Algorithmen und Datenstrukturen 1	schriftliche Einzelpruefung	14.10.2003	1234567 Anonymous	1

Aufgabe 1 [2.0]

Gegeben ist folgendes rekursive Programm:

```
int f(n) {
    int s = 0;
    if(n<=1) return 1;
    else s = s + f(n/2) + f(n/2);
    return s;
}
```

a. [0.2] Entwickeln Sie die *Rekurrenzgleichung*, die das Laufzeitverhalten der Funktion f(n) beschreibt.

b. [0.5] Bestimmen Sie das *asymptotische Laufzeitverhalten* der Funktion f(n) mit Hilfe des Master Theorems.

c. [0.4] Definieren Sie die Begriffe $O(n)$ und $\Omega(n)$ bei der Laufzeitabschätzung eines Programms?

d. [0.4] Was verstehen Sie unter dem Begriff der *Algorithmischen Lücke*? Geben Sie jeweils ein Beispiel für eine Problem an, wo die Algorithmische Lücke geschlossen bzw. offen ist.

e. [0.5] Skizzieren Sie den Programmgraphen des obigen Programms und berechnen Sie dessen zyklomatische Komplexität nach der Metrik von McCabe.

Aufgabe 2 [2.5]

a. [1.2] Fügen Sie die Werte 5, 8, 3, 4, 6, 7, 1, 2, 9, 11, 10 und 12 in dieser Reihenfolge in einen ursprünglich leeren *2-3-4 Baum* entsprechend der Vorlesung ein.

b. [0.8] Löschen Sie die Werte 9, 5, 7 und 10 (in dieser Reihenfolge) in dem oben erstellten 2-3-4 Baum.

Geben Sie alle Zwischenschritte bei der Veränderung des Baums an.

c. [0.5] Beschreiben Sie die Eigenschaften eines 2-3-4 Baumes.

Aufgabe 3 [1.4]

Die Werte 5, 7, 8, 1, 6, 3, 2, 4 sind in dieser Reihenfolge in einem Vektor der Länge 8 gespeichert.

Sortieren Sie diesen Vektor jeweils mit dem Verfahren

a. [0.5] Heapsort,

b. [0.5] Countingsort

Geben Sie alle Zwischenschritte bei den Sortievorgängen an.

c. [0.4] Geben Sie für jeden der obigen Sortieralgorithmen eine Situation in der Praxis an, bei der es sinnvoll ist, gerade ihn einzusetzen.

Aufgabe 4 [2.5]

Die Werte 4, 14, 11, 18, 3, 12 und 1 sollen (in der gegebenen Reihenfolge) in eine ursprünglich leere *Hashtabelle* der Größe 7 eingetragen werden.

a. [0.2] Wählen Sie eine geeignete Hash-Funktion.

b. [0.7] Stellen Sie die Struktur der Hashtabelle dar, wenn Sie als Kollisionsfunktion *linear Probing* verwenden.

c. [0.3] Löschen Sie danach die Werte 4, 11 und 3 in der gegebene Reihenfolge aus der entstandenen Hashtabelle.

Zeigen Sie alle Zwischenschritte beim Eintragen und Entfernen anhand der Struktur der Hashtabelle.

d. [0.3] Geben Sie die Aufwände in O-Notation für die Hashtabellen-Operationen Einfügen, Löschen und Zugriff sowohl für das reine Hash-Verfahren als auch für die linear Probing Kollisionsmethode an.

e. [0.2] Geben Sie eine geeignete Datenstruktur in C++ zur Verwaltung einer *Hashtabelle mit dem linear Probing Kollisionsverfahren* an.

f. [0.8] Skizzieren Sie in Pseudocode einen Algorithmus zur Suche eines gespeicherten Wertes unter Berücksichtigung der Kollisionsbehandlung.

   Dokumentieren Sie Ihr Programm ausführlich.

Aufgabe 5 [1.6]

a. [0.7] Fügen Sie die Werte 8, 4, 3, 6, 7, 5, 1 und 2 in dieser Reihenfolge in einen ursprünglich binären Suchbaum ein.

b. [0.3] Löschen Sie die Werte 2, 4 und 8 in dieser Reihenfolge.

Stellen Sie alle Zwischenschritte bei der Veränderung des Baums dar.

c. [0.6] Traversieren Sie den entstandenen Baum mit der *Preorder, Inorder und Postorder* Methode und geben Sie die entsprechenden Reihenfolgen der Werte der besuchten Knoten aus.