Hochschule für angewandte Wissenschaften München
Fakultät für Informatik und Mathematik

Studiengruppe IF 4 B
Sommersemester 2016

Prüfung Algorithmen und Datenstrukturen I

Datum	:	12.07.2016, 14:30 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 5 Aufgaben auf 7 Seiten erhalten haben.

Aufgabe	1	2	3	4	5	Summe
max. Punkte	30	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 (30 Punkte)

Gegeben sei folgende C++-Klasse für eine doppelt verkettete **und sortierte** Liste:

```
class List {
private:
    struct Elem {
        const int value;
        Elem *prev = nullptr, *next = nullptr;
        Elem(const int v) : value(v) {}
    };
    Elem *head = nullptr;
public:
    bool search(const int) const;
    void deleteElem(const int);
    void insert(const int);
};
```

(a) Geben Sie eine Implementierung für `deleteElem` an: (10)

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

(c) Geben Sie eine Implementierung für `insert` an: (13)

Aufgabe 2 (15 Punkte)

Gegeben sei folgende Liste, die in einem Array abgelegt ist:

```
81, 33, 45, 14, 20, 21, 13, 4
```

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: (10,12), (15,22), (8,4), (7,5), (22,28), (18,18), (12,9) (10)

(b) Erläutern Sie wie das Element (17,6) 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 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 5 (15 Punkte)

Gegeben sei der AVL-Baum der nur aus einer Wurzel mit dem Schlüssel 13 besteht.

Fügen Sie in den AVL-Baum nacheinander die Werte 11, 31, 29, 30 und 28 ein und geben Sie alle **Zwischenergebnisse** als AVL-Bäume inklusive der Balancefaktoren an, d.h. den AVL-Baum mit 13 und 11, den AVL-Baum mit 13, 11 und 31, …

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