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

Schriftliche Abschlussprüfung
Fachgebiet Informatik in deutscher Sprache

VARIANTE 2

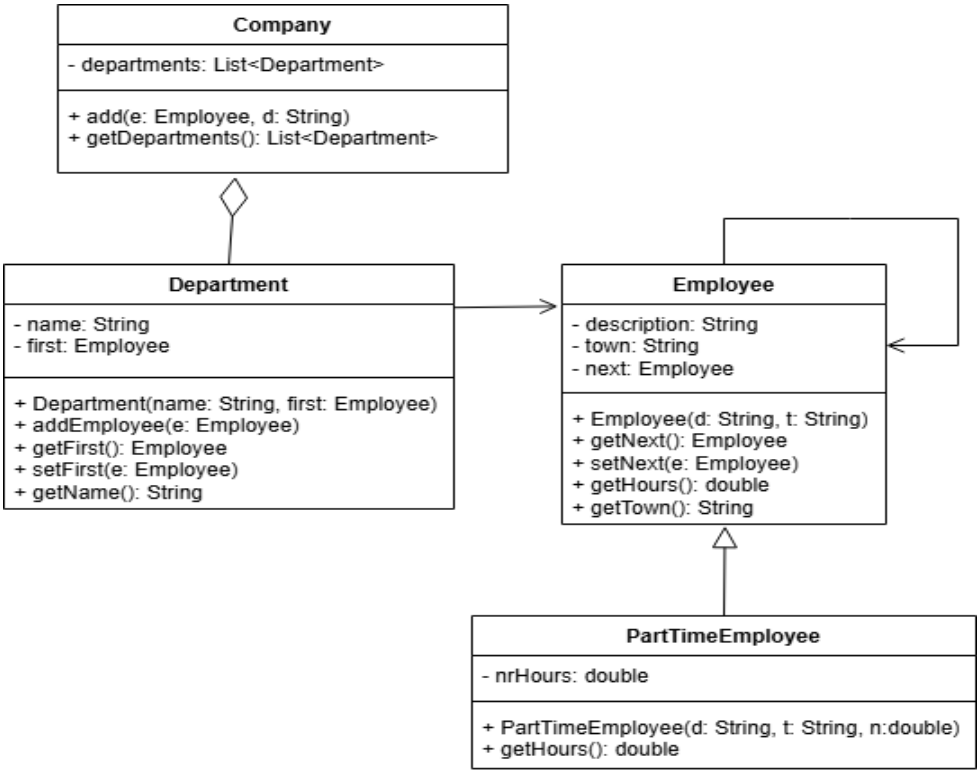
BEMERKUNG.

- Alle Prüfungsthemen sind verpflichtend. Bei allen Prüfungsthemen müssen vollständige Lösungen angegeben werden.
- Die Mindestnote für das Bestehen der Abschlussprüfung ist 5.00.
- Die Arbeitszeit beträgt 3 Stunden.

THEMA Algorithmen und Programmierung

- **Geben Sie die verwendete Programmiersprache an.**
- **Die erforderlichen Implementierungen sind nicht verpflichtet, dynamisch zugewiesene Speicherbereiche freizugeben.**
- **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- oder Suchoperationen verwendet.**
- **Für Datentypen können Sie vorhandene Bibliotheken verwenden (Python, C++, Java, C#).**

1. **(2.5 Punkte)** Definieren Sie eine Funktion in einer der Programmiersprachen **Python, C++, Java** oder **C#**, die als Parameter erhält, eine alphabetisch sortierte Liste *lst* von Zeichenketten (die die Namen von Abteilungen innerhalb eines Unternehmens darstellen) und eine Zeichenkette *name*, die den Namen einer Abteilung darstellt. Unter der Annahme (als Vorbedingung der Funktion), dass die Zeichenkette *name* nicht in der Liste *lst* enthalten ist, fügt die Funktion die Zeichenkette *name* zur Liste *lst* hinzu. Das Hinzufügen erfolgt so, dass die Liste *lst* alphabetisch nach Abteilungsnamen sortiert bleibt. Es wird eine rekursive Implementierung verwendet. Es können Hilfsfunktionen implementiert werden. Iterative Lösungen sind teilweise gepunktet.
2. **(1 Punkt)** Geben Sie die Zeitkomplexität der Funktion aus **Aufgabe 1.** für den *besten*, *mittleren* und *schlechtesten* Fall an. Begründen Sie Ihre Antwort.
3. **(5.5 Punkte)** Gegeben sei das folgende UML-Diagramm, das die Klassen **Company** (Firma), **Department** (Abteilung, die eine Sammlung von Mitarbeitern darstellt), **Employee** (Mitarbeiter), **PartTimeEmployee** (Teilzeitmitarbeiter) enthält.



- Die Attribute *description* und *town* in der Klasse **Employee** (Stellenbeschreibung und Heimatort) und *name* in der Klasse **Department** (Abteilungsname) dürfen nicht leer sein, und das Attribut *nrHours* in der Klasse **PartTimeEmployee** (monatliche Stundenzahl des Mitarbeiters) muss streng positiv und kleiner als 160 sein. Konstruktoren müssen die Bedingungen durchsetzen.
- Die Klasse **Employee** stellt eine **getHours()-**Methode bereit, die 160 zurückgibt, und eine **getTown()-**Methode, die den Heimatort des Mitarbeiters zurückgibt. Die Methoden **getNext()** und **setNext(e: Employee)** geben zurück bzw. setzen das Attribut *next*.
- Die Methode **getHours()** der Klasse **PartTimeEmployee** gibt die monatliche Stundenzahl des Teilzeitmitarbeiters zurück (Attribut *nrHours* der Klasse).
- Die Klasse **Department** speichert eine Sammlung von Mitarbeitern (Voll- oder Teilzeit) in der Abteilung. Die Methode **addEmployee(e: Employee)** fügt der Abteilung einen Mitarbeiter hinzu, und die Methode **getName()** gibt den Namen der Abteilung zurück. Die Methoden **getFirst()** und **setFirst(e: Employee)** geben zurück bzw. setzen das Attribut *first*.
- Die Klasse **Company** speichert eine Liste von Abteilungen. Die Methode **add(e: Employee, d: String)** fügt den Mitarbeiter *e* der Abteilung mit dem Namen *d* hinzu (wenn es keine Abteilung mit dem Namen *d* gibt, wird sie angelegt). Die Methode **getDepartments()** gibt die Liste der Abteilungen der Firma zurück.

Schreiben Sie ein Programm in einer der Programmiersprachen **C++, Java** oder **C#**, das die folgenden Anforderungen erfüllt:

a) Deklarieren Sie alle Klassen, Attribute und Methoden gemäß dem obigen Diagramm. Implementieren Sie nur die folgenden Methoden:

a1) Die Methode **setNext(e: Employee)** in der Klasse **Employee**.

a2) Die Konstruktoren der Klassen **Department** und **PartTimeEmployee**.

a3) Die Methoden **addEmployee(e: Employee)** und **setFirst (e: Employee)** in der Klasse **Department**. Für die Methode **addEmployee(e: Employee)** wird eine Implementierung mit der Zeitkomplexität im schlechtesten Fall $\theta(1)$ benötigt. Lösungen, die die geforderte Komplexität nicht erfüllen, werden teilweise bewertet.

a4) Die Methode **add(e: Employee, d: String)** der Klasse **Company**.

b) Implementieren Sie eine Funktion, die als Parameter ein Objekt *c* vom Typ **Company** annimmt und die maximale Anzahl der Mitarbeiter (*m*) berechnet, die aus derselben Stadt kommen, und dann alle Städte ausgibt, aus denen *m* Mitarbeiter kommen (eine Stadt wird nur einmal angezeigt). Unter der Annahme, dass die Städtenamen aus höchstens 20 Zeichen bestehen, wird eine Implementierung mit einer Zeitkomplexität von im mittleren Fall $\theta(n)$ benötigt (*n* ist die Anzahl der Mitarbeiter des Unternehmens). Lösungen, die die geforderte Komplexität nicht erfüllen, werden teilweise bewertet. **Es ist nicht erlaubt, vordefinierte Bibliotheksfunktionen zu verwenden, um eindeutige Elemente in einer Sequenz zu bestimmen.**

c) Erzeugen Sie ein Objekt vom Typ **Company**, in das Sie 4 Mitarbeiter einfügen, von denen 2 teilzeitbeschäftigt sind (nach Ihrer Wahl). Rufen Sie dann die Funktion aus **b)** auf.

THEMA Datenbanken **VARIANTE 2**

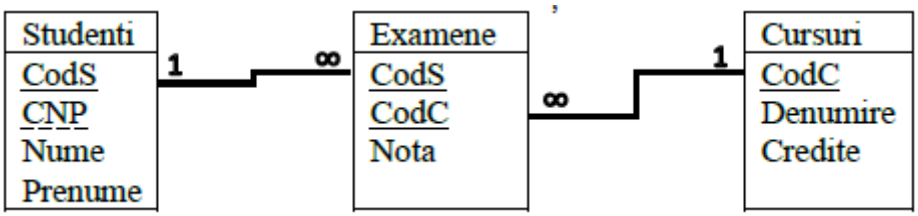
Aufgabe 1 (4 Punkte)

Ein Softwareunternehmen stellt Anwendungen für verschiedene Branchen her und verwendet dabei Technologien, die auf jede dieser Branchen zugeschnitten sind. Das Unternehmen verwendet eine relationale Datenbank, um die Zuweisung der Mitarbeiter der Abteilungen zu den in der Firma produzierten Projekten zu verwalten:

- Ein Mitarbeiter hat einen Code, ein Einstellungsdatum, einen Namen (kann ein einfaches Attribut, eine Zeichenkette sein), eine E-Mail-Adresse (eindeutig für jeden Mitarbeiter) und einen Wert für sein Monatsgehalt; außerdem wird ein Mitarbeiter einer einzigen Abteilung und einem einzigen Projekt zugeordnet.
- Jede Abteilung hat einen Code und einen Namen; eine Abteilung kann mehr als ein Projekt zur gleichen Zeit bearbeiten.
- Ein Projekt hat einen Code, einen eindeutigen internen Projektnamen, ein Budget und eine dominante Technologie: Das Projekt mit dem internen Projektnamen P1T1 hat beispielsweise ".NET" als dominante Technologie. Es wird in einer Abteilung realisiert.
- Das Unternehmen hat mehrere Standorte, an denen die Mitarbeiter arbeiten können: Ein Standort hat einen Code, eine Adresse (kann ein einfaches Attribut, eine Zeichenfolge sein) und eine Telefonnummer (für jeden Standort eindeutig); ein Standort hat einen Parkplatz oder nicht, hat eine Cafeteria oder nicht.
- An einem Standort können Mitarbeiter aus mehreren Abteilungen arbeiten, aber für einen Standort ist von Anfang an festgelegt, welche Abteilungen an diesem Standort arbeiten. Einem Standort können mehrere Abteilungen zugeordnet sein, und eine Abteilung kann mehreren Standorten zugeordnet sein.

Realisieren Sie ein relationales BCNF-Schema für die Datenbank, wobei Sie Primärschlüssel, Kandidatenschlüssel und Fremdschlüssel genau markieren. Realisieren Sie das Schema auf eine der im folgenden Beispiel gezeigten Arten:
* Beispiel für die Tabellen Studenten, Kurse und Prüfungen:

V1. Diagramm mit Tabellen, Primärschlüssel durch eine durchgezogene Linie unterstrichen, Kandidatenschlüssel durch eine gestrichelte Linie unterstrichen, Verknüpfungen direkt zwischen Fremdschlüsseln und den entsprechenden Primär-/Kandidatenschlüsseln gezeichnet (z. B. Verknüpfung zwischen der Spalte CodS in Prüfungen und der Spalte CodS in Studierende gezeichnet).



V2.
Studenten [CodS, CNP, Name, Nachname]
Lehrveranstaltungen [CodC, Name, Credits]
Prüfungen [CodS, CodC, Note]
Die Primärschlüssel sind mit einer durchgezogenen Linie unterstrichen, die Kandidatenschlüssel mit einer gestrichelten 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.

Aufgabe 2. (5 Punkte)

Die folgenden relationalen Schemata sind gegeben:

Authoren[AuthorCode, Name, Land, Geburtsjahr]

Bücher [CodeBuch, Titel, Bereich, Seitenanzahl, *AuthorCode*]

Publisher[PublisherCode, Name]

BücherPublishers *[CodeBuch, PublisherCode*, Preis, Erscheinungsjahr]

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

a. Schreiben Sie eine SQL-Abfrage, die den Code und den Namen jedes Autors zurückgibt, der mindestens 2 Bücher in einem Jahr geschrieben hat und mindestens ein Buch bei "DE2" veröffentlicht hat.

b. Geben Sie die folgenden Instanzen für die Relationen Books , Publishers und BücherPublishers an:

Bücher

BuchCode	Titel	Bereich	NrSeiten	AuthorCode
1	T1	D2	125	A1
2	T2	D1	175	A1
3	T3	D2	210	A3
4	T4	D3	98	A2
5	T5	D2	113	A4

Publisher

Publisher Code	Name
E1	DE1
E2	DE2
E3	DE3

BücherEditionen

BuchCode	PublisherCode	Preis	Erscheinungsjahr
1	E2	34	2021
2	E1	28	2022
3	E3	55	2021
4	E2	21	2024
5	E1	42	2023

b1. Geben Sie das Ergebnis der Auswertung der nachstehenden Abfrage auf den gegebenen Instanzen an. Geben Sie ausschließlich die Werte des/der Tupel(s) und die Spaltennamen im Ergebnis an, ohne alle Schritte der Abfrageauswertung darzustellen.

```
SELECT t1.AuthorCode, t1.Title, t1.BuchCode
FROM Bücher t1
WHERE
EXISTS
        (SELECT BuchCode FROM BücherPublishers t2
        INNER JOIN
        SELECT ce.PublisherCode FROM BücherPublisher ce
                GROUP BY ce.PublisherCode
                HAVING COUNT(*)>1 AND SUM(ce.Preis)<60
        UNION ALL
        SELECT e.PublisherCode FROM Publishers e WHERE e.Name='DE2'
        )t3 ON t3.PublisherCode=t2.PublisherCode
                WHERE t2.BuchCode=t1.BuchCode
        )
```

b2. Erläutern Sie, ob die folgenden funktionalen Abhängigkeiten durch die Daten in der Instanz Bücher erfüllt werden oder nicht:

- {NrSeiten} → {BuchCode}
- {AuthorCode} → {Bereich}

VARIANTE 2

THEMA Betriebssysteme

Aufgabe 1 (5 Punkte). Beantworten Sie die folgenden Fragen zur Ausführung des nachstehenden Programms `./a`, das anfangs ohne Argumente aufgerufen wird, unter der Annahme, dass alle erforderlichen Includes vorhanden sind und dass der Systemaufruf `fork` und mindestens einer der Systemaufrufe `exec` erfolgreich ausgeführt werden.

```
 1  int main(int argc, char** argv) {
 2      if(argc > 1) {
 3          printf("%s\n", argv[1]);
 4      }
 5      fork();
 6      execlp("a", "a", "y", NULL);
 7      execl("./a", "./a", "z", NULL);
 8      printf("%d\n", wait(NULL));
 9      return 0;
10  }
```

a) Erklären Sie im Detail, wie Zeile 6 funktioniert.

b) Geben Sie die Anzahl der Vorkommen des Wertes von `argv[1]` auf der Konsole an. Begründen Sie Ihre Antwort.

c) Was wird auf der Konsole angezeigt, wenn die Umgebungsvariable `PATH` das aktuelle Verzeichnis nicht enthält?

d) Wie viele Prozesse werden den Systemaufruf `wait` ausführen?

e) Was wird auf der Konsole angezeigt, wenn die Umgebungsvariable `PATH` das aktuelle Verzeichnis enthält und Zeile 5 entfernt wird?

Aufgabe 2 (4 Punkte). Beantworten Sie die folgenden Fragen zur Ausführung des nachstehenden UNIX-Shell-Skripts, falls die Datei `a.txt` existiert und in jeder Zeile Buchstabenfolgen enthält, die durch Leerzeichen getrennt sind.

```
 1  #!/bin/bash
 2
 3  while read L; do
 4    R=""
 5    for W in $L; do
 6      X=""
 7      C=`echo $W | sed -E "s/(.)/\1 /g"`
 8      for A in $C; do
 9        X="$A$X"
10      done
11      R="$X $R"
12    done
13    echo $R
14  done < a.txt
```

a) Welche Werte werden die Variablen L und W annehmen?

b) Erläutern Sie im Detail, wie Linie 7 funktioniert.

c) Was wird beim Ausführen des Skripts in der Konsole angezeigt, wenn die Datei `a.txt` den folgenden Inhalt hat?

```
abc def
pqr xyz
```

d) Was wird beim Ausführen des Skripts für die unter Punkt c) definierte Datei `a.txt` auf der Konsole angezeigt, wenn das Leerzeichen nach der Zahl `1` in Zeile 7 entfernt wird?

BAREM INFORMATICĂ
VARIANTA 2

Subiect Algoritmică și Programare

Oficiu – **1p**

Cerința **1.** **– 2.5p**

- signatura – **0.2p**
- implementare recursivă – **2.3p** din care
  - adăugare lista vidă – 0.5
  - adăugare la începutul/sfârșitul listei (pentru caz favorabil) – 0.5
  - apelul recursiv – 1.3

* implementare iterativă corectă – **0.7**p

Cerința **2.** **– 1p**

- caz favorabil **0.2p** din care
  - complexitate $\theta(1)$ – 0.1
  - justificare – 0.1
- caz mediu **0.4p** din care
  - complexitate $\theta(n)$ – 0.2 (n – dimensiunea listei *lst*)
  - justificare – 0.2
- caz defavorabil **0.4p** din care
  - complexitate $\theta(n)$ – 0.2 (n – dimensiunea listei *lst*)
  - justificare – 0.2

Cerința **3.a)** **– 2.6p**

Definirea clasei Employee – **0.6p** din care
- atribute – 0.1
- contructor – 0.1
- metoda **setNext (a1)** – 0.2
- metode **getNext**, **getHours**, **getTown** – 0.2

Definirea clasei PartTimeEmployee– **0.55p** din care
- relația de moștenire – 0.2
- atribut – 0.05
- **constructor (a2)** – 0.2
- metoda **getHours** – 0.1

Definirea clasei Department– **1p** din care
- atribute – 0.1
- **constructor (a2)**– 0.1
- metoda **addEmployee (a3)** – 0.5
  - * punctajul se acordă pentru implementare având complexitate $\theta(1)$
  - * pentru implementari având complexitate mai mare decat $\theta(1)$ se acorda 0.25p
- metoda **setFirst (a3) –** 0.2
- metode **getFirst**, **getName –** 0.1

Definirea clasei Company – **0.45p** din care
- atribut, metoda **getDepartments** – 0.1p
- metoda **add (a4)** – 0.35p

Funcția **3.b)** – **2.6p**

- signatura – **0.1p**
- implementare având complexitatea timp în caz mediu $\theta(n)$ – **2.5p**
  - * implementare având complexitatea timp în caz mediu $\theta(n\log_2 n)$ – **1.5p**
  - * implementare având complexitatea timp în caz mediu $\theta(n^2)$ – **0.75p**

Funcția principală **3.c)** – **0.3p**

- construire obiect de tip Company – 0.1p
- adăugare angajați - 0.1p
- apel funcție b) pentru expresie– 0.1p

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:

CodAutor	Titlu	CodCarte
A1	T1	1
A2	T4	4

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

- **b2** - {NrPagini}→{CodCarte} este satisfăcută – **0.25p**; **0.25p** explicație
  - {CodAutor}→{Domeniu} 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.

VARIANTA 2

SUBIECT Sisteme de operare

Barem:

1p – oficiu

Problema 1 (5p)

1p – a) Execută programul `a` cu argumentul `y`, dacă locaţia programului `a` se găseşte in `PATH`

1p – b) Se va afişa de o infinitate de ori, pentru că programul se reexecută la linia 6 sau 7.

1p – c) Se vor afişa o infinitate de caractere `z`

1p – d) Nicunul pentru că unul dintre cele două apeluri sistem `execl` şi `execlp` se vor executa cu succes şi vor suprascrie codul procesului înainte să se ajungă la linia 8.

1p – e) Se vor afişa o infinitate de caractere `y`

Problema 2 (4p)

1p – a) L va lua ca valori fiecare linie din fişierul a.txt, variabila W va lua ca valori fiecare secvenţe de litere din linia stocată în L

1p – b) După fiecare literă din variabila W se adaugă un spaţiu, rezultatul stocându-se in variabila C

1p – c) Se va afişa

```
fed cba
zyx rqp
```

1p – d) Se va afişa

```
def abc
xyz pqr
```