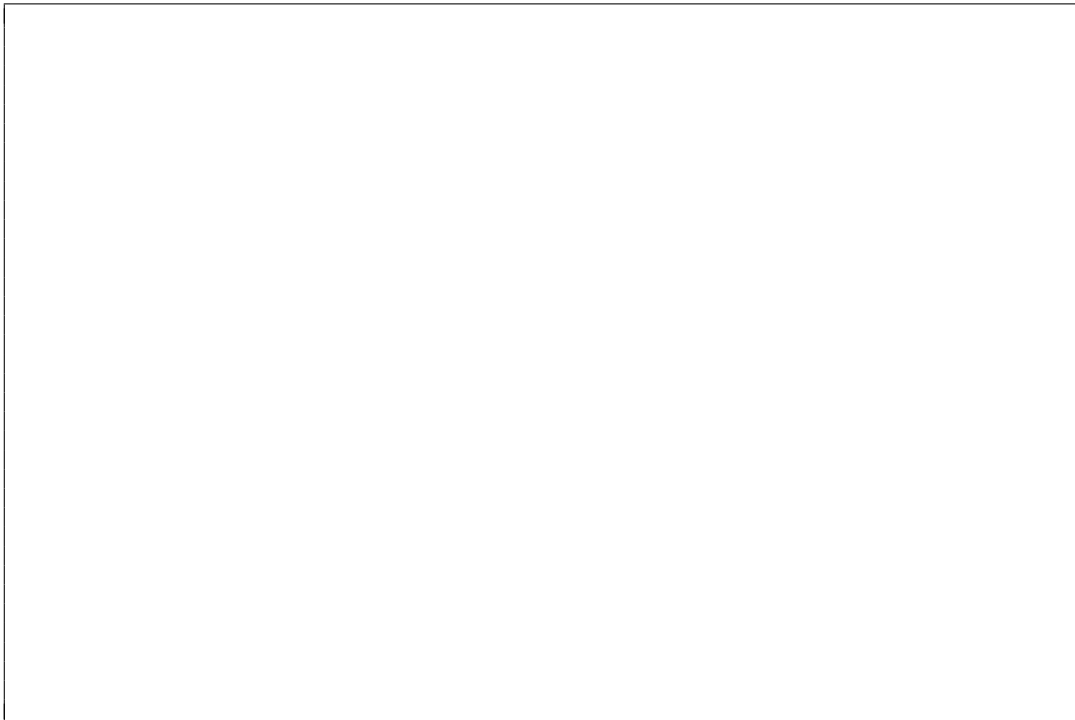


Aufgabe 3 (15 Punkte)

- (a) Erstellen Sie einen Treap aus folgenden Werten, wobei die erste Komponente des Tupels jeweils der *Schlüssel* und die zweite Komponente die *Priorität* ist: (10)
(20,22), (22,20), (11,4), (21,21), (12,3), (19,2), (28,28), (18,5)



- (b) Erläutern Sie wie das Element (17,1) schrittweise in den Treap eingefügt wird? (5)
Geben sie außerdem den Treap nach dem abgeschlossenen Einfügen an.



Aufgabe 4 (15 Punkte)

Gegeben sei folgende C++-Klasse für eine Liste:

```
class List {
private:
    int val;
    List *next = nullptr;
public:
    List(const int v) : val(v) {}
    List(const int v, List *l) : val(v), next(l) {}
    List *cons(const int v) {
        return new List(v, this);
    }
    void sort();
    friend std::ostream& operator<<(std::ostream&, const List&);
};
```

- (a) Geben Sie eine Implementierung für den Operator << an, so dass eine Liste folgendermaßen ausgegeben wird: [5, 7, 18, 12, 13, 19, 5, 1] (5)

- (b) Geben Sie eine Implementierung für `sort` an, die die Liste **in situ**, nach einem beliebigen **elementaren Sortierverfahren**, sortiert. Geben Sie an welches Sortierverfahren sie implementiert haben. (10)

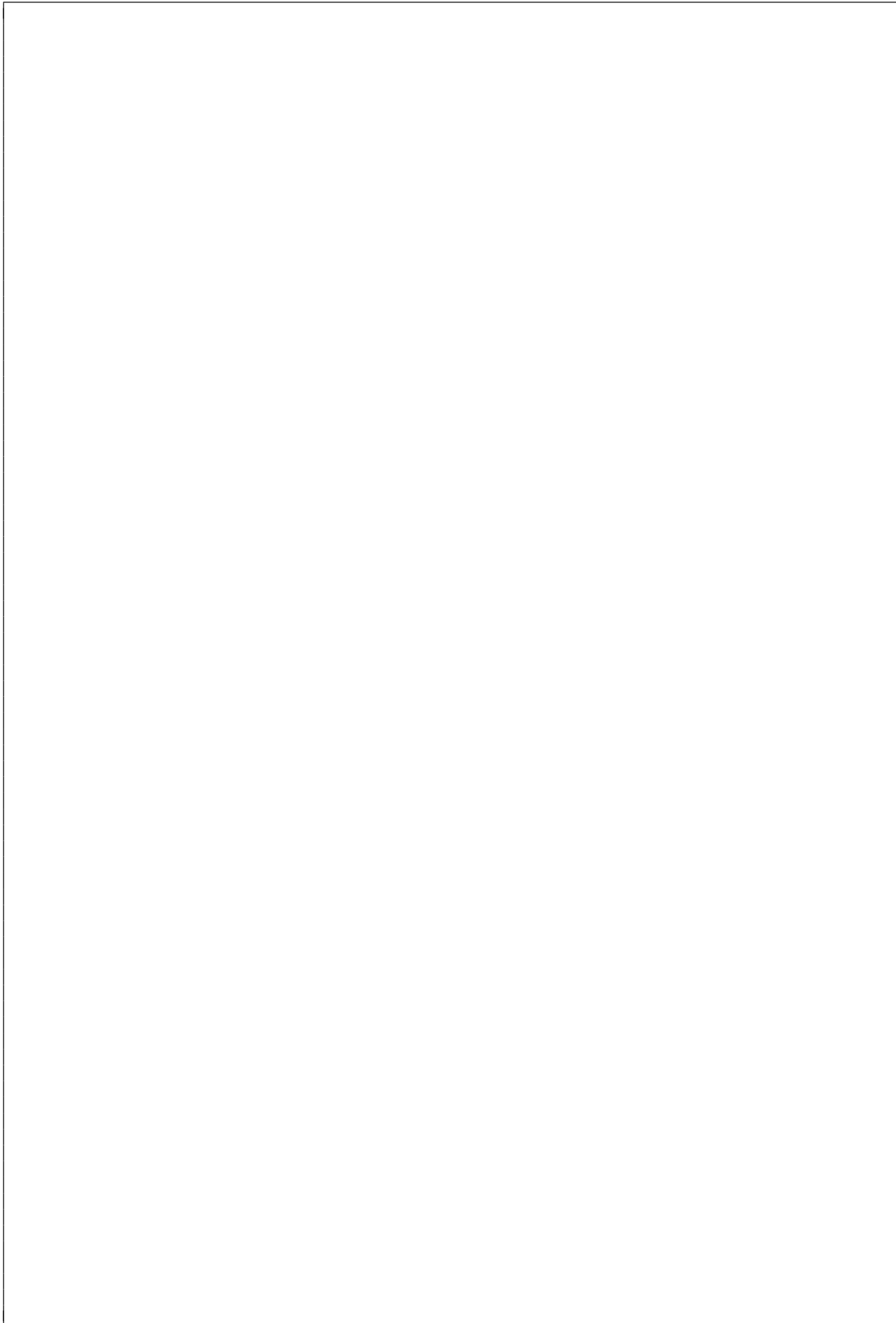
Eine `main`-Funktion könnte so aussehen:

```
int main()
{
    auto list = new List(1);
    list = list->cons(5)->cons(19)->cons(13)
           ->cons(12)->cons(18)->cons(7)->cons(5);
    cout << *list << endl;
    list->sort();
    cout << *list << endl;
    return 0;
}
```

Die Ausgabe dieser Funktion ist:

```
[ 5, 7, 18, 12, 13, 19, 5, 1]
```

```
[ 1, 5, 5, 7, 12, 13, 18, 19]
```



Aufgabe 5 (15 Punkte)

Gegeben sei der AVL-Baum der nur aus einer Wurzel mit dem Schlüssel 50 besteht.

Fügen Sie in den AVL-Baum nacheinander die Werte 10, 30, 40, 60 und 55 ein und geben Sie alle **Zwischenergebnisse** als AVL-Bäume inklusive der Balancefaktoren an, d.h. den AVL-Baum mit 10 und 50, den AVL-Baum mit 10, 50 und 30, ...

Anmerkung: Sie brauchen keine Blätter zeichnen, es reichen die inneren Knoten.

Aufgabe 6 (15 Punkte)

Gegeben sei der folgende C++-Algorithmus:

```
void alg(vector<int> &list)
{
    for (int i = 0; i < list.size()-1; i++) {
        for (int j = list.size(); j > 0;) {
            if (i < j) {
                int k;
                for (k = 0; k < i; k++) {
                    list[i] += list[j-1];
                }
                j = i - k;
            }
        }
    }
}
```

- (a) Geben Sie die Abschätzung des Wachstums der Laufzeit des Algorithmus in der *Groß-Oh*-Notation an und **begründen** Sie Ihre Angabe. (8)

- (b) Geben Sie einen Algorithmus an, der die Liste exakt gleich verändert, aber in Bezug auf die Laufzeit effizienter ist. Geben Sie den Wert Ihres Algorithmus in der *Groß-Oh*-Notation an. (7)