BABEŞ-BOLYAI-UNIVERSITÄT CLUJ-NAPOCA
FAKULTÄT FÜR MATHEMATIK UND INFORMATIK

Schriftliche Prüfung für den Bachelor-Abschluss, 3. September 2024
Informatik in deutscher Sprache

VARIANTE 2

ANMERKUNG

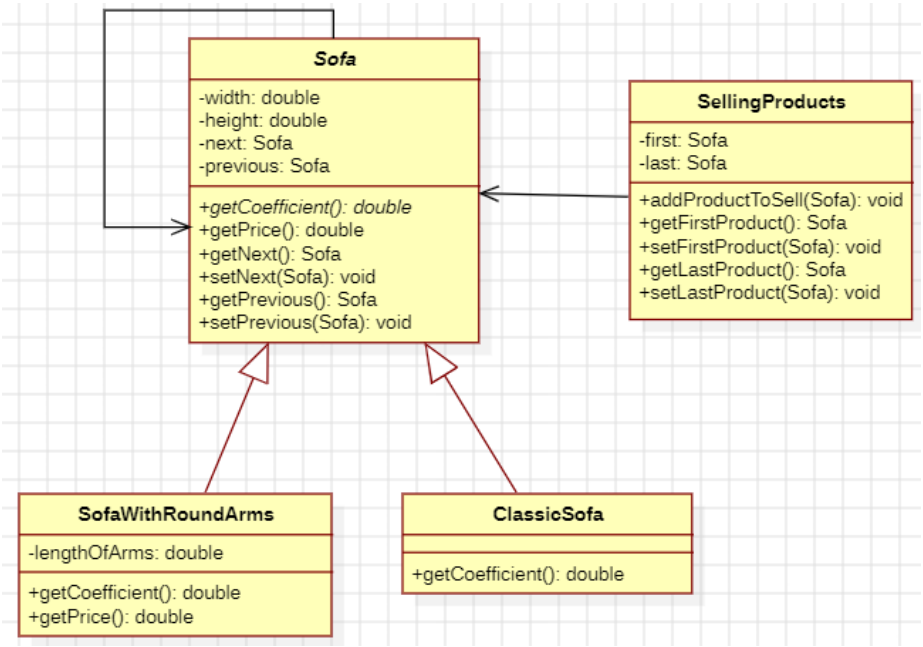
- Alle Aufgaben sind Pflichtaufgaben. Für alle Aufgaben sind Lösungen mit vollständigen Lösungen erforderlich.
- Die Mindestnote zum Bestehen ist 5,00.
- Die tatsächliche Arbeitszeit beträgt 3 Stunden.

THEMA Algorithmen und Programmierung

- **Geben Sie die verwendete Programmiersprache an.**
- **Das Fehlen eines angemessenen Programmierstils (aussagekräftige Variablennamen, Einrückung des Codes, Kommentare, falls erforderlich, Lesbarkeit des Codes) führt zu einem Verlust von 10 % der Fachnote.**
- **Fügen Sie keine anderen als die in der Anweisung genannten Attribute und Methoden hinzu, mit Ausnahme von Konstruktoren und Destruktoren, falls vorhanden. Ändern Sie nicht die Sichtbarkeit der in der Anweisung angegebenen Attribute.**
- **Es werden keine sortierten Container und vordefinierten Sortier- und Suchoperationen verwendet.**

*Für **Datentypen** können Sie vorhandene Bibliotheken verwenden* (*C++, Java, C#*).

a) Das folgende UML-Diagramm enthält die Klassen **Sofa**, **SofaWithRoundArms**, **ClassicSofa**, **SellingProducts** (speichert die Sofas zum Verkauf). Die Konstruktoren sind auf dem Diagramm nicht angegeben.



- Die Attribute *width* und *height* der Klasse **Sofa** (Breite und Höhe) und das Attribut *lengthOfArms* der Klasse **SofaWithRoundArms** (Länge der Armlehnen gemessen in Metern) müssen streng positive Werte sein. Die Konstrukteure müssen die Beschränkungen durchsetzen.
- Die abstrakte Klasse **Sofa** verfügt über eine abstrakte Methode *getCoefficient*(), die einen Koeffizienten zurückgibt, der zur Berechnung des Preises eines Sofas verwendet wird.

- Der Koeffizient für die Berechnung des Preises für ein **ClassicSofa-Objekt** ist 1,5 und für ein **SofaWithRoundArms-Objekt** ist der Koeffizient 2.
- Der Preis eines Sofas errechnet sich aus dem Koeffizienten multipliziert mit der Summe aus *Breite* und *Höhe*. Bei **SofaWithRoundArms** wird der Preis für die Armlehnen zu diesem Preis addiert. Ein Meter Armlehnen kostet 10 Lei.
- Die Methode **addProductToSell(Sofa)** der Klasse **SellingProducts** fügt das angegebene Objekt als Parameter zu der bereits vorhandenen Sofasequenz hinzu**.** Das Hinzufügen erfolgt **so,** dass die Anzeige der Sofasequenz in absteigender Reihenfolge des Preises in linearer Zeit erfolgen kann.

Schreiben Sie ein Programm in einer der Programmiersprachen C++, Java oder C# mit den folgenden Anforderungen:

a1) Deklarieren Sie alle Klassen, Attribute und Methoden gemäß dem obigen Diagramm.

Implementieren Sie nur die folgenden Methoden:

a2) Die Methode **getPrice()** der Klasse **SofaWithRoundArms.**
a3) Die Konstruktoren der Klassen **Sofa, SofaWithRoundArms** und **SellingProducts**.
a4) Die Methode **addProductToSell(Sofa)** der Klasse **SellingProducts**.
a5) Die Methode **getCoefficient()** der Klasse **ClassicSofa** und die Methode **setPrevious(Sofa)** der Klasse **Sofa**.

b) Definieren Sie eine Funktion, die als Parameter ein Objekt *s* vom Typ **SellingProducts** erhält und in linearer Zeit die Preise der Sofas in ***s*** ***in*** aufsteigender Reihenfolge ihrer Preise ausgibt. Implementierungen, die nicht in die angegebene Komplexitätsklasse fallen, werden teilweise bewertet.

c) Definieren Sie eine Funktion, die als Parameter ein Objekt *s* vom Typ **SellingProducts** und zwei reelle Werte *startPrice* und *endPrice* annimmt und aus *s* die Sofas entfernt, deren Preis im Bereich [*startPrice*, *endPrice*] liegt. Wir verwenden eine Implementierung mit der Zeitkomplexität $O(n)$ (n ist die Anzahl der Sofas in der mit dem **SellingProducts-Objekt** verbundenen Liste). Implementierungen, die nicht in die angegebene Komplexitätsklasse fallen, werden teilweise gewertet.

d) Konstruieren Sie ein Objekt vom Typ **SellingProducts**, in das Sie 4 Sofas einfügen (wählen Sie Werte für nicht spezifizierte Eigenschaften)**:** zwei vom Typ **ClassicSofa** und zwei vom Typ **SofaWithRoundArms** mit 1,2 bzw. 0,9 laufenden Metern Armlänge**.** Rufen Sie die Funktion in b) und dann die Funktion in c) auf (wählen Sie Werte für *startPrice* und *endPrice*).

e) Geben Sie die Zeitkomplexität der Methode **addProductToSell(Sofa)** der Klasse **SellingProducts** für den *günstigen*, den *durchschnittlichen* und den *ungünstigen* Fall an.

SUBJECT Datenbanken

Problem 1 (4 Punkte)

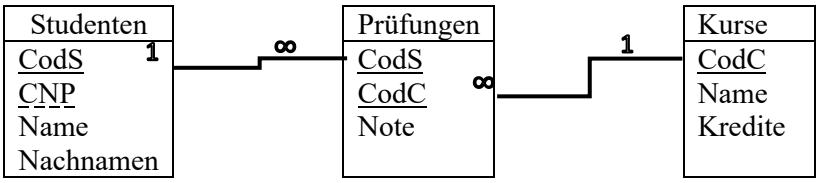
Die Organisatoren von Wettbewerben zur Muschelernte speichern die folgenden Informationen in einer relationalen Datenbank:

- Ein Teilnehmer hat einen Code, einen Namen, eine E-Mail-Adresse und ein Herkunftsland.
- Ein Strand hat einen Code, einen Namen, eine Beschreibung und eine Örtlichkeit.
- Ein Wettbewerb hat einen Code, einen Namen, ein Datum, an dem er stattfinden soll, eine Startzeit, eine Endzeit und einen zugehörigen Strand. An einem Strand kann mehr als ein Wettbewerb stattfinden.
- Ein Muscheltyp hat einen Code, einen Namen und eine Beschreibung.
- Bei jedem Wettbewerb sammeln die Teilnehmer verschiedene Arten von Muscheln am Strand, an dem der Wettbewerb stattfindet. In der Datenbank wird nur die Gesamtzahl der Muscheln gespeichert, die jeder Teilnehmer in jedem Wettbewerb für jede Muschelart gesammelt hat (z. B. Teilnehmer *p* sammelte: 10 Muscheln der Art *t1* und 15 Muscheln der Art *t2* im Wettbewerb *c1*; 50 Muscheln der Art *t1*, 15 Muscheln der Art *t2* und 20 Muscheln der Art *t3* im Wettbewerb *c2* usw.).

Erstellen Sie ein relationales BCNF-Schema für die Datenbank, wobei Sie Primärschlüssel, Kandidatenschlüssel und Fremdschlüssel sorgfältig hervorheben. Erstellen Sie das Schema auf eine der im folgenden Beispiel gezeigten Arten:

* Beispiel für die Tabellen Schüler, Kurse und Prüfungen:

V1. Tabelle mit Tabellen, Primärschlüssel unterstrichen mit durchgezogener Linie, Kandidatenschlüssel unterstrichen mit gestrichelter Linie, direkte Verknüpfungen zwischen externen Schlüsseln und entsprechenden Primär-/Kandidatenschlüsseln (z. B. Verknüpfung zwischen Spalte CodS in Prüfungen und Spalte CodS in Studenten).



V2.

Studenten [CodS, CNP, Name, Nachname]
Kurse [CodeC, Name, Credits]
Prüfungen [CodS, CodC, Note].
Primärschlüssel sind durch eine durchgezogene Linie und Kandidatenschlüssel durch eine gestrichelte Linie unterstrichen. {CodS} ist der Fremdschlüssel in Prüfungen und verweist auf {CodS} in Studenten. {CodC} ist der Fremdschlüssel in Prüfungen und verweist auf {CodC} in Kurse.

Problem 2 (5 Punkte)

Betrachten Sie die folgenden Beziehungen aus einer Datenbank über einen privaten Kindergarten:
Gruppen [GruppenCode, GroupenName, RaumNummer]
Eltern [ElternCode, ElternName, Adresse, Telefonnummer]
Kinder[KindCode, KindName, *ElternCode,* Geburtsdatum, *GrupenCode*]
Zahlungen[ZahlungCode, *KindCode*, ZahlungDatum, Betrag]

Primärschlüssel sind unterstrichen. Externe Schlüssel sind kursiv geschrieben und haben denselben Namen wie die Spalten, auf die sie sich beziehen.

a. Schreiben Sie eine SQL-Abfrage, die für jede Gruppe, für die mindestens eine Gebühr gezahlt wurde, folgendes zurückgibt: Code, Gruppenname und Gesamtbetrag der für die Kinder in der Gruppe gezahlten Gebühren.

b. Nennen Sie die folgenden Beispiele für die Beziehungen Gruppen, Eltern und Kinder:

Gruppen:

GruppenCode	GruppenName	RaumNummer
1	Marienkäfer	1
2	Eichhörnchen	5
3	Bienen	8

Eltern:

EternCode	ElternName	Adresse	Telefonummer
1	P1	A1	1111111111
2	P2	A2	2222222222
3	P3	A3	3333333333

Kinder:

Kind Code	KindName	ElternCode	Geburtsdatum	GruppenCode
1	C1	1	2018.02.12	1
2	C2	2	2018.06.25	1
3	C3	2	2018.06.25	1
4	C4	3	2018.06.30	1
5	C5	1	2020.04.15	2
6	C6	1	2022.06.05	3

b1. Geben Sie das Ergebnis der Auswertung der untenstehenden Abfrage für die angegebenen Instanzen an. Geben Sie ausschließlich die Werte des/der Tupel(s) und die Spaltennamen im Ergebnis an, ohne alle Schritte der Abfrageauswertung zu nennen.

```
SELECT p.ElternName, p.Adresse
VON Eltern p
    INNER JOIN Kinder c ON p.ElternCode = c. ElternCode
WHERE c.GruppenCode = (SELECT g.GruppenCode
                       FROM Gruppen g
                       WHERE g.GruppenName = 'Marienkäfer')
INTERSECT
SELECT p.ElternName, p.Adresse
VON Eltern p
    INNER JOIN Kinder c ON p.ElternCode = c.ElternCode
GROUP BY p. ElternCode, p.ElternName, p.Adresse
HAVING COUNT(DISTINCT c.GruppenCode) = 1
```

b2. Erkläre, ob die folgenden funktionalen Abhängigkeiten durch die Daten in der Instanz Kinder erfüllt werden oder nicht:

- {KindCode} → {ElternCode}
- {ElternCode} → {GruppenCode}.

THEMA Betriebssysteme

Aufgabe 1 (5 Punkte). Beantworten Sie die folgenden Fragen zur Ausführung des Programms `./a.out`, das aus dem untenstehenden Quellcode kompiliert wurde, unter der Annahme, dass alle notwendigen Includes vorhanden sind, der Systemaufruf `fork` und der Systemaufruf `execlp` erfolgreich ausgeführt werden und die FIFOs `fa` und `fb` zuvor angelegt wurden. Der `bash`-Befehl mit der Option `-c` führt den Befehl aus der Zeichenkette aus, die als Wert für die Option `-c` angegeben wurde.

```
 1  int main(int argc, char** argv) {
 2    int fa, fb, k; char s[32];
 3
 4    if(fork() == 0) {
 5      execlp("bash","bash","-c","sort <fa >>fb",NULL);
 6      exit(0);
 7    }
 8    if(fork() == 0) {
 9      fa = open("fa", O_WRONLY);
10      for(k=1; k<argc; k++) {
11        write(fa, argv[k], strlen(argv[k]));
12        write(fa, "\n", 1);
13      }
14      close(fa);
15      exit(0);
16    }
17    if(fork() == 0) {
18      fb = open("fb", O_RDONLY);
19      while((k = read(fb, s, 32)) > 0) {
20        write(1, s, k);
21      }
22      close(fb);
23      exit(0);
24    }
25    wait(NULL); wait(NULL); wait(NULL);
26    return 0;
27  }
```

a) Was wird in der folgenden Ausführung angezeigt? Begründen Sie Ihre Antwort.

```
./a.out a d b c
```

b) Wie wird die Ausführung beeinflusst, wenn der IF-Block in den Zeilen 8-16 zwischen die Zeilen 3 und 4 verschoben wird? Bitte begründen Sie Ihre Antwort.

c) Was wird die folgende Ausführung anzeigen, wenn in Zeile 20 die Zahl `1` in `2` geändert wird? Begründen Sie Ihre Antwort.

```
./a.out y z x > /dev/null
```

d) Wie wird die Ausführung beeinflusst, wenn Zeile 14 kommentiert wird? Bitte begründen Sie Ihre Antwort.

e) Wie wird die Ausführung beeinflusst, wenn die Zeilen 4, 6 und 7 kommentiert werden? Bitte begründen Sie Ihre Antwort.

Aufgabe 2 (4 Punkte). Beantworten Sie die folgenden Fragen zur Ausführung des UNIX-Shell-Skripts `./a.sh`.

```
 1  #!/bin/bash
 2
 3  RE='^(.*)([+-]) (-?[0-9]+) (-?[0-9]+)(.*)$'
 4  L="$1"
 5  while true; do
 6    if echo $L | grep -q -v -E "$RE"; then
 7      exit 1
 8    fi
 9
10    A=`echo $L | sed -E "s/$RE/\1/"`
11    B=`echo $L | sed -E "s/$RE/\3 \2 \4/"`
12    C=`echo $L | sed -E "s/$RE/\5/"`
13    L="$A `expr $B` $C"
14
15    echo "A='$A'  B='$B'  C='$C'"
16
17    if echo $L | grep -E -q "^ *-?[0-9]+ *$"; then
18      break
19    fi
20  done
21  echo $L
```

a) Was wird die folgende Ausführung zeigen?

```
./a.sh "+ 1 2"
```

b) Was werden die Werte der Variablen A, B und C bei jeder Iteration des `while`-Zyklus sein, und welcher Wert wird bei der folgenden Ausführung angezeigt?

```
./a.sh "+ + 1 2 3"
```

c) Was werden die Werte der Variablen A, B und C bei jeder Iteration des `while`-Zyklus sein, und welcher Wert wird bei der folgenden Ausführung angezeigt? Begründen Sie Ihre Antwort.

```
./a.sh "+ + -1 2 -3 4"; echo $?
```

d) Erklären Sie im Detail, wie die Zeilen 17-19 funktionieren.

BAREM INFORMATICĂ
VARIANTA 2

Subiect Algoritmică şi Programare

Oficiu – 1p
Cerința **a) – 4.6p**
- Definirea clasei Sofa – **0.55p** din care
  - clasă abstractă – 0.05
  - atribute - 0.1
  - **constructor (a3)** - 0.1
  - metoda **setPrevious (a5)** – 0.1
  - metode **getCoefficient**, **getPrice**, **getNext**, **setNext**, **getPrevious** - 0.2
- Definirea clasei SofaWithRoudArms– **0.85p** din care
  - relația de moștenire – 0.2
  - atribut – 0.1
  - **constructor (a3)** – 0.2
  - metoda **getPrice (a2)** – 0.25
  - metoda **getCoefficient** - 0.1
- Definirea clasei ClassicSofa – **0.4p** din care
  - relația de moștenire – 0.25
  - metoda **getCoefficient (a5)** - 0.15
- Definirea clasei SellingProducts– **2.8p** din care
  - atribute– 0.2
  - **constructor (a3)** - 0.1
  - metoda **addProductToSell (a4)** – 2.5
    - adăugare listă vidă – 0.3
    - adăugare început – 0.3
    - adăugare sfârșit – 0.3
    - inserare în listă – 1.6

Funcția **b) - 1.1p**
- signatura - 0.1
- afișare în $\theta(n)$ - 1p
  * afișare în complexitate timp mai mare decât cea cerută– 0.25

Funcția **c) - 1.9p**
- signatura - 0.1
- ștergere în $O(n)$ – 1.8
  - ștergere început – 0.3
  - ștergere sfârșit – 0.3
  - ștergere în listă – 1.2

  * ștergere în complexitate timp mai mare decât cea cerută– 0.5

Funcția principală **d)** – **0.4p**
- construire obiecte – 0.2p
- apel funcție b) – 0.1p
- apel funcție c) - 0.1 p

Cerința **e)** – **1p**
- favorabil (0.25p)
- mediu (0.5p)
- defavorabil (0.25p)

BAREM INFORMATICĂ
VARIANTA 2

Subiect Baze de date

Oficiu – **1p**

Problema 1. Punctaj - 4p

- relații cu atribute corecte, chei primare, chei candidat: **3p**
- legături modelate corect (chei externe): **1p**

Problema 2. Punctaj - 5p

- **a** - rezolvarea completă a interogării: **2.5p**

- **b1** - rezultat evaluare interogare:

NumeParinte	Adresa
P2	A2
P3	A3

  - coloane – **0.5p**
  - valori tuplu – **1p**

- **b2** - {CodCopil}→{CodParinte} este satisfăcută – **0.25p**; **0.25p** explicație
  - {CodParinte}→{CodGrupa} nu este satisfăcută – **0.25p**; **0.25p** explicație

Notă: La specializările Informatică engleză și Informatică maghiară se iau în considerare versiunile traduse în limbile corespunzătoare.

BAREM INFORMATICĂ
VARIANTA 2

Subiect: Sisteme de Operare

1p – oficiu

Problema 1 (5p)

1p – a) a b c d, pentru că sort citeşte din fa ce scrie programul şi apoi programul citeşte din fb ce afişează sort

1p – b) nicicum, cele trei procese fiu sunt concurente şi ordinea creării lor nu are nicciun efect asupra execuţiei

1p – c) x y z, pentru ca deşi redirecţionăm ieşirea standard în /dev/null, programul scrie în ieşirea de eroare

1p – d) nicicum, procesul fiu se încheie şi FIFO-ul e închis automat, astfel sort nu ramâne blocat la citire

1p – e) se va bloca fără a afişa nimic, pentru că nu poate deschide FIFO-urile, procesele fiu nemaifiind create din cauză că execlp suprascrie codul procesului părinte

Problema 2 (4p)

1p – a) 3 – scriptul calculează suma celor două numere

1p – b) A='+ ' B='1 + 2' C=' 3'
A=" B='3 + 3' C="
Se afișează 6

1p – c) A='+ ' B='-1 + 2' C=' -3 4'
A=" B='1 + -3' C=' 4'
Se va afişa 1 pentru că $L nu se va potrivi cu expresia regulară de la linia 6 şi se va face exit 1, iar $? va conţine această valoare

1p – d) Dacă variabila L conţine rezultatul final, adică o valoare întreagă cu eventuale spaţii înainte şi după, valoarea de adevăr a comenzii grep va fi TRUE şi ca urmare se intră in IF şi se va executa comanda break, astfel încheindu-se cu succes execuţia scriptului