FernUniversität in Hagen

Fakultät für Wirtschaftswissenschaft

Matrikelnummer _____

Name _____

Vorname _____

Klausur: Entwurf und Implementierung von Informationssystemen (32561)
Termin: 07.09.2018, 09:00–11:00 Uhr
Prüfer: Univ.-Prof. Dr. rer. pol. habil. S. Strecker

Aufbau und Bewertung der Klausur

Aufgabe	1 (OE)	2 (A&D)	3 (PiC)	Summe
Maximal erreichbare Punktzahl	**20**	**40**	**40**	100
Erreichte Punktzahl				

Datum:

Note:

Allgemeine Hinweise

Tragen Sie bitte jetzt Ihre **Matrikelnummer**, Ihren **Namen** und **Vornamen** auf dem **Deckblatt** ein. Versehen Sie bitte zusätzlich **jede Seite** mit Ihrer **Matrikelnummer**.

Hinweise zur Bearbeitung

Für die Bearbeitung der insgesamt drei Klausuraufgaben auf den folgenden 16 Seiten dieser Klausur stehen Ihnen 120 Minuten zur Verfügung.

1. Außer Schreibgeräten sind keine Hilfsmittel zugelassen.
2. Die Lösungen müssen in den vorgesehenen Raum auf den Aufgabenblättern eingetragen werden.
3. Notizen können auf den Rückseiten der Aufgabenblätter gemacht werden. Diese Anmerkungen werden in die Bewertung nicht einbezogen.
4. Bei Beendigung der Klausur müssen das Deckblatt und die Aufgabenblätter abgegeben werden. Trennen Sie bitte nicht einzelne Blätter ab.

Viel Erfolg!

Aufgabe 1 (Objektorientierter Entwurf) 20P

a. Erläutern Sie in eigenen Worten, was unter den im Lehrbrief dargestellten Begriffen der »Klasse«, »Datenkapselung« und »Datenabstraktion« zu verstehen ist. Erörtern Sie dazu zunächst den Zweck und anschließend die Zusammenhänge zwischen den Begriffen. (8P)

b. Nennen Sie zwei Einflussfaktoren und zwei grundlegende Entscheidungen des objektorientierten Entwurfs betrieblicher Anwendungssysteme. (2P)

c. Geben Sie an, ob die nachfolgend aufgeführten Aussagen zutreffen oder nicht. Tragen Sie hierzu jeweils in dem vorgegebenen Kreis ein »**R**« für richtig oder ein »**F**« für falsch ein. Für diese Aufgabe gibt es maximal 10 Punkte. Für jede richtige Antwort erhalten Sie 1 Punkt. (10P)

Bei einer Schichten-Architektur gestattet die Separierung des fachlichen Kerns einer Anwendung in einer Fachkonzept-Schicht die kontinuierliche Weiterentwicklung des OOA-Modells zum OOD-Modell der Fachkonzept-Schicht.	○
Eine Assoziation modelliert eine Gesamtheit von direkten Objektverbindungen. Assoziationen bilden die Grundlage der Kooperation von Objekten verschiedener Klassen.	○
Schnelle Algorithmen basieren auf geeignet gewählten Datenstrukturen, z. B. ermöglichen lineare Listen eine viel raschere Suche von Elementen als binäre Suchbäume.	○
Gilt für die Schichten einer Anwendung das Prinzip der asymmetrischen Kooperation und der unidirektionalen Abhängigkeit, so gilt dies auch für die Komponenten innerhalb einer Schicht untereinander.	○
Eine Assoziation modelliert eine Gesamtheit von direkten Objektverbindungen zwischen zwei Objekten. Assoziationen bilden die Grundlage der Kooperation von Objekten verschiedener Klassen.	○
Eine zentrale Aufgabe des objektorientierten Entwurfs einer Anwendung ist die Spezifikation ihrer Architektur.	○
Durch die Nutzung der Vererbung in den Topologien für Klassenbibliotheken, wie z. B. der Baum- und der Waldtopologie, wird die Laufzeit verkürzt.	○
Von einer Wald-Topologie wird gesprochen, wenn eine Klassenbibliothek in mehrere Vererbungshierarchien zerfällt, wobei jede Hierarchie dabei mehrere logisch zusammenhängende Klassen zu einer abhängigen Komponente bündelt.	○
Innerhalb eines Prozesses können mehrere *Threads* ablaufen, die üblicherweise jeweils auf einen separaten Speicherbereich zugreifen.	○
Dynamischer Polymorphismus kann die Laufzeit im Vergleich zum statischen Binden in gewissem Umfang erhöhen.	○

Aufgabe 2 (Algorithmen und Datenstrukturen) 40P

1) In der gesamten Aufgabe 2 wird von Ihnen erwartet, dass Sie die im Lehrbrief dargestellte, an PASCAL angelehnte Pseudocode-Notation ausnahmslos anwenden. Für das Algorithmieren mit diesem Pseudocode stehen damit die spezifischen Konzepte von PASCAL zur Verfügung, nämlich verschiedene einfache und zusammengesetzte Datentypen, Konstrukte der strukturierten Programmierung und das Prozedurkonzept. Alle Teilaufgaben sind als Codefragmente in der im Lehrbrief dargestellten Pseudocode-Notation zu erstellen. Andere Pseudocode-Notationen oder Programmiersprachen werden *nicht* bewertet.

a. Entwickeln Sie eine rekursive Funktion zur Berechnung der n-ten Fibonacci-Zahl f_n mit $n \geq 0$, die auf der rekursiven Definition der Zahl basiert: (6P)

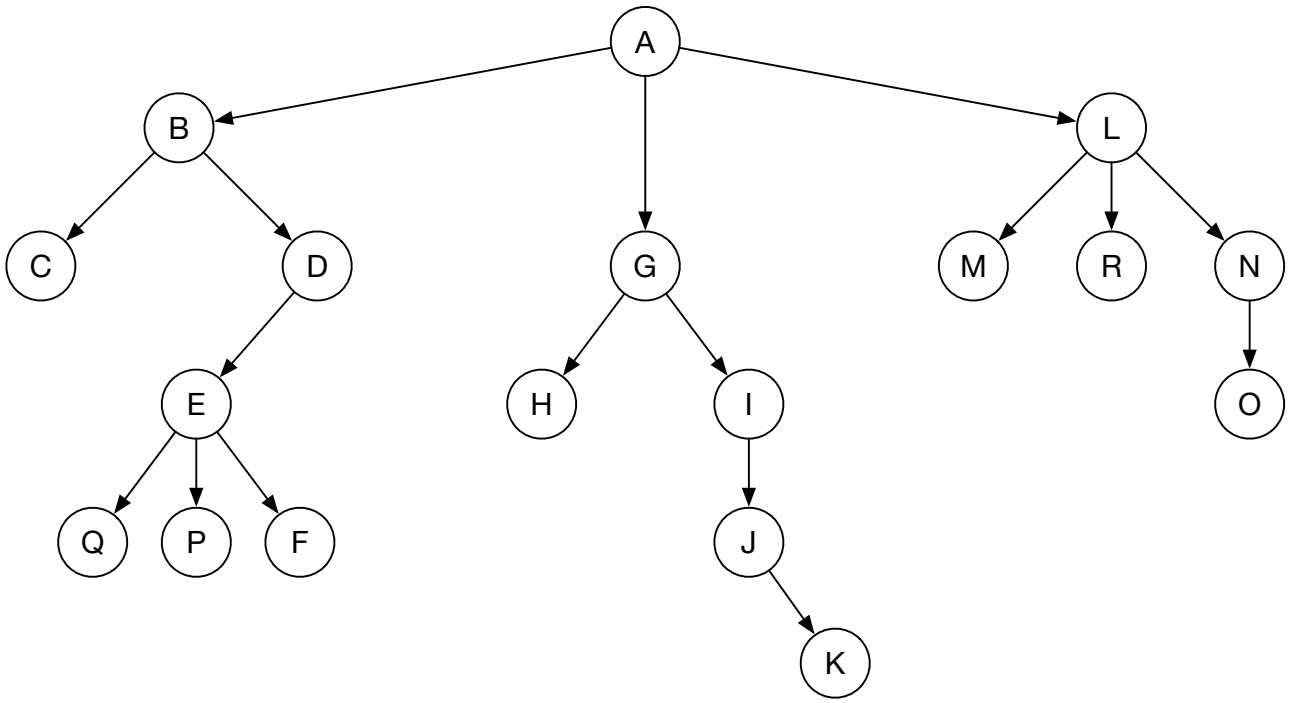
$$f_1 = f_0 = 1$$
$$f_n = f_{n-1} + f_{n-2} \quad \text{für} \quad n \geq 2$$

Damit lauten die ersten Zahlen: $1, 2, 3, 5, 8, 13, 21, \ldots$

b. Schätzen Sie die Anzahl der Funktionsaufrufe bei der Berechnung der n-ten Fibonacci-Zahl aus Teilaufgabenteil (1a) grob ab und ziehen Sie eine Schlussfolgerung hinsichtlich des Rechenzeitverhaltens der rekursiven Berechnungsfunktion. (5P)

2) Mit dem Begriff »Traversieren« wird das Durchlaufen sämtlicher Knoten eines Baumes in einer bestimmten Reihenfolge bezeichnet. In der Regel wird mit dem Traversieren die Bearbeitung vieler oder aller Knoten bzw. Datenobjekte verbunden sein. Für Binärbäume eignen sich insbes. auch rekursive Traversierungsalgorithmen.

a. Wandeln Sie den nachfolgend dargestellten k-nären Baum, mit einer Ordnung von $k = 3$, in einen Binärbaum um. Anschließend sollen Sie die Reihenfolge des Ansprechens der Knoten für den Fall des Traversierens in *Symmetrischer Ordnung* angeben. (8P)



3) Ein Online-Supermarkt stellt bezüglich der Belieferung von Kunden infolge der Atomisierung von Warensendungen steigende Transportkosten im Verhältnis zum Auftragsvolumen fest. Um die Transportkosten wieder zu senken, wurde ein Kunden-zur-Ware-Konzept entwickelt, das wie folgt aussieht:

- Es werden so genannte Pick-Up-Stellen (PUS) eingerichtet, in denen paketförmige Sendungen auf Stellplätzen gelagert werden können. Eine PUS umfasst jeweils ungekühlte, gekühlte und tiefgekühlte Stellplätze.
- Nach einer Bestellung beim Online-Supermarkt wird die Auslieferung der entsprechenden Sendung zu einer PUS in der Nähe des Kundenwohnortes veranlasst. Die Sendungen werden in Kisten verpackt und in LKW-Container geladen. Dann wird die Sendung auf einem Stellplatz, der den jeweiligen Kühlanforderungen entspricht, abgelegt. Unmittelbar danach wird der Kunde per E-Mail über das Eintreffen der Ware informiert, wobei alle für das Abholen der Sendung erforderlichen Daten übermittelt werden.
- Der Kunde hat nun 24 Stunden Zeit, die Sendung von der PUS abzuholen. Danach wird sie von der PUS zurückbefördert, wobei dem Kunden entstandene Kosten in Rechnung gestellt werden.

Um die Gesamtanzahl der Stellplätze einer PUS und ihre Verteilung auf die oben genannten Kühltypen geeignet bestimmen zu können, sollen das Eintreffen und Abholen von Sendungen computergestützt simuliert werden. Dabei sind die Kundenbestellungen in einer linearen Liste (Bestell-Liste) zu verwalten, die durch folgende Typen definiert ist:

```
TYPE KUEHLTYP = (ungekuehlt, gekuehlt, tiefgekuehlt);
TYPE STATUS = (bestellt, aufgenommenInPus, abgeholtVonPus);
TYPE LISTEZGR = ↑LISTEELEM;

TYPE LISTEELEM = RECORD
  BestNr: INTEGER;        {Bestellnummer, > 0}
  Kuehltyp: KUEHLTYP;     {erforderlicher Kühltyp in der PUS}
  BestZeit: INTEGER;      {Zeitpunkt der Aufgabe der Bestellung}
  AufnahmeZeit: INTEGER; {ggf. Zeitpunkt der Aufnahme in PUS}
  AbholZeit: INTEGER;     {ggf. Zeitpunkt der Abholung von PUS}
  BestStatus: STATUS;     {Status der Bestellung, siehe Statustyp}
  next: LISTEZGR;         {Zeiger auf folgendes Listenelement}
END;
```

Bei der Lösung der folgenden Teilaufgaben sind die angegebenen Hinweise zu beachten. Für beide Teilaufgaben ist die parameterlose Funktion `GetDelay()` zu nutzen. Diese gibt bei jedem Aufruf eine positive ganzzahlige Zeitspanne zurück. Es sollen keine globalen Variablen verwendet werden.

a. Formulieren Sie eine Funktion `lagereSendung()` mit dem angegebenen Funktionskopf. Der Zeiger `first` verweist auf das erste Element der Bestellliste. Die Funktion soll zuerst prüfen, ob sich bereits eine Bestellung mit der Nummer `Nr` in der Bestell-Liste befindet und gibt in diesem Fall den Wert **FALSE** zurück. Andernfalls wird eine neue Bestellung mit den übergebenen Bestelldaten und dem Status »bestellt« erzeugt. Besitzt der Parameter `full` nicht den Wert **TRUE**, so kann die Sendung in die PUS aufgenommen werden. In diesem Fall sind der Status der Bestellung zu aktualisieren und die Aufnahmezeit der Sendung zu ermitteln, indem eine mittels der Funktion `GetDelay()` erhaltene Zeitspanne zur Bestellzeit

addiert wird. Wurde eine Bestellung erzeugt, so ist sie am Ende der Bestell-Liste einzufügen und der Wert **TRUE** ist zurückzugeben. (14P)

```
FUNCTION lagereSendung(Nr: INTEGER; BestZeit: INTEGER; Kuehltyp: KUEHLTYP
    ; full: BOOLEAN; VARIABLE first: LISTEZGR): BOOLEAN;
```

```
END lagereSendung;
```

b. In der obigen Teilaufgabe (3a), werden Daten in einer *linearen Liste* gespeichert und verarbeitet. Eine weitere mögliche Datenstruktur, die zum Verwalten von Datenobjekten dient, ist der Stapel. Warum wird der Stapel auch als *Kellerspeicher* bezeichnet? Erläutern Sie die *Arbeitsweise* des Stapels. Angenommen in Ihrem Anwendungssystem treten *mehrere* Stapel auf. Durch welche *Art des Speicherns* könnten Sie Speicherplatz sparen? (7P)

Aufgabe 3 (Programmieren in C) 40P

a. Eine mögliche »Divide & Conquer«-Vorgehensweise für die Suche nach einem bestimmten Wert in einem *aufsteigend* sortierten Feld der Länge *n* ist die *binäre Suche* (engl. »binary search«). Entwerfen Sie eine Funktion, welche diese Vorgehensweise abbildet. Ihnen stehen folgende Hinweise zur Verfügung:

   1. Vergleichen Sie den gesuchten Wert mit dem in der Mitte des Feldes.
   2. Bei Gleichheit: $n/2$ ist die gesuchte Stelle. Es folgt ein Abbruch des Programms.
   3. Andernfalls muss überprüft werden, ob der gesuchte Wert größer ist. Wiederholen Sie das Verfahren für das Teilfeld von rechts der Feldmitte.

   Weiterhin steht Ihnen folgender Prototyp zur Verfügung: (6P)

```
int binaer_suche(int f[], int untere, int obere, int werte)

{

}
```

b. Entwickeln Sie ein C-Programm, welches zwei eingegebene Zeichenketten vertauscht. Im ersten Schritt entwerfen Sie dazu eine Funktion mit dem Prototypen `tausche_string(char *s1, char *s2)`, welche die eingegebenen Zeichenketten tauscht. Anschließend soll die Hauptroutine des C-Programmes Zeichenketten einlesen und vertauschen. Geben Sie abschließend die getauschten Zeichenketten aus. (8P)

```
#include <stdio.h>
#include <string.h>

{
```

```

```

```
}
```

c. Schreiben Sie folgende `for`-Schleife als `while`-Schleife mit gleicher Bedeutung. (5P)

```
int summand, summe;
for(summand=10, summe=0; summand>0; summand--)
{
  printf("\nZwischensumme:%d",summe);
  summe += summand;
  summand--;
}
```

```
{
```

```
}
```

d. Nennen Sie drei Typen verketteter Datenstrukturen und erläutern Sie kurz, wie die Knoten eines Baumes mit »Vorgängern« und »Nachfolgern« zusammenhängen. Orientieren Sie sich an den im Lehrbrief dargestellten Typen *verketteter Datenstrukturen*. (6P)

e. Das folgende C-Programm soll eigentlich in Abhängigkeit von einer eingelesenen Zahl $n \in \mathrm{N}$ alle natürlichen Zweierpotenzen $\leq n$ ausgeben. Warum funktioniert das C-Programm nicht bzw. welche Ausgabe erwarten Sie? (5P)

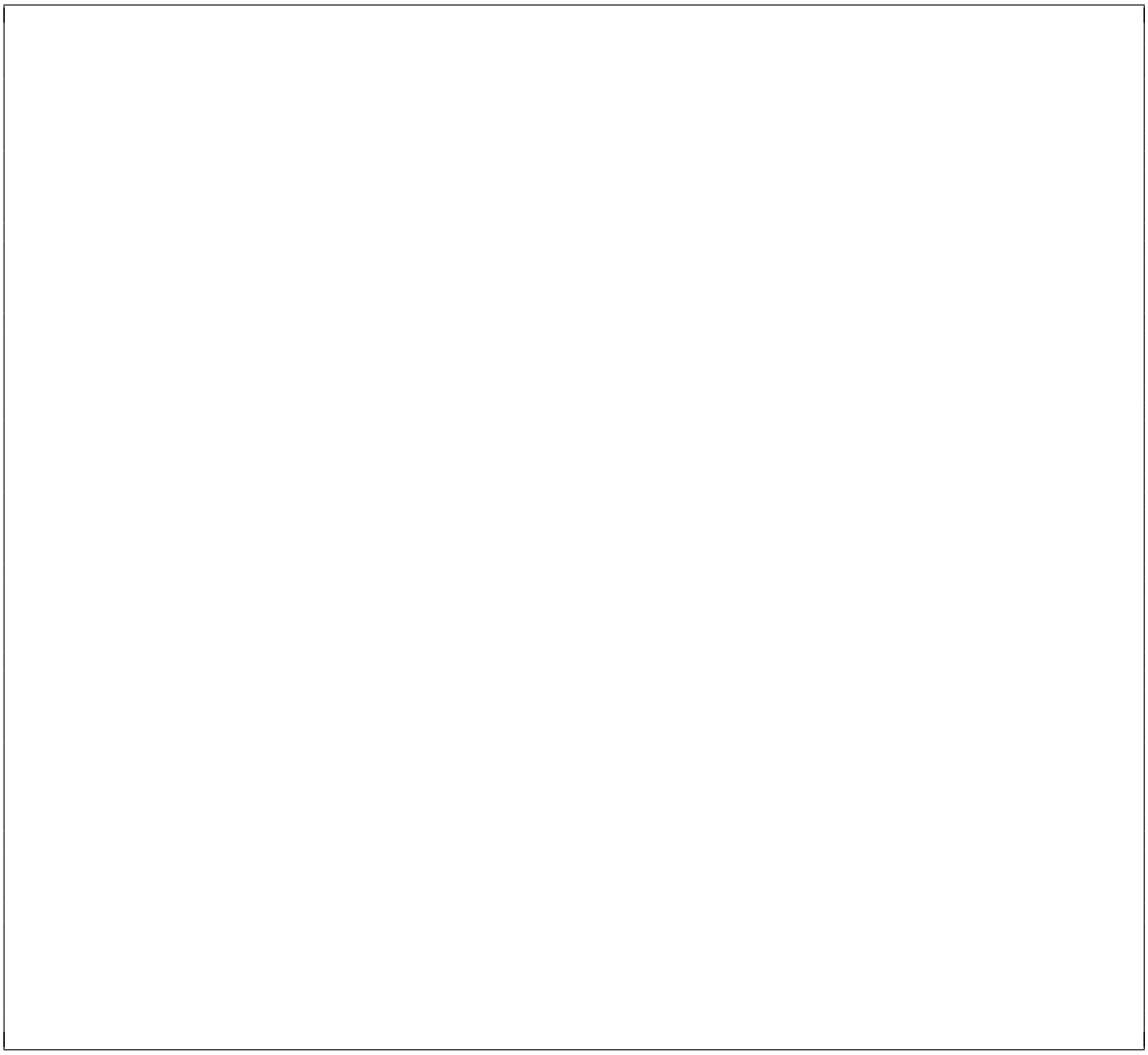
```
#include <stdio.h>

int main()
{
   int i, n;
   scanf("%i", &n);
   if (n>0)
    for (i=0; i<=n; i*=2)
      printf("%i\n", i);
}
```

f. Betrachtet sei ein Sortieralgorithmus für sequentiell gespeicherte Objekte. Die Tabelle unten zeigt die Schritte eines solchen Algorithmus sowie die Ausgangs- und Zielfolge. Benennen Sie den Algorithmus und erläutern Sie kurz den Sortiervorgang. Entwickeln Sie für den dargestellten Sortieralgorithmus eine C-Funktion. (10P)

Ausgangszustand:	42	51	13	48	92	16	9	68
1. Schritt	42	51	13	48	92	16	9	68
2. Schritt	42	51	13	48	92	16	9	68
3. Schritt	13	42	51	48	92	16	9	68
4. Schritt	13	42	48	51	92	16	9	68
5. Schritt	13	42	48	51	92	16	9	68
6. Schritt	13	16	42	48	51	92	9	68
7. Schritt	9	13	16	42	48	51	92	68
8. Schritt	9	13	16	42	48	51	68	68

Erläuterungen und Benennung:

```c
#include <stdio.h>
#include <string.h>

int main(void) {
   int i;

   /* Array zum Sortieren */
   int array[] = { 42, 51, 13, 48, 92, 16, 9, 68 };

   int N = sizeof(array)/sizeof(int);

   sort(array, N-1);

   for(i = 0; i < N; i++)
      printf("%d ", array[i]);
   printf("\n");
}

void sort(int *array, int elemente) {
```



```c
}
```