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

Schriftliche Abschlussprüfung 2026
Fachgebiet Informatik in deutscher Sprache

VARIANTE 1

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

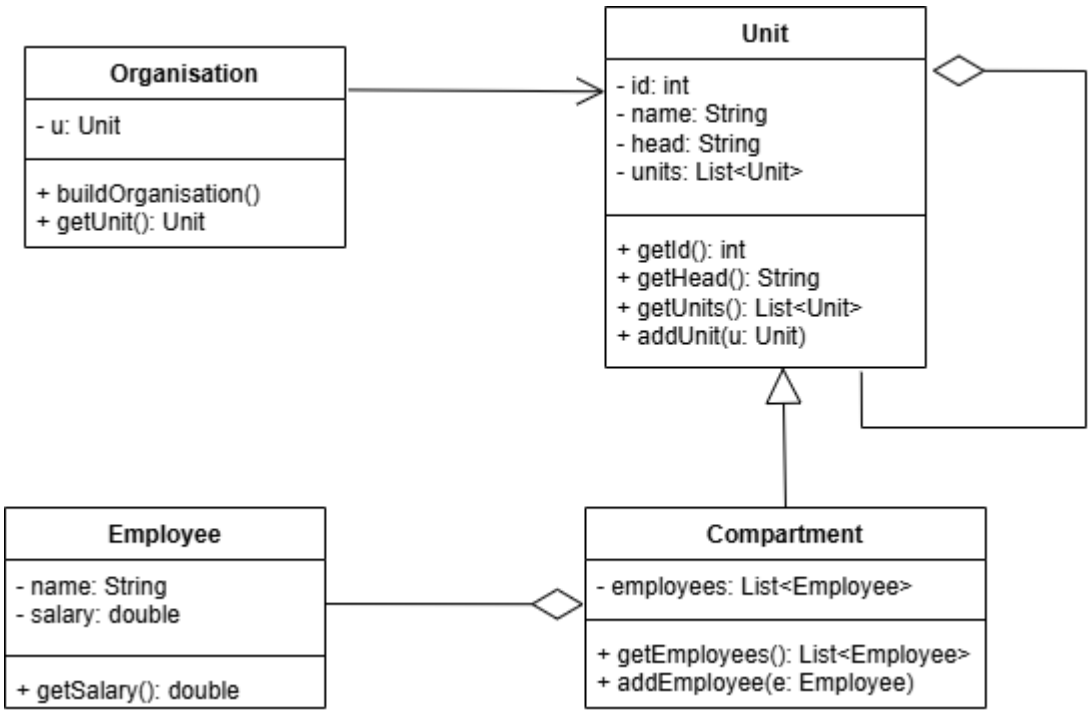
- **Geben Sie die verwendete Programmiersprache an.**
- **Die erforderlichen Implementierungen sind nicht verpflichtet, dynamisch zugewiesene Speicherbereiche freizugeben.**
- **Das Fehlen eines angemessenen Programmierstils (aussagekräftige Variablennamen, Einrückung des Codes, Kommentare, falls erforderlich, 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. **(3 Punkte)** Schreiben Sie eine Funktion in einer der Programmiersprachen **Python**, **C++**, **Java** oder **C#**, die als Parameter ein Array *v* mit reellen Zahlen erhält, das in absteigender Reihenfolge sortiert ist (stellt die Gehälter der Mitarbeiter in einer Abteilung dar), sowie eine reelle Zahl *Gehalt* (stellt das Gehalt eines neuen Mitarbeiters dar). Die Funktion soll *Gehalt* in das Array *v* einfügen, sodass das Array weiterhin in absteigender Reihenfolge der Gehälter sortiert bleibt. Verwenden Sie eine Hilfsfunktion mit iterativer Implementierung, die in Zeitkomplexität $O(\log_2 n)$ die Position bestimmt, an der *Gehalt* in das Array *v* eingefügt werden soll, wobei *n* die Länge des Arrays *v* ist. *Rekursive Lösungen werden nur teilweise bewertet.*
   Hinweis: Das Einfügen eines Elements an Position *k* in das Array *v* bedeutet, dass die Elemente an den Positionen *k, k+1, ..., n* um eine Position nach rechts verschoben werden, danach erhöht sich die Länge des Arrays um 1. ***Verwenden Sie keine vordefinierten Bibliotheksfunktionen, um ein Element an einer bestimmten Position in das Array einzufügen.***

2. **(1 Punkt)** Geben Sie die Zeitkomplexität im besten, durchschnittlichen und schlechtesten Fall für die Funktion aus Aufgabe **1** an. Begründen Sie Ihre Antwort.

3. **(5 Punkte)** Betrachten Sie das folgende UML-Diagramm, das die Klassen **Organisation** (stellt die hierarchische Struktur einer Organisation dar), **Unit** (Organisationseinheit), **Compartment** (Abteilung) und **Employee** (Mitarbeiter) enthält.

Hinweis: Die Konstruktoren der Klassen sind im Diagramm nicht dargestellt.



- Die Klasse **Organisation** stellt eine Organisation dar (in Form einer hierarchischen Struktur aus Organisationseinheiten). Die Methode **getUnit()** gibt das Attribut *u* der Klasse zurück.
    - Die Methode **buildOrganisation()** erstellt die hierarchische Struktur der Organisation. *Die Implementierung dieser Methode ist nicht erforderlich.*
- Die Klasse **Unit** stellt die hierarchische Struktur einer Organisationseinheit dar. Sie besitzt einen Identifikationscode (*id*), einen Namen (*name*) und einen Leiter, dessen Name im Attribut *head* gespeichert ist. Die Einheit hat mehrere direkt untergeordnete (Unter-)Einheiten, die im Attribut *units* gespeichert werden. Die Klasse stellt Methoden für den Zugriff auf *id*, *head* und *units* bereit sowie eine Methode **addUnit(*u*: Unit)**, die die Organisationseinheit *u* am Ende der Liste *units* hinzufügt.
- Die Klasse **Compartment** stellt eine Organisationseinheit dar, die keine weiteren direkt untergeordneten (Unter-)Einheiten besitzt. Sie speichert eine Liste von Mitarbeitern (**Employee**) im Attribut *employees*. Die Liste *employees* speichert die Mitarbeiter in absteigender Reihenfolge ihrer Gehälter. Die Klasse besitzt eine Methode **addEmployee(*e*: Employee)**, die einen Mitarbeiter zur sortierten Liste *employees* hinzufügt, sowie eine Methode **getEmployees()**, die die Liste der Mitarbeiter in der Abteilung zurückgibt.
- Die Klasse **Employee** stellt einen Mitarbeiter dar, der durch einen Namen (*name*) und ein Gehalt (*salary*) charakterisiert wird. Die Klasse besitzt eine Methode **getSalary()**, die das Attribut *salary* zurückgibt.

Hinweis: Die Organisationseinheiten (**Unit**) innerhalb der Organisation (**Organisation**) besitzen unterschiedliche Identifikationscodes (*id*).

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 entsprechend dem oben dargestellten Diagramm. Implementieren Sie nur den Konstruktor der Klasse **Compartment**.

b) Definieren Sie eine Funktion, die als Parameter ein Objekt *o* vom Typ **Organisation** und eine ganze Zahl *n* erhält und eine Liste zurückgibt, die alle Mitarbeiter enthält, die der Organisationseinheit mit dem *id* gleich *n* direkt oder indirekt unterstellt sind. Die zurückgegebene Liste ist leer, wenn die Organisation keine Organisationseinheit mit dem *id* gleich *n* besitzt.

c) Definieren Sie eine Funktion, die als Parameter ein Objekt *o* vom Typ **Organisation** erhält und eine Liste der Leiter der Abteilungen (**Compartment**) innerhalb der Organisation *o* zurückgibt.

d) Erstellen Sie ein Objekt *o* vom Typ **Organisation**, rufen Sie die Methode **buildOrganisation()** auf und anschließend die Funktionen aus **b)** und **c)** für die erstellte Organisation.

THEMA Datenbanken **VARIANTE 1**

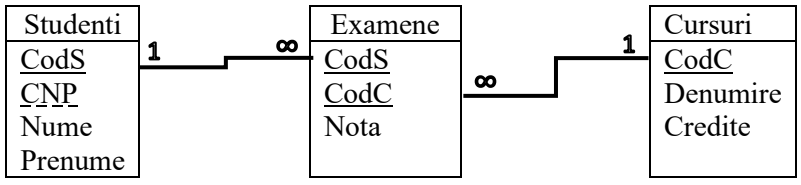
Aufgabe 1. (4 Punkte)

Ein Forschungsinstitut speichert Informationen über die Herbarien, die in seinen Forschungsaktivitäten verwendet werden. Die Herbarien werden in verschiedenen Lagerräumen des Institutsgebäudes aufbewahrt und von den Angestellten des Forschungsinstituts genutzt. Zur Verwaltung der Informationen über Angestellte, Lagerräume, Herbarien, Pflanzen und geografische Regionen wird eine relationale Datenbank verwendet:

- Ein Angestellter des Forschungsinstituts hat einen Code, einen Namen, eine E-Mail-Adresse (für jeden Angestellten eindeutig), ein Geburtsdatum, die Position, die er innerhalb des Forschungsinstituts hat, sowie ein Gehalt.
- Ein Lagerraum hat einen Code, eine Bezeichnung, eine Beschreibung und einen Angestellten, der für seine Verwaltung verantwortlich ist. Jeder Angestellte kann für die Verwaltung mehrerer Lagerräume verantwortlich sein, jedoch kann ein Lagerraum nur von einem einzigen Angestellten verwaltet werden.
- Eine geografische Region hat einen Code, eine Bezeichnung, eine Beschreibung und ein Land.
- Ein Herbarium hat einen Code, einen Titel, eine Beschreibung, ein Datum, an dem es in die Verwaltung des Forschungsinstituts übernommen wurde, und wird in einem bestimmten Lagerraum des Instituts aufbewahrt. Jeder Lagerraum kann mehrere Herbarien enthalten, jedoch wird ein Herbarium in genau einem Lagerraum aufbewahrt.
- Eine Pflanze hat einen Code, einen wissenschaftlichen Namen, einen gebräuchlichen Namen, die Klasse, die Ordnung sowie einen Wert, der angibt, ob sie geschützt ist oder nicht. Ein Herbarium kann mehrere Pflanzen enthalten, und eine Pflanze kann in mehreren Herbarien vorkommen. Jede Pflanze kann einem bestimmten Herbarium nur einmal zugeordnet werden. Für jedes Paar aus einer Pflanze und einem Herbarium (zum Beispiel das Paar bestehend aus der Pflanze p1 und dem Herbarium i1) werden außerdem das Datum gespeichert, an dem die Pflanze dem Herbarium hinzugefügt wurde, sowie die geografische Region, aus der die Pflanze gesammelt wurde.

Erstellen Sie ein relationales Schema in BCNF für die Datenbank, wobei die Primärschlüssel, Kandidatenschlüssel und Fremdschlüssel streng hervorgehoben werden. Erstellen Sie das Schema in einer der im folgenden Beispiel angegebenen Formen dar:

* Beispiel für die Tabellen Studenti, Cursuri und Examene:
V1. Diagramm mit Tabellen, Primärschlüssel durch eine durchgezogene Linie unterstrichen, Kandidatenschlüssel durch eine gestrichelte Linie unterstrichen, Beziehungen direkt zwischen den Fremdschlüsseln und den entsprechenden Primär-/Kandidatenschlüsseln (z. B. Beziehung zwischen der CodS-Spalte in Examene und der CodS-Spalte in Studenti).



V2.
Studenti[CodS, CNP, Nume, Prenume]
Cursuri[CodC, Denumire, Credite]
Examene[CodS, CodC, Nota]
Primärschlüssel sind durch eine durchgezogene Linie und Kandidatenschlüssel durch eine gestrichelte Linie unterstrichen. {CodS} ist Fremdschlüssel in Examene, der auf {CodS} in Studenti verweist. {CodC} ist Fremdschlüssel in Examene, der auf {CodC} in Cursuri verweist.

Aufgabe 2. (5 Punkte)

Wir betrachten die folgenden Relationen aus einer Datenbank, die Informationen über Kunstwerke und temporäre Ausstellungen (expoziții) eines Museums speichert:

Expozitii[CodExpozitie, Titlu, Descriere, DataVernisajului, NumarSala]
PerioadeArtistice[CodPerioada, Denumire]
Artisti[CodArtist, Nume, DataNasterii, *CodPerioada*]
OpereDeArta[CodOperaDeArta, Titlu, AnulCrearii, *CodArtist, CodExpozitie*, TipOperaDeArta]
Primärschlüssel sind unterstrichen. Fremdschlüssel sind kursiv geschrieben und haben denselben Namen wie die Spalten, auf die sie sich beziehen.

a. Schreiben Sie eine SQL-Abfrage, die alle Ausstellungen zurückgibt (Titel der Ausstellung, Datum der Vernissage, Saalnummer), die mindestens ein Kunstwerk enthalten, das vom Künstler mit dem Namen „Pablo Picasso“ geschaffen wurde, aber kein Kunstwerk enthalten, das vom Künstler mit dem Namen „Claude Monet“ geschaffen wurde. Entfernen Sie Duplikate.
b. Es werden die folgenden Instanzen der Relationen PerioadeArtistice, Artisti und OpereDeArta gegeben:

PerioadeArtistice:

Cod Perioada	Denumire
1	Cubism
2	Impresionism
3	Postimpresionism

Artisti:

Cod Artist	Nume	DataNasterii	Cod Perioada
1	Pablo Picasso	25.10.1881	1
2	Claude Monet	14.11.1840	2
3	Vincent van Gogh	30.03.1853	3
4	Edgar Degas	19.07.1834	2

OpereDeArta:

CodOpera DeArta	Titlu	Anul Crearii	Cod Artist	Cod Expozitie	TipOperaDeArta
1	Guernica	1937	1	1	Pictura
2	Jacqueline with Flowers	1954	1	1	Pictura
3	Water lilies	1919	2	2	Pictura
4	The Little Fourteen-Year-Old Dancer	1880	4	2	Sculptura
5	The Ballet Class	1874	4	2	Pictura
6	Sunflowers	1888	3	3	Pictura
7	The Starry Night	1889	3	2	Pictura
8	Cafe Terrace at Night	1888	3	1	Pictura

b1. Geben Sie das Ergebnis der Auswertung der folgenden Abfrage für die angegebenen Instanzen an. Geben Sie ausschließlich die Werte der Tupel und Spaltennamen im Ergebnis an, ohne alle Schritte der Abfrageauswertung darzustellen.

```
SELECT A.Nume, PA.Denumire, COUNT (O.CodOperaDeArta) as Nr
FROM Artisti A
        INNER JOIN PerioadeArtistice PA ON A.CodPerioada = PA.CodPerioada
        INNER JOIN OpereDeArta O ON O.CodArtist = A.CodArtist
GROUP BY A.CodArtist, A.Nume, PA.Denumire
HAVING COUNT(O.CodOperaDeArta) >=
  (SELECT AVG(T.Nr)
   FROM         (SELECT A1.CodArtist, COUNT(*) AS Nr
                 FROM Artisti A1
                        INNER JOIN PerioadeArtistice PA1 ON A1.CodPerioada = PA1.CodPerioada
                        INNER JOIN OpereDeArta O1 ON O1.CodArtist = A1.CodArtist
                 GROUP BY A1.CodArtist
                        ) T
          )
```

b2. Begründen Sie für jede der folgenden funktionalen Abhängigkeiten, ob diese für die Daten in der Instanz OpereDeArta gilt oder nicht:

- { CodOperaDeArta } → { CodArtist }
- { CodExpozitie } → { CodOperaDeArta }.

VARIANTE 1

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 erforderlichen Include-Dateien vorhanden sind und dass die Systemaufrufe `fork` und `write` erfolgreich ausgeführt werden.

```
 1  int main() {
 2    int i;
 3    char buf[3] = {'L','0','\n'};
 4    for (i = 0; i < 3; i++) {
 5      buf[1] = '0' + i;
 6      write(i%2+1, buf, 3);
 7      fork();
 8    }
 9    write(1, "END\n", 4);
10    for (i = 0; i < 3; i++)
11      wait(NULL);
12    return 0;
13  }
```

a) Wie oft wird die Zeile `END` bei der Ausführung von `./a.out` angezeigt? Begründen Sie Ihre Antwort.

b) Wie oft erscheint jede der Zeilen `L0`, `L1` und `L2` in der Datei `out` nach der Ausführung von `./a.out > out`? Begründen Sie Ihre Antwort.

c) Wie viele Prozesse (außer dem Startprozess) werden insgesamt während der Ausführung erstellt? Begründen Sie Ihre Antwort.

d) Beschreiben Sie detailliert die Funktionsweise der Schleife in den Zeilen `10-11` und geben Sie an, auf wie viele direkte Kinder der Startprozess wartet. Begründen Sie Ihre Antwort.

e) Wie oft werden die Zeilen `L (L0+L1+L2)` bei der Ausführung von `./a.out` insgesamt angezeigt, wenn Zeile `6` direkt nach Zeile `7` verschoben wird? Begründen Sie Ihre Antwort.

Aufgabe 2. (4 Punkte) Beantworten Sie die folgenden Fragen zur Ausführung des untenstehenden UNIX-Shell-Skripts `./a.sh`, wobei davon auszugehen ist, dass dieses die angegebene Datei `f.txt` verwendet. Hinweis: Die Zeilennummerierung bezieht sich NICHT auf den Inhalt der Dateien.

```
1  #!/bin/bash
2  while read NAME GRADE; do
3    if [ $GRADE -ge 5 ]; then
4      echo "$NAME"
5    fi
6  done < f.txt | sort | uniq -c | sort -nr | head -n 1
```

`f.txt`

```
1  Anca 7
2  Radu 9
3  Anca 8
4  Radu 2
5  Anca 3
```

a) Was wird nur von der `while`-Schleife (Zeilen `2-6`, vor dem ersten `| sort`) angezeigt? Begründen Sie Ihre Antwort.

b) Welche Wirkung hat der Ausdruck `sort | uniq -c`? Begründen Sie Ihre Antwort.

c) Was wird vom gesamten Skript angezeigt? Begründen Sie Ihre Antwort.

d) Wie ändert sich das Ergebnis, wenn der Befehl `sort` vor `uniq -c` entfernt wird? Begründen Sie Ihre Antwort.

BAREM INFORMATICĂ
VARIANTA 1

Subiect Algoritmică și Programare

Oficiu – **1p**

Cerința **1. – 3p**

Funcția de bază – **0.4p** din care
- signatura – 0.1p
- implementare – 0.3p

Funcția auxiliară – **2.6p** din care
- signatura – 0.1p
- implementare iterativă funcție auxiliară având complexitate timp $O(\log_2 n)$ – 2.5p
  * soluție recursivă având complexitate timp $O(\log_2 n)$ – 1p
  * soluție iterativă având complexitate timp $O(n)$ – 0.5p

Cerința **2. – 1p**
- caz favorabil **0.3p** din care
  - complexitate – 0.15
  - justificare – 0.15
- caz mediu **0.4p** din care
  - complexitate – 0.2
  - justificare – 0.2
- caz defavorabil **0.3p** din care
  - complexitate – 0.15
  - justificare – 0.15

Cerința **3.a) – 1.7p**

Definirea clasei Organisation – **0.2p** din care
- atribut – 0.1
- metode **buildOrganisation**, **getUnit** – 0.1

Definirea clasei Unit – **0.5p** din care
- atribute – 0.2
- constructor – 0.1
- metode **getId, getHead, getUnits, addUnit** – 0.2

Definirea clasei Compartment– **0.7p** din care
- relația de moștenire – 0.2
- atribut – 0.1
- **constructor (a1)** – 0.3
- metode **getEmployees, addEmployee** – 0.1

Definirea clasei Employee – **0.3p** din care
- atribute– 0.2
- metoda **getSalary** – 0.1

Funcția **3.b)** – **2p**
- signatura – **0.1p**
- implementare funcție – **1.8p**
- returnare rezultat – **0.1p**

Funcția **3.c)** – **1p**
- signatura – **0.1p**
- implementare funcție – **0.8p**
- returnare rezultat – **0.1p**

Funcția principală **3.d)** – **0.3p**
- construire obiect **o** – 0.1p
- apel metoda **buildOrganisation**() – 0.1p
- apel funcții b) și c) – 0.1p

BAREM INFORMATICĂ
VARIANTA 1

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:

Nume	**Denumire**	**Nr**
Pablo Picasso	Cubism	2
Vincent van Gogh	Postimpresionism	3
Edgar Degas	Impresionism	2

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

- **b2 -** { CodOperaDeArta } → { CodArtist } este satisfăcută – **0.25p**; **0.25p** explicație
  - { CodExpozitie } → { CodOperaDeArta } 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 INFORMATICA
VARIANTA 1

Sisteme de Operare

1p – oficiu

Problema 1 (5p)

1p – a) 8 = 2^3 procese

1p – b) L0 o dată, L1 niciodată, L2 de patru ori

1p – c) 7 procese fii, 2^3 fără procesul inițial

1p – d) se așteaptă 3 procese fiu, dacă acestea nu există, wait returnează -1; procesul inițial așteaptă 3 procese copil

1p – e) Liniile L se afișează în total de 14 ori (L0×2 + L1×4 + L2×8)

Problema 2 (4p)

1p – a) Anca, Radu, Anca (în această ordine, câte una pe linie); numele doar pentru liniile cu nota >=5

1p – b) `sort` sortează în odine alfabetică liniile primite, iar `uniq` elimină liniile consecutive identice și afișează de câte ori apare fiecare, folosind opțiunea `-c`

[opțional: Aici obținem: 2 Anca şi 1 Radu (numărul precedă numele);]

1p – c) 2 Anca - `sort -nr` ordonează descrescător numeric după numărul din față, `head -n 1` păstrează prima linie

1p – d) Neavând linii consecutive identice, uniq -c produce 1 Anca, 1 Radu, 1 Anca, rezultatul final fiind 1 Anca