Universität des Saarlandes
FR 6.2 Informatik
Prof. Dr. Hans-Peter Lenhof
Dipl. Inform. Jan Küntzer
Dipl. Inform. Alexander Rurainski

12. Übung zur Vorlesung Programmierung II, SS 2004

Hinweis: Dies ist das letzte Übungsblatt zur Vorlesung Programmierung II in diesem Semester. Gleichzeitig handelt es sich um eine Probeklausur, mit der Sie sich auf die Klausur am 23.07. vorbereiten können. Die Aufgaben die mit 0 Punkten versehen sind gehen nicht in die Wertung ein, werden aber trotzdem korrigiert. *Geben Sie diesmal jede Aufgabe auf einem eigenen Blatt ab, und versehen Sie jedes Blatt mit Ihrem Namen, Ihrer Matrikelnummer und dem Namen Ihres Übungsgruppenleiters! Geben Sie bei jeder Aufgabe bitte auch mit an, wieviel Zeit Sie zur Bearbeitung benötigt haben.*
Da Sie dieses Blatt unter Klausurbedingungen bearbeiten sollten, sollten Sie die C++-Implementierungsaufgaben diesmal nicht am Computer sondern nur auf dem Papier lösen.

Aufgabe 1: Wissenstest (0 Punkte)

(a) Erklären Sie *kurz* anhand eines `vector` das Konzept einer `C++` Template-Klasse.

(b) Definieren Sie $\Theta(n)$

(c) Geben Sie die Worst-Case Laufzeiten der Operationen *vorne anfügen* und *Suche* für eine *sortierte doppelt verkette Liste* und für ein *unsortiertes Array* an.

(d) Beschreiben Sie kurz drei Techniken zur Kollisionsauflösung bei Hashing mit *open adressing*.

(e) Definieren Sie den Begriff *Universelle Klasse von Hashfunktionen*.

(f) Was versteht man unter einem *Euler-Circuit*?

Aufgabe 2: Programmieraufgabe (10 Punkte)
Erstellen sie eine C++-Templateklasse `liste<T>`, die eine doppelt verkettete Liste implementiert. Sie dürfen hierfür keine STL-Container verwenden.

(a) Geben Sie dazu (nur) die Klassendeklaration (keine Implementierung) vollständig an. Die Klasse enthält: Konstruktor, Copykonstruktor, Destruktor, Zuweisungsoperator, Klassen- oder Strukturdeklaration des Knotentyps, Einfügen eines Elements am linken Ende, Bestimmung der Elementanzahl, Suche eines Elements und Löschen aller Vorkommen eines Elements.

(b) Geben Sie zusätzlich die Implementierung letzterer Methode `void loesche(const T& i)` an, welche alle Vorkommen von `i` in der Liste löscht.

Aufgabe 3. Klassendeklaration (10 Bonuspunkte)
Leiten Sie die Klasse `Quadrat` von der Basisklasse `Form`

```
enum Farbe {rot, violett, blau, gruen, gelb, orange};

class Form
{
public:

  // Konstruktor
  Form()
  {
    // tue nichts
  };

  // Destruktor
  ~Form()
  {
    // tue nichts
  }

  // liefert den Fl"acheninhalt
  virtual double flaecheninhalt() = 0;

  // liefert den Umfang
  virtual double umfang() = 0;

  // liefert die Farbe der Form
  Farbe farbe()
  {
    return farbwert;
  }

protected:

  // Farbe der Form
  Farbe farbwert;
};
```

ab und geben Sie sowohl die Klassendeklaration als auch die Implementierung an. Achten Sie auf einen sinnvollen Konstruktor (sinnvolle Argumente) und implementieren Sie die virtuellen Funktionen.

Aufgabe 4. $\mathcal{O}$-Notation (10 Bonuspunkte)
Finden Sie eine Anordnung der folgenden Funktionen, so dass gilt $f_i \in \mathcal{O}(f_{i+1}) \forall i$ und geben Sie für jedes Paar f_i, f_{i+1} eine kurze Begründung Ihrer Anordnung an.

(a) $n!$

(b) e^n

(c) $4^{\log_2(n+1)}$

(d) $n \ln(n)$

Aufgabe 5. Laplace Wahrscheinlichkeit (10 Bonuspunkte)
Das folgende Problem wurde bereits 1654 von Pascal und Fermat diskutiert:
Zwei Spieler spielen eine Folge von Münzwürfe, wobei unabhängig vom bisherigen Spielverlauf jedes Mal jeder der beiden Spieler mit Wahrscheinlichkeit $\frac{1}{2}$ gewinnt. Derjenige Spieler, der zuerst 10 Spiele gewonnen hat, soll den gesamten Einsatz gewinnen. Unerwarteterweise werden die beiden Spieler gezwungen, die Spielfolge nach 15 Spielen abzubrechen. Zu diesem Zeitpunkt hat der erste Spieler 8 Spiele gewonnen und der zweite Spieler dementsprechend 7. Die beiden Spieler einigen sich darauf den Einsatz so aufzuteilen, daß jeder Spieler einen Betrag proportional zu der Wahrscheinlichkeit erhält, daß er die gesamte Spielfolge gewinnen würde, wenn sie zu Ende gespielt wird. Wie ist der Betrag aufzuteilen?

Aufgabe 6: Substitutionsmethode (10 Punkte)
Lösen Sie die folgende Rekurrenzgleichung mit Hilfe der Substitutionsmethode, d.h., geben Sie ein $f(n)$ an mit: $T(n) \in \Theta(f(n))$. Beweisen Sie: $T(n) \in \mathcal{O}(f(n))$. Das auch gilt $T(n) \in \Omega(f(n))$, müssen Sie hier ausnahmsweise *nicht* beweisen!

$$T(n) = 2T\left(\frac{n}{3}\right) + n^3$$

Aufgabe 7. Mastertheorem (10 Bonuspunkte)
Zeigen Sie für jede der folgenden Rekurrenzen ob das Mastertheorem zur Lösung angewandt werden kann. Bestimmen Sie in den Fällen, in denen sich das Theorem anwenden lässt, die Lösung der Rekurrenz.

- $T(n) = 64T(\frac{n}{8}) + 1042n^2 + 1000000n$
- $T(n) = 27T(\frac{n}{3}) + n^3 \ln(n) + 42$

Aufgabe 8: Bäume (10 Punkte)
Ein 2–4 - Baum hat an jedem Knoten mindestens 2 und höchstens 4 Kinder. Ausserdem haben alle Pfade von der Wurzel zu einem beliebigen Blatt die gleiche Länge h. Die Schlüssel werden in einem 2–4 - Baum sortiert in den Blättern gespeichert. D.h. die inneren Knoten enthalten keinen Schlüssel! (Siehe Beispiel) Zwei Mengen A und B seien in den zugehörigen 2–4 - Bäumen T_A und T_B gespeichert. Beschreiben Sie einen Algorithmus, der den 2–4 - Baum $T_{A \cup B}$ der vereinigten Menge $A \cup B$ in Zeit $\mathcal{O}(|A| + |B|)$ berechnet. (Es reicht aus, wenn Sie Ihren Algorithmus in Pseudocode formulieren!) Begründen Sie Korrektheit und Laufzeit Ihres Algorithmus!
Hinweis: Zeigen Sie dazu zuerst, dass Sie den 2–4 - Baum einer schon sortierten Sequenz $a_1, \ldots, a_{n-1}$ in Zeit $\mathcal{O}(n)$ aufbauen können.

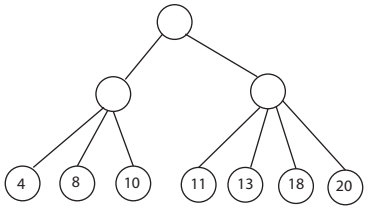

Abbildung 1: Beispiel eines 2–4 - Baums.

Aufgabe 9: Graphen (10 Punkte)
Zeigen Sie, dass jeder ungerichtete zusammenhängende Graph $G = (V, E)$ mindestens einen Knoten enthält, bei dessen Entfernung aus G der Graph zusammenhängend bleibt (wenn Sie den Knoten entfernen, entfernen Sie natürlich auch die zu ihm gehörigen Kanten aus dem Graphen). Entwickeln Sie einen Algorithmus, der einen solchen Knoten bestimmt und eine worst-case Laufzeit von $\mathcal{O}(|V| + |E|)$ aufweist (es reicht aus, wenn Sie diesen in Pseudocode formulieren!). Beweisen Sie die Korrektheit Ihres Verfahrens und begründen Sie kurz dessen Laufzeit.

Abgabe: 16.07.2004 (Vor der Vorlesung)