Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 2007

46114

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach: **Informatik (Unterrichtsfach)**

Einzelprüfung: **Algorithmen/Datenstrukturen/Programmiermethoden**

Anzahl der gestellten Themen (Aufgaben): **2**

Anzahl der Druckseiten dieser Vorlage: **6**

Bitte wenden!

Thema Nr. 1

Aufgabe 1

Eine Liste ist ein abstrakter Datentyp, der die Ablage einer Folge von Elementen gleichen Typs erlaubt und der die folgenden Operationen unterstützt:

- Einfügen eines Elements am Anfang oder an einer beliebigen Position der Liste
- Löschen eines Elements am Anfang oder an einer beliebigen Position der Liste
- Suchen eines Elements in der Liste

Wir betrachten im Folgenden Listen, in denen Integerwerte in aufsteigender Reihenfolge ihres Wertes sortiert abgelegt sind.
Zur Realisierung des abstrakten Datentyps betrachten wir doppelt verkettete Listen, die für jeden Integerwert ein separates Datenobjekt verwenden, welches mit dem Vorgänger- und dem Nachfolgerobjekt durch jeweils einen Verweis verbunden ist. Realisieren Sie die folgenden Aufgaben in C, C++ oder Java ohne Verwendung vordefinierter Klassen oder Bibliotheken zur Realisierung von Listen!

a) Deklarieren Sie einen Datentyp ELEM, der als Datenstruktur für eine doppelt verkettete Liste zur Ablage von Integerwerten verwendet werden kann!

b) Geben Sie eine Funktion `make_element(int value)` an, die ein Listenelement für eine doppelt verkettete Liste erzeugt, den als Parameter mitgelieferten Wert ablegt und einen Verweis auf das Listenelement als Rückgabewert zurückliefert!

c) Geben Sie eine Funktion `insert_element()` an, die einen als Parameter mitgelieferten Integerwert in eine doppelt verkettete, sortierte Liste so einfügt, dass die resultierende Liste wieder sortiert ist! Beachten Sie dabei auch die Spezialfälle, dass das einzufügende Element an den Anfang oder das Ende der Liste gehört! Ist der einzufügende Integerwert bereits in der Liste enthalten, soll er nicht erneut eingefügt werden. Eine Referenz auf die Liste, in die das Element eingefügt werden soll, soll ebenfalls als Parameter übergeben werden. Die Funktion `insert_element()` soll eine Referenz auf die neue Liste als Rückgabewert zurückliefern.

d) Geben Sie eine Funktion `delete_element()` an, die einen als Parameter mitgelieferten Integerwert aus einer doppelt verketteten, sortierten Liste löscht, falls der Integerwert in der Liste vorhanden ist!
Beachten Sie auch die Spezialfälle, dass das Element am Anfang oder am Ende der Liste gelöscht wird! Eine Referenz auf die Liste, aus der das Element gelöscht werden soll, soll ebenfalls als Parameter übergeben werden! Die Funktion `delete_element()` soll eine Referenz auf die neue Liste als Rückgabewert zurückliefern.

e) Geben Sie eine Funktion `compare_lists()` an, die zwei doppelt verkettete, sortierte Listen miteinander vergleicht! Die Funktion soll `true(1)` zurückliefern, falls die beiden Listen die gleichen Elemente in der gleichen Reihenfolge enthalten. Andernfalls soll `false(0)` zurückgeliefert werden. Referenzen auf die beiden Listen sollen als Parameter mitgeliefert werden.
Beachten Sie auch den Spezialfall, dass eine der beiden Listen leer ist!

Fortsetzung nächste Seite!

f) Geben Sie eine Funktion `merge_lists()` an, die zwei doppelt verkettete, sortierte Listen so mischt, dass die resultierende Liste ebenfalls sortiert ist und jedes in einer der beiden Eingabelisten enthaltene Element genau einmal enthält! Referenzen auf die beiden Eingabelisten sollen als Parameter mitgeliefert werden! Die neu entstehende Liste soll als Rückgabewert zurückgeliefert werden.

Aufgabe 2

Sortieren durch Mischen (Mergesort) ist ein Sortierverfahren, das nach dem divide-and-conquer-Prinzip arbeitet. Wir betrachten im Folgenden die Anwendung dieses Verfahrens zum Sortieren von Integerzahlen. Die Sortierung soll in aufsteigender Reihenfolge der Werte erfolgen. Wir nehmen dabei an, dass die zu sortierenden Zahlen in einer doppelt verketteten Liste abgelegt sind (vgl. Aufgabe 1).

a) Beschreiben Sie die Arbeitsweise des divide-and-conquer-Prinzips im allgemeinen Fall! Geben Sie dabei die Bedeutung der Schritte divide (Aufteilen), conquer (Lösung der Unterprobleme) und combine (Zusammensetzen) an!

b) Beschreiben Sie die Arbeitsweise des Algorithmus Mergesort. Geben Sie dabei an, worin die Schritte divide, conquer und combine im konkreten Fall bestehen!

c) Geben Sie eine Funktion `split()` in C, C++ oder Java an, die eine vorliegende (unsortierte) doppelt verkettete Liste von Integerzahlen so aufspaltet, dass dies dem divide-Schritt des Mergesort-Algorithmus entspricht!

d) Geben Sie eine Funktion `mergesort()` in C, C++ oder Java an, die in Form einer (unsortierten) doppelt verketteten Liste vorliegende Zahlen nach dem Prinzip von Mergesort sortiert! Verwenden Sie für das Aufspalten der Liste die Funktion `split()` aus Aufgabenteil c. Verwenden Sie für das Zusammenführen zweier sortierter Listen die Funktion `merge_lists()` aus Aufgabenteil f von Aufgabe 1. Formulieren Sie die Funktion `mergesort()` nach Vorgabe des divide-and-conquer-Prinzips als rekursive Funktion!

e) Identifizieren Sie in der von Ihnen im Aufgabenteil d realisierten Funktion die Schritte divide, conquer und combine. Schätzen Sie anhand der von Ihnen realisierten Funktion die asymptotische Laufzeit dieser Schritte ab, indem Sie die O-Notation verwenden!
Leiten Sie aus der Laufzeit der Einzelschritte die Laufzeit des Gesamtverfahrens ab! Begründen Sie Ihre Antwort!

Aufgabe 3

Ein Keller ist ein abstrakter Datentyp zur Ablage von Elementen des gleichen Typs, wobei ein Zugriff nur an einem Ende des Kellers möglich ist. Auf einem Keller sind folgende Operationen vorgesehen:

- `push(e)` : Einfügen eines Elements `e` am freien Ende des Kellers
- `e=pop()` : Entfernen eines Elements vom freien Ende des Kellers

Fortsetzung nächste Seite!

Wir betrachten im Folgenden einen Keller zur Ablage von Integerwerten. Für die Realisierung des Datentyps verwenden wir eine doppelt verkettete Liste. Verwenden Sie zur Realisierung der folgenden Aufgabenpunkte C, C++ oder Java ohne Verwendung vordefinierter Klassen oder Bibliotheken!

a) Geben Sie eine Funktion `push()` zur Ablage eines als Parameter mitgelieferten Elements am freien Ende des Kellers an!

b) Geben Sie eine Funktion `pop()` zur Entnahme des obersten Kellerelements an; dieses soll als Rückgabewert zurückgeliefert werden!

c) Geben Sie eine Funktion `count()` an, die die Anzahl der in dem Keller aktuell abgelegten Werte zählt!

Thema Nr. 2

Allgemeines zu den Aufgaben

Entwerfen und programmieren Sie ein Telefonverzeichnis, d. h. eine Liste mit einzelnen Einträgen, die Name, Vorname, Telefonvorwahl und Telefonnummer enthalten! Verwenden Sie als Programmier-Modell OOP („Objekt orientierte Programmierung") (z. B. Java, C++). In jedem Fall sollte Ihr Programm modular aufgebaut sein!

Entwerfen Sie zunächst in Aufgabe 1 eine Spezifikation für diese Aufgabe und begründen Sie dabei die einzelnen Entscheidungen! Formulieren Sie dann in Aufgabe 2 Ihren Entwurf in einer Ihnen geläufigen Programmiersprache, wobei Sie sich nicht exakt an die Syntax der verwendeten Programmiersprache halten müssen! Sie dürfen dabei, soweit es notwendig ist, elementare Hilfsfunktionen (etwa für die Ein-/Ausgabe) als vorhanden annehmen.

Aufgabe 1 (Entwurf)

Das Telefonverzeichnis soll in jedem Fall die beiden folgenden Klassen (Module) enthalten: Eine Klasse **„Telefoneintrag"**, die einen einzelnen Telefoneintrag beschreibt und eine zweite Klasse **„Liste"**, die eine Listenverwaltung realisiert, aber so allgemein, dass sie auch andere Objekte verwalten könnte (z. B. Bücher). Entwerfen Sie zusätzlich ein Hauptprogramm, mit dem Sie ein Telefonverzeichnis bearbeiten könnten! (Nur skizzieren!)

1.1 Klasse „Telefonbucheintrag"

1.1.1 Die Klasse **TelefonEintrag**. soll folgende Eigenschaften (Einträge) haben:

- Familienname
- Vorname
- Vorwahl
- Telefonnummer

Welche Datentypen verwenden Sie für die einzelnen Eigenschaften? Begründen Sie Ihre Entscheidung!

Fortsetzung nächste Seite!

1.1.2 Skizzieren Sie folgende Methoden (Funktionen/Prozeduren) für die Klasse Telefoneintrag: (Ein neues Objekt muss mindestens Name und Vorname enthalten!)

- setTelefonNummer // soll Vorwahl und Telefonnummer in das Objekt eintragen.
- getTelefonNummer // liefert Vorwahl und Telefonnummer des Objekts.
- vergleicheMit // soll zwei Objekte nach Name und Vornamen miteinander.
  // lexikographisch vergleichen und größer, kleiner, gleich liefern.

(Beim Vergleich soll ein Eintrag als größer als ein anderer gelten, wenn er in einem üblichen Telefonbuch hinter diesem stehen würde.)

1.2 Listen

1.2.1 Beschreiben Sie zunächst verschiedene Datenstrukturen, mit denen man Listen implementieren kann! Diskutieren Sie Vor- und Nachteile der verschiedenen Möglichkeiten!

1.3 Klasse Liste

1.3.1 Skizzieren Sie eine **Klasse „Liste"**, die folgende Methoden zur Verfügung stellt:

- eintragen // trägt ein Objekt in die Liste ein.
  // Die Liste darf keine zwei „gleichen" Einträge enthalten.
- loeschen // löscht ein Objekt aus der Liste.
- suchen // sucht ein Objekt in der Liste. Verwenden Sie dabei zum
  // Vergleichen die in der Klasse der verwalteten Objekte
  // angebotene Methode „**vergleicheMit**".
- eintrag_aendern // damit kann eine Telefonnummer geändert werden.
  // Diese Methode muss so allgemein sein, dass sie auch
  // für Listeneinträge einer anderen Klasse funktioniert!

Aufgabe 2: (Programmierung)

2.1 Programmieren Sie die Klasse Telefoneintrag!

2.2 Programmieren Sie die Klasse Liste!

2.3 Skizzieren Sie ein Hauptprogramm, mit dem Sie ein Telefonverzeichnis anlegen und bearbeiten können, mit den folgenden Funktionen:
- Eintragen, - Eintrag löschen, - Telefonnummer suchen, - Telefonnummer ändern.
Dazu können Sie beliebige Ein-/Ausgabe-Methoden aufrufen, mit denen Sie Werte von der Tastatur einlesen und in einem Bildschirmfenster ausgeben können. Zusätzlich stehen Ihnen nach Belieben „Befehlsschalter" zur Verfügung.

Fortsetzung nächste Seite!

Aufgabe 3: (Systementwurf)

Ein Programm zum Lesen und Bearbeiten eines Telefonbuches soll im Allgemeinen ja von mehreren Personen gleichzeitig benutzt werden können.

3.1 Beschreiben Sie die Probleme, die bezüglich der Datenintegrität auftreten können, wenn mehrere Personen berechtigt sind, das Telefonbuch zu ändern!

3.2 Skizzieren Sie eine Lösung für dieses Problem!