Hochschule für angewandte Wissenschaften München
Fakultät für Informatik und Mathematik

Studiengruppe IF 3 B
Wintersemester 2014/15

Prüfung Algorithmen und Datenstrukturen I

Datum	:	20.01.2015, 10:30 Uhr
Bearbeitungszeit	:	90 Minuten
Prüfer	:	Prof. Dr. Oliver Braun
Hilfmittel	:	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> *preorder()  const;
};
```

(a) Geben Sie eine Implementierung für `search` an: (7)

(b) Geben Sie eine Implementierung für `preorder` (Hauptreihenfolge) 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 folgende Liste, die in einem Array abgelegt ist:

```
19, 13, 14, 7, 8, 12, 3, 2
```

Die Liste soll mit dem Heapsort-Verfahren aus der Vorlesung sortiert werden.

Geben Sie alle Heaps an, die als Zwischenergebnisse beim Sortieren erzeugt werden.

Kennzeichnen Sie die Heaps immer mit `HeapX`, wobei Sie für `X` die Anzahl der noch zu sortierenden Elemente einsetzen, also `Heap8`, `Heap7`, …, `Heap3`. Die einelementige und die zweielementige Heap müssen Sie natürlich nicht mehr angeben.

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: (7,10), (8,9), (10,12), (13,6), (14,8), (15,4), (18,7), (19,18) (10)

(b) Erläutern Sie wie das Element (17,3) schrittweise in den Treap eingefügt wird? Geben sie außerdem den Treap nach dem abgeschlossenen Einfügen an. (5)

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 10 besteht.

Fügen Sie in den AVL-Baum nacheinander die Werte 20, 30, 40, 50, 60 und 55 ein und geben Sie alle **Zwischenergebnisse** als AVL-Bäume inklusive der Balancefaktoren an, d.h. den AVL-Baum mit 10 und 20, den AVL-Baum mit 10, 20 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)