FernUniversität in Hagen

Fakultät für Wirtschaftswissenschaft

Matrikelnummer _____

Name _____

Vorname _____

Klausur: Entwurf und Implementierung von Informationssystemen (32561)
Termin: 08.03.2019, 09:00–11:00 Uhr
Prüferin: Dr. K. Rosenthal

Aufbau und Bewertung der Klausur

Aufgabe	1 (OE)	2 (A&D)	3 (PiC)	Summe
Maximal erreichbare Punktzahl	**20**	**42**	**38**	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 17 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 vorteilhafte Merkmale des objektorientierten Entwurfs sind. Nennen Sie anschließend Komponentenbeziehungen, die im objektorientierten Entwurf eine wesentliche Rolle spielen. Beachten Sie, dass im objektorientierten Entwurf die objektorientierten Konzepte der objektorientierten Analyse weiterhin verwendet werden. Orientieren Sie sich an den im Lehrbrief dargestellten Ausführungen zum »objektorientierten Entwurf (OOD)«. (10P)

b. 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)

Die Anwendung des Konzepts »Polymorphismus/Polymorphie« der objektorientierten Programmierung ermöglicht, dass ein Bezeichner abhängig von seiner Verwendung Objekte unterschiedlichen Datentyps annimmt. ○

Klassenbibliotheken mit Baum-Topologien nutzen das Vererbungskonzept, stellen jedoch keine Vererbungshierarchie dar. ○

Eine Baustein-Topologie liegt vor, wenn eine Klassenbibliothek aus lauter abhängigen Klassen besteht, die über Vererbungsbeziehungen verbunden sind. ○

Das Konzept des »dynamischen Polymorphismus« ermöglicht, dass Objekte verschiedener Klassen einer Vererbungshierarchie formal einheitlich verarbeitet werden. ○

Die Einsparung der Versendung von Botschaften stellt ein entscheidendes Mittel der Erhöhung der Laufzeiteffizienz einer Anwendung dar. ○

Neben der einfachen Vererbung mit nur einer Oberklasse kann im objektorientierten Entwurf auch die Mehrfachvererbung verwendet werden, bei der abgeleitete Klassen mehrere Oberklassen besitzen. ○

In einer Fünf-Schichten-Architektur übernimmt die Fachkonzept-Zugriffsschicht Aufgaben der Datenkonvertierung, um die GUI-Schicht hiervon zu entlasten. ○

Während im OOA-Modell die üblichen Verwaltungsoperationen zum Erfassen, *Ändern* und *Löschen* von Objekten, den Fachkonzept-Klassen zugeordnet sind, muss im Entwurf zwischen GUI- und Fachkonzept-Schicht unterschieden werden. ○

Schnelle Algorithmen basieren vielfach auf geeignet gewahlten Datenstrukturen: Daher ermoglichen binare Suchbäume eine viel raschere Suche von Elementen als lineare Listen ○

Eine zentrale Aufgabe des objektorientierten Entwurfs einer Anwendung ist die Spezifikation ihrer Architektur. ○

Aufgabe 2 (Algorithmen und Datenstrukturen) 42P

1) In der gesamten Aufgabe 2 wird von Ihnen erwartet, dass Sie die im Lehrbrief dargestellte, an PASCAL angelehnte Pseudocode-Notation ausnahmslos verwenden. 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* gewertet.

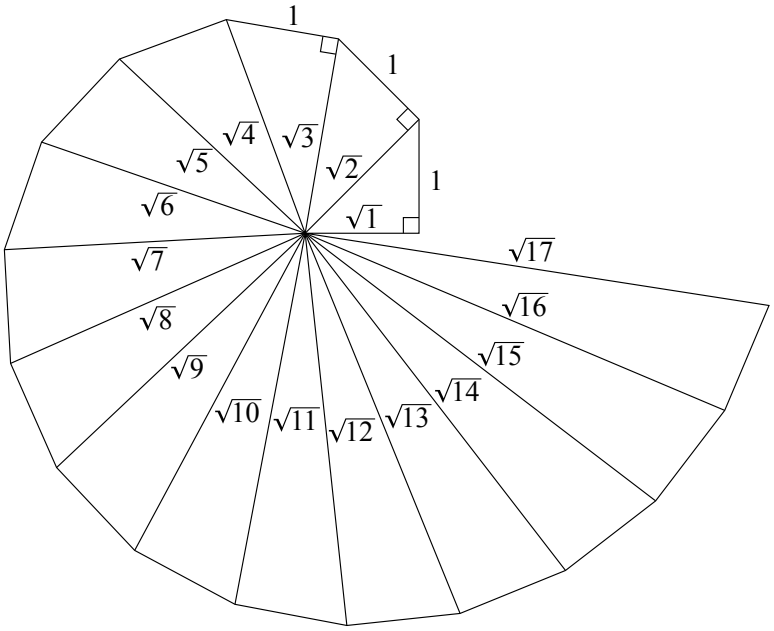


Abbildung 1: Das Rad des Theodorus (geometrische Abbildung).

Das Rad des Theodorus ist eines der ersten Beispiele einer Rekursion (s. Abb. 1). Um etwa 465 v. Chr. hat Theodorus von Kyrene gezeigt, dass nicht nur die Quadratwurzel aus 2, sondern auch die Quadratwurzel aus den nichtquadratischen natürlichen Zahlen von 3 bis 17 irrational sind. Dabei ging Theodorus geometrisch vor, wobei es bis heute keine Überlieferung seiner Beweisführung gibt und ebenfalls angezweifelt wird, ob er den Beweis tatsächlich durchgeführt hat. Der zu entwickelnde Algorithmus kann nach folgender Bildungsregel rekursiv angewandt werden: (i) D_1 ist ein rechtwinkliges, gleichschenkliges Dreieck mit Schenkellänge l; (ii) D_n ist ein rechtwinkliges Dreick, wobei ein Schenkel gleich der Hypotenuse von D_{n-1} ist und der andere Schenkel die Länge 1 hat. Die Länge der Hypotenuse c_n von D_n ist somit rekursiv definiert und kann folgendermaßen notiert werden:

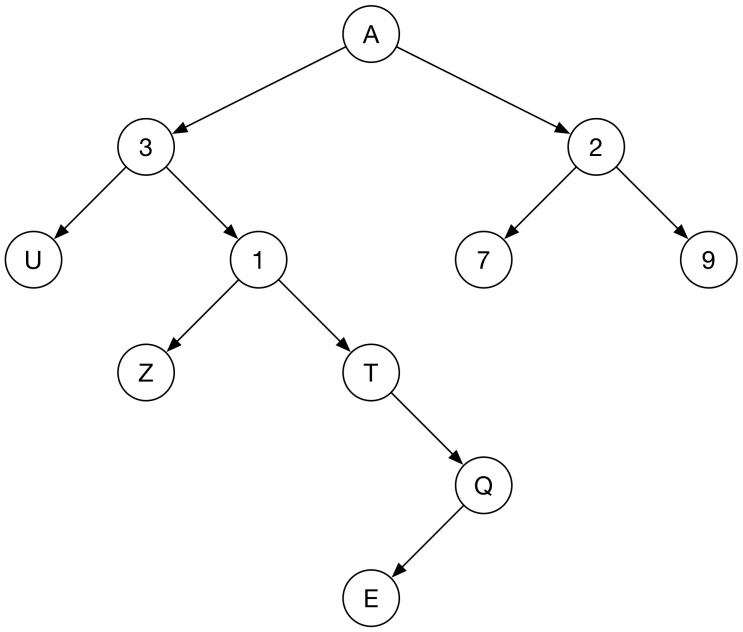
$$c_1 = \sqrt{2}$$
$$c_n = \sqrt{c_{n-1}^2 + 1}$$

a. Entwerfen Sie ein C-Programm, welches in Abhängigkeit einer eingegebenen natürlichen Zahl n im Intervall I (mit $I := \{n \in \mathbb{N} \mid 1 \leq n \leq 17\}$) die Länge der Hypotenuse c_n rekursiv berechnet. Hinweis: Die Wurzel kann mit der Funktion `SQRT(...)` berechnet werden. (6P)

b. Erläutern Sie jeweils stichpunktartig, welche Vor- bzw. Nachteile mit der Verwendung einer Rekursion gegenüber einer Iteration einhergehen, was unter einer »Rekursion« zu verstehen ist und an welche Bedingungen ein rekursiver Aufruf einer Prozedur geknüpft ist. (4P)

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. Geben Sie die Ausgabe des Traversierens des dargestellten Binärbaumes in (1) symmetrischer Ordnung, (2) Präordnung und (3) Postordnung an und beschreiben Sie kurz Ihre Vorgehensweise. (6P)



3) Bei der sequentiellen Speicherung einer Schlange (engl. »Queue«) belegen die verwalteten Objekte einen zusammenhängenden Bereich eines Feldes. Je ein Index markiert das `front`- und das `rear`-Element. Im Folgenden sollen Prozeduren zum Einfügen, Entfernen und Lesen von Auftragsdaten entworfen werden. Hinweis: Die Aufträge werden tatsächlich in der Reihenfolge ihres Eingangs abgearbeitet, sodass die Organisation über eine Liste sinnvoll ist. In der Schlange können maximal 200 Aufträge abgelegt werden. Die aktuell abgelegte Anzahl wird mittels der Variablen `entries` (Einträge) festgehalten. Als Ausgangsbasis dieser Aufgabe dient Ihnen die folgende Datendefinition.

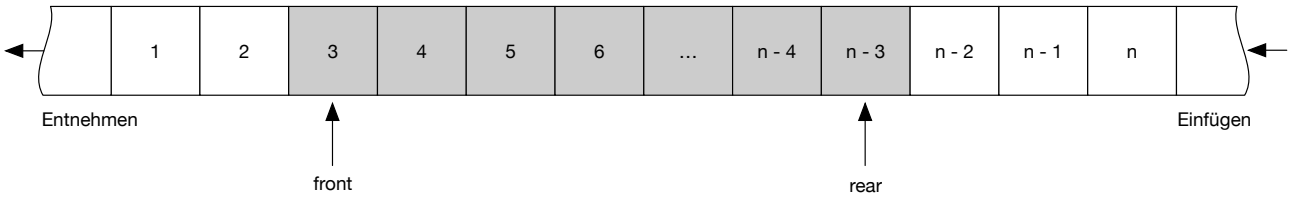


Abbildung 2: Sequentielle Speicherung einer Schlange.

```
DATA
 CONST n = 200;
  TYPE
   ORDER = RECORD
    ordnerno : ARRAY [1..8] OF CHAR
    company : ARRAY [1..8] OF CHAR
    reviser : ARRAY [1..8] OF CHAR
    END;
   QUEUE = ARRAY [1..n] OF ORDER;
   INDEX = [0..n];
  VARIABLE
   qqueue : QUEUE;
   front, rear : INDEX;
   entries : INTEGER;
   newelement, frontelement : ORDER;
   full, empty : BOOLEAN;
```

a. Entwerfen Sie eine Prozedur `append(..)` (Einfügen). Dafür steht Ihnen folgender Programmquellcode bereits zur Verfügung. Der einzufügende Auftrag soll mit `newelement` übergeben werden. Ein Einfügen ist nur möglich, wenn die Schlange weniger als n Objekte enthält. Weist jedoch bei nicht voller Schlange der Index `rear` auf das n-te Element, so müssen sämtliche Objekte erst zur `front`-Seite hin verschoben werden, bevor das Einfügen erfolgen kann. (8P)

```
PROCEDURE append ( newelement : ORDER;
                   VARIABLE front, rear : INDEX;
                   VARIABLE entries : INTEGER;
                   VARIABLE full : BOOLEAN;
                   VARIABLE qqueue : QUEUE);
  DATA
   VARIABLE i, length : INDEX;
```

b. Entwerfen Sie die Prozedur `remove(..)` (Entfernen) zum Entfernen eines Elements in der Schlange. Eine Entnahme ist nur bei einer nichtleeren Schlange möglich. Nach der Entnahme eines Elements wird der Index `front` auf das neue `front`-Element umdirigiert und die Anzahl der Einträge wird um 1 reduziert. (5P)

```
PROCEDURE remove ( VARIABLE frontelement : ORDER;
                   VARIABLE front : INDEX;
                   VARIABLE entries : INTEGER;
                   VARIABLE empty : BOOLEAN;
                   VARIABLE qqueue : QUEUE);
```

c. Entwerfen Sie eine Prozedur **read**(..) zum Auslesen des front-Elements. Beachten Sie, dass diese Prozedur lediglich dem Lesen des front-Elements und nicht dem Entfernen dienen soll. (5P)

```
PROCEDURE read ( VARIABLE frontelement : ORDER;
                 front : INDEX;
                 entries : INTEGER;
                 empty : BOOLEAN;
                 qqueue : QUEUE);
```

d. Erläutern Sie, was unter dem FIFO-Prinzip zu verstehen ist, nennen Sie eine Alternative zum FIFO-Prinzip und beschreiben Sie kurz die Funktionsweise. (4P)

e. Welche Traversierungsform ist für die Auflösung eines binären (Such-)Baumes geeignet und was ist dabei zu beachten? (4P)

Aufgabe 3 (Programmieren in C) 38P

In der gesamten Aufgabe 3 wird von Ihnen erwartet, dass Sie die im Lehrbrief dargestellte Programmiersprache C ausnahmslos verwenden. Für die Implementierung mit dieser Programmiersprache stehen Ihnen damit die spezifischen Sprachkonzepte von C zur Verfügung.

a. Ein Unternehmen für Pumpentechnik zeichnet im Rahmen des Qualitätsmanagements Daten einiger neuer Prototypen auf. Für eine Präsentation bei einem Kunden wird ein schnell auszuführendes C-Programm benötigt, welches das arithmetische Mittel von reellen Messwerten bestimmt. Für eine Folge von reellen (Mess-)Werten $x_1, x_2, \ldots, x_n$ mit $x_n \in \mathbb{R}, n \geq 1$, ist das arithmetische Mittel $\overline{x}$ gemäß (10P)

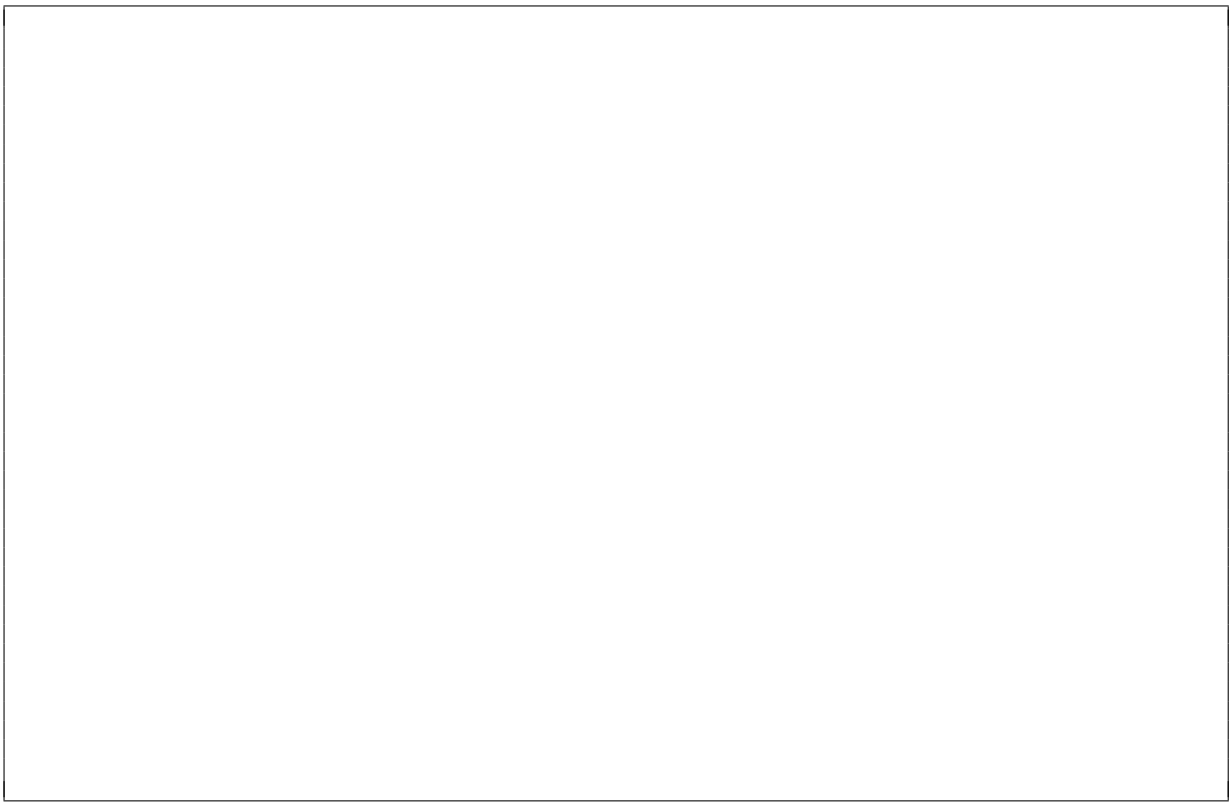
$$\overline{x}_n = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$$

zu berechnen. Gegeben sei ein Vektor mit fünf spezifischen Messpunkten, dessen arithmetisches Mittel bestimmt werden soll. Im ersten Schritt erwartet Ihr Arbeitgeber, dass Sie eine iterative Funktion zur Berechnung des arithmetischen Mittels entwickeln, die den Prototypen `ermittle_am` verwenden soll. Anschließend sollen Sie aus Effizienzgründen eine rekursive Funktion zur Bestimmung des arithmetischen Mittels entwerfen, die folgenden Prototypen `ermittle_am_rek` aufweisen soll. Es existieren für beide Funktionen bislang nur die Funktionsköpfe, die Ihnen als Entwickler nun als Grundlage dienen sollen. Bitte ergänzen Sie den fehlenden Funktionsrumpf für beide Funktionen an den dafür vorgesehenen Stellen. Nutzen Sie bei der Formulierung der rekursiven Berechnung folgende Beziehung:

$$\overline{x}_n = \frac{1}{n} \cdot (\overline{x}_{n-1} \cdot (n-1) + x_n)$$

```c
#include <stdio.h>

double ermittle_am(double *x, int pos){ ..
} // Zeiger auf 1. Vektorelement (Index 0) & Index des letzten
   Vektorelements
double ermittle_am_rek(double *x, int pos){..
}

void main(void)
{
 int length = 5;
 double x[length-1] = {12.3, 3.5, 15.0, 7.9, 8.2};
 double result1, result2;

 result1 = ermittle_am(werte, laenge-1);
 result2 = ermittle_am_rek(werte, laenge-1);
}
```

```
double ermittle_am(double *x, int pos){
```



```
}
double ermittle_am_rek(double *x, int pos){
```

```
}
```

b. Das Unternehmen Schleierwerke produziert verschiedene Müslisorten. Pro Monat belaufen sich die Fixkosten auf 600 GE (Geldeinheiten). Bei der Produktion von Müsli ab 2 ME (Mengeneinheiten, was zwei $500g$ Packungen entspricht) entstehen monatliche Gesamtkosten von 1392 GE, bei 4 ME sind es bereits 2016 GE und bei 6 ME betragen die monatlichen Gesamtkosten 2544 GE. Ein Mitarbeiter aus der Organisationseinheit Controlling hat bereits darauf basierend eine Kostenfunktion entworfen: (10P)

$$K_f(x) = \frac{3}{2} \cdot x^3 - 30 \cdot x^2 + 450 \cdot x + 600$$

Sie als C-Entwickler/in sollen nun ein Programm entwickeln, welches die Wertetabelle der Kostenfunktion ausgibt. Dabei sind folgende Anforderungen an das Programm zu berücksichtigen: (1) Die Obergrenze `obergrenze` (10.0) und Untergrenze `untergrenze` (0.0) der Wertetabelle (x-Werte) sollen statisch sein; (2) der Nutzer soll eine Argumentanzahl (Anzahl der x-Werte) eingeben können, auf der basierend dann die x-Werte bzw. die Ausgabewerte (Funktionswerte) berechnet werden (Hinweis: `delta`: $delta = (obergrenze - untergrenze)/anz$); (3) die Ausgabe der Wertetabelle soll durch die Realisierung einer Funktion vollzogen werden, sodass der folgende gegebene Prototyp beim Aufruf im Hauptprogramm eine *formatierte* Wertetabelle ausgibt:

```
void wertetabelle(char *fbez, int anz, float (*funk_zgr) (float))
```

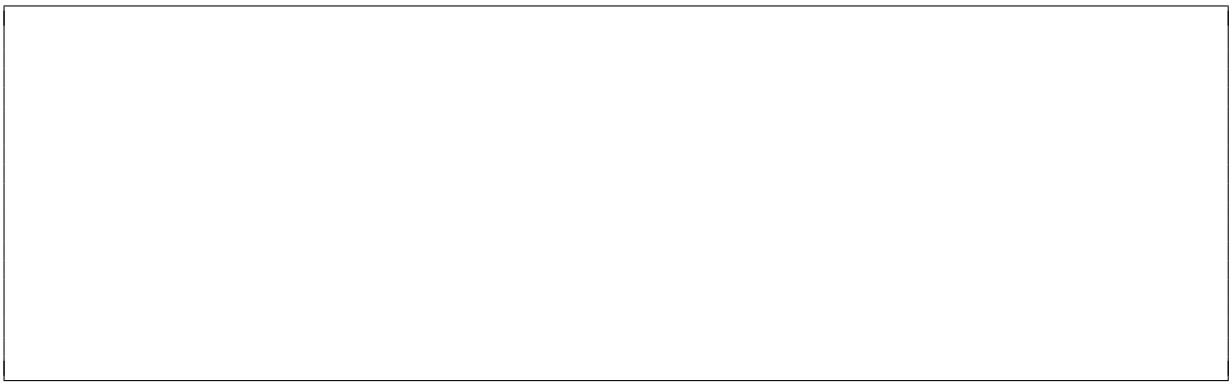
Tragen Sie an den entsprechenden Stellen den fehlenden Programmquellcode ein. Denken Sie auch an die vom Nutzer einzugebende Argumentanzahl und die dazugehörige Abfrage. Dafür dient Ihnen folgender Programmquellcode als Ausgangspunkt:

```
#include <stdio.h>

float Kf(float x); //Kostenfunktion
void wertetabelle(char *fbez, int anz, float (*funk_zgr) (float));

void main(void){
```

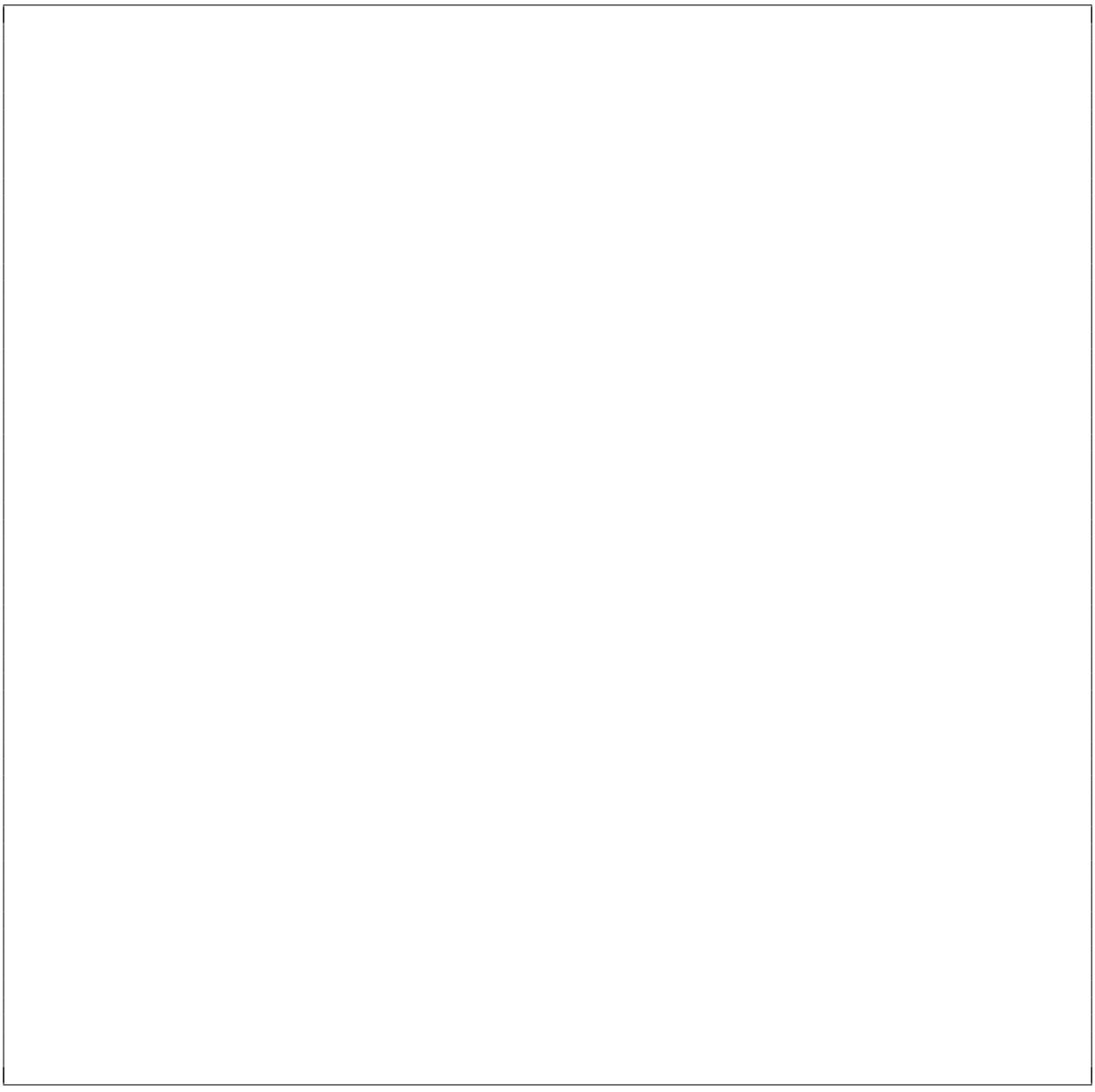
```

```

```
}
```

```
float Kf(float x){
```



```
}
void wertetabelle(char *fbez, int anz, float (*funk_zgr) (float)){
```



```
}
```

c. Im Folgenden finden Sie ein Fragment zu logischen Operatoren. Kreuzen Sie deutlich an, ob bei einer Auswertung 0 (FALSE) oder 1 (TRUE) resultiert. (6P)

```
int a = 0, b = 1, c = 5, e = -1, res;
float d = 2.3;
```

	0 (FALSE)	1 (TRUE)
`res = a || (b && c);`		
`res = (!c || b) && (b && c);`		
`res = (a && b) && (c && b);`		
`res = (!e && b) || (e && !d);`		
`res = !b && (a<d);`		
`res = ((e<0 && d<e) && (c<5)) || (e && a<b);`		

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

```
int summand=10, summe=0;
 while (summand>0){
  printf("\nZwischenstand: %d", summe);
  summe += summand;
  summand -= 2;
 }
```

e. »Der Einsatz verketteter Datenstrukturen sollte statischen Datenstrukturen stets vorgezogen werden.« Diskutieren Sie diese Aussage unter verschiedenen Gesichtspunkten. (7P)