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

# Schriftliche Abschlussprüfung 2026
# 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.

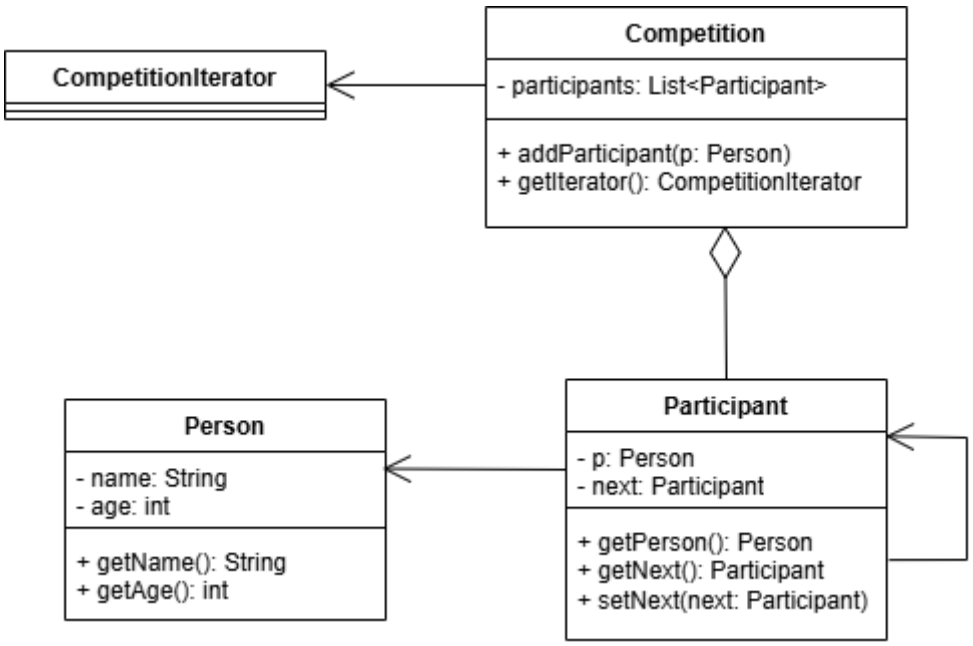
**THEMA Algorithmen und Programmierung**

- **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#).**

Teilnehmer aus 7 Alterskategorien sind für einen Wettbewerb registriert: $c_0$ (5-9 Jahre), $c_1$ (10-19 Jahre), $c_2$ (20-29 Jahre), $c_3$ (30-39 Jahre), $c_4$ (40-49 Jahre), $c_5$ (50-59 Jahre) und $c_6$ (60-69 Jahre). Die Teilnehmer werden nach Alterskategorien in der Reihenfolge $c_0$, $c_1$, …, $c_6$ gespeichert; innerhalb jeder Alterskategorie werden die Teilnehmer alphabetisch geordnet.

1. **(2.5 Punkte)** Schreiben Sie eine Funktion in einer der Programmiersprachen **Python**, **C++**, **Java** oder **C#**, die als Parameter eine Liste *lst* von 7 Listen mit alphabetisch geordneten Zeichenketten (die Namen der Teilnehmer der Altersgruppe $c_i$ werden in alphabetischer Reihenfolge an Position *i* in der Liste *lst* gespeichert), eine natürliche Zahl *Alter* und eine Zeichenkette *Name* (die das Alter und den Namen eines Teilnehmers darstellen) erhält und den Namen des Teilnehmers in die Liste der entsprechenden Alterskategorie einfügt, sodass diese Liste weiterhin alphabetisch geordnet bleibt. ***Verwenden Sie keine vordefinierten Bibliotheksfunktionen, um ein Element an einer bestimmten Position in die Liste 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. **(5.5 Punkte)** Betrachten Sie das folgende UML-Diagramm, das die Klassen **Competition** (Wettbewerb), **Participant** (Teilnehmer), **Person** (Person) und **CompetitionIterator** (Wettbwert-Iterator) enthält.

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




- Die Klasse **Competition** besitzt ein Attribut *participants*, das eine Liste mit der Größe der Anzahl der Alterskategorien ist. Die Teilnehmer der Alterskategorie $c_i$ werden an Position $i$ in der Liste *participants* gespeichert. Die Klasse besitzt folgende Methoden:
  - **addParticipant(*p*: Person)**, die die Person *p* zur Liste hinzufügt, die ihrer Altersgruppe entspricht.
  - **getIterator()**, die einen Iterator zum Durchlaufen der Wettbewerbsteilnehmer zurückgibt.
- Die Klasse **Participant** stellt eine Person dar, die am Wettbewerb in einer bestimmten Alterskategorie teilnehmen. Die Teilnehmer jeder Alterskategorie werden alphabetisch nach Namen geordnet gespeichert.
- Die Klasse **Person** speichert den Namen (*name*) und das Alter (*age*) einer am Wettbewerb teilnehmenden Person. Die Klasse besitzt Methoden, die ihre Attribute zurückgeben. Das Alter einer Person ist eine natürliche Zahl im Bereich [5–69].
- Die Klasse **CompetitionIterator** definiert einen Iterator für die am Wettbewerb teilnehmenden Personen, sortiert nach ihren Alterskategorien ($c_0$, $c_1$, …, $c_6$), und innerhalb jeder Alterskategorie alphabetisch nach Namen.

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

**a)** Deklarieren Sie die Klassen **Competition**, **Person** und **Participant** sowie deren Attribute und Methoden entsprechend dem oben dargestellten Diagramm. Implementieren Sie nur die folgenden Methoden:
- den Konstruktor der Klasse **Competition**.
- die Methode **setNext(**p**: Participant)** in der Klasse **Participant**.

**b)** Definieren Sie die Klasse **CompetitionIterator**, einschließlich der für einen Iterator spezifischen Attribute und Methoden. Implementieren Sie alle Methoden der Klasse **CompetitionIterator**. Die Attribute und die implementierten Methoden müssen einen Speicherplatzbedarf von $\theta(1)$ verwenden. Die Methoden müssen eine Zeitkomplexität von $\theta(1)$ haben.

**c)** Definieren Sie eine Funktion, die als Parameter ein Objekt *c* vom Typ **Competition** erhält und unter Verwendung der in **b)** definierten Klasse, die Wettbewerbsteilnehmer in der Reihenfolge ihrer Alterskategorien und innerhalb jeder Alterskategorie alphabetisch nach Namen ausgibt.

**d)** Definieren Sie eine Funktion, die als Parameter ein Objekt *c* vom Typ **Competition** und eine Zeichenkette *name* erhält und die Alterskategorie zurückgibt, in der die Person mit dem Namen *name* am Wettbewerb teilnimmt, oder eine leere Zeichenkette, falls es keinen Teilnehmer mit dem Namen *name* gibt. Wenn es mehrere Teilnehmer mit demselben Namen gibt, wird die erste Altersklasse zurückgegeben, in der sich ein Teilnehmer mit dem angegebenen Namen befindet.

**e)** Erstellen Sie ein Objekt *o* vom Typ **Competition**; fügen Sie diesem Objekt 2 Alterskategorien und 4 Teilnehmer hinzu, davon jeweils 2 in jeder Alterskategorie. Rufen Sie die Funktion aus **c)** auf. Rufen Sie für einen der Teilnehmer die Funktion aus **d)** auf.

**THEMA Datenbanken** **VARIANTE 2**

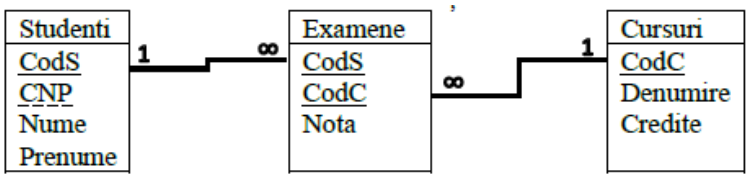
**Aufgabe 1. (4 Punkte)**

Ein Medienunternehmen verwendet eine relationale Datenbank zur Verwaltung seiner Aktivitäten.

- Jede Zeitung hat einen eindeutigen Identifikationscode, einen Namen (der für jede Zeitung eindeutig ist), ein Gründungsjahr und eine Veröffentlichungsfrequenz (dies kann ein einfaches Attribut in Form einer Zeichenkette sein; z. B. *täglich, wöchentlich, monatlich*). Jede Zeitung wird von einem einzigen Zentralsitz verwaltet, jedoch kann ein Zentralsitz mehrere Zeitungen verwalten.
- Ein Zentralsitz hat einen Identifikationscode und eine Telefonnummer (die für jeden Zentralsitz eindeutig ist).
- Jeder Journalist hat einen Identifikationscode, einen Nachnamen, einen Vornamen, ein Geburtsdatum, ein Geschlecht und ein Einstiegsjahr. Ein Journalist kann Kooperationsverträge mit mehreren Zeitungen haben. Eine Zeitung kann mit mehreren Journalisten zusammenarbeiten. Ein Journalist kann jedoch nur einen einzigen Kooperationsvertrag mit einer bestimmten Zeitung haben. Für jeden Kooperationsvertrag werden folgende Daten gespeichert: die Zeitung, der Journalist, das Anfangsdatum des Vertrags, die Vertragsdauer (eine ganze Zahl, die die Anzahl der Monate angibt) sowie die Vergütung.
- Ein Artikel hat einen Identifikationscode, einen Titel und wird von einem Journalisten für eine der Zeitungen geschrieben, mit denen er zusammenarbeitet.
- Ein Thema hat einen Identifikationscode, eine Bezeichnung (die pro Thema eindeutig ist; z. B. *Wirtschaft*, *Sport*) und eine Beschreibung. Ein Artikel kann mehrere Themen behandeln, und ein Thema kann in mehreren Artikeln vorkommen.

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)**

Gegeben seien folgende Relationenschemata:
CreatoriContinut[CodCreator, Nume, Tara, AnulNasterii]
Videoclipuri[CodVideo, Titlu, Categorie, Durata, *CodCreator*]
Platforme[CodPlatforma, Denumire]
VideoclipuriPlatforme[*CodVideo, CodPlatforma*, Monetizare, AnPublicare]
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 den Code und den Namen jedes Content-Erstellers (creator de conținut) zurückgibt, der die folgenden zwei Bedingungen erfüllt:
1. er hat mindestens 2 Videos im selben Jahr veröffentlicht und
2. er hat mindestens ein Video, das auf der Plattform „StreamNow“ veröffentlicht wurde.
Ein Video gilt als *veröffentlicht*, wenn es auf einer Plattform erscheint.

Beispiel für die Bedingung 1:
- der Ersteller C1 hat die Videos veröffentlicht: V1 im Jahr 2020, V2 im Jahr 2020 und V3 im Jahr 2021; C1 erfüllt die Bedingung 1.
- der Ersteller C2 hat die Videos veröffentlicht: V4 im Jahr 2020, V5 im Jahr 2021, V6 im Jahr 2022; C2 erfüllt die Bedingung 1 nicht.

**b.** Es werden die folgenden Instanzen der Relationen CreatoriContinut, Videoclipuri, Platforme und VideoclipuriPlatforme gegeben:

**CreatoriContinut**

| Cod Creator | Nume | Tara | AnulNasterii |
|---|---|---|---|
| 1 | n1 | Romania | 2005 |
| 2 | n2 | Romania | 2005 |
| 3 | n3 | Franta | 2005 |

**Platforme**

| Cod Platforma | Denumire |
|---|---|
| 1 | p1 |
| 2 | p2 |
| 3 | p3 |

**Videoclipuri**

| Cod Video | Titlu | Categorie | Durata | Cod Creator |
|---|---|---|---|---|
| 1 | v1 | c1 | 3 | 1 |
| 2 | v2 | c2 | 3 | 1 |
| 3 | v3 | c1 | 4 | 1 |
| 4 | v4 | c1 | 2 | 2 |
| 5 | v5 | c1 | 3 | 2 |
| 6 | v6 | c2 | 4 | 1 |
| 7 | v7 | c2 | 2 | 1 |

**VideoclipuriPlatforme**

| Cod Video | Cod Platforma | Monetizare | An Publicare |
|---|---|---|---|
| 1 | 2 | T | 2020 |
| 2 | 2 | F | 2020 |
| 3 | 3 | T | 2021 |
| 4 | 3 | T | 2020 |
| 5 | 3 | T | 2021 |
| 6 | 1 | F | 2020 |
| 7 | 1 | F | 2021 |

**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 vp.CodPlatforma, MIN(vp.AnPublicare) AnMin
FROM CreatoriContinut c INNER JOIN Videoclipuri v ON c.CodCreator = v.CodCreator
        INNER JOIN VideoclipuriPlatforme vp ON v.CodVideo = vp.CodVideo
WHERE c.Tara = 'Romania'
GROUP BY vp.CodPlatforma
HAVING COUNT(vp.CodVideo) >
        (SELECT COUNT(*)
        FROM VideoclipuriPlatforme vp2 INNER JOIN Platforme p2
           ON vp2.CodPlatforma = p2.CodPlatforma
        WHERE p2.Denumire = 'p2')
```

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

- $\{Categorie\} \rightarrow \{Titlu\}$
- $\{CodCreator\} \rightarrow \{CodVideo\}$

## VARIANTE 2

**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 `wait` erfolgreich ausgeführt werden. Wir weisen darauf hin, dass das Makro `WEXITSTATUS(st)` den vom Kindprozess über `exit` übermittelten Exit-Code (eine 8-Bit-Ganzzahl im Bereich von 0 bis 255) abruft.

```
 1  int main(int argc,
 2           char** argv){
 3    int i, st, total = 0;
 4    for (i=1; i<argc; i++) {
 5      if (fork() == 0) {
 6        exit(strlen(argv[i]));
 7      }
 8      wait(&st);
 9      total += WEXITSTATUS(st);
10    }
11    printf("%d\n", total);
12    return 0;
13  }
```

a) Was wird bei der Ausführung von `./a.out ab cde f` angezeigt? Begründen Sie Ihre Antwort.

b) Wie viele Kindprozesse erzeugt `./a.out ab cde f`? Und wie viele bei `./a.out` (ohne Argumente), und was wird in diesem Fall angezeigt? Begründen Sie Ihre Antwort.

c) Erläutern Sie ausführlich die Funktionsweise von Zeile `6` und wie der von `exit` übergebene Wert zum Elternprozess gelangt.

d) Erläutern Sie ausführlich die Funktionsweise der Zeilen `8-9`.

e) Was wird angezeigt, wenn `./a.out` mit einem Argument aus 260 Zeichen aufgerufen wird? Begründen Sie Ihre Antwort.

**Aufgabe 2. (4 Punkte)** Beantworten Sie die folgenden Fragen zur Ausführung des untenstehenden UNIX-Shell-Skripts `./a.sh` unter Berücksichtigung der angegebenen Datei `d.txt`. Hinweis: Die Zeilennummerierung bezieht sich NICHT auf den Inhalt der Dateien.

```
 1  #!/bin/bash
 2  while read L; do
 3    N=0;
 4    for X in `echo "$L" | sed -E "s/(.)/\1 /g"`; do
 5         N=`expr $N + 1`
 6    done
 7    if [ $N -ge 4 ]; then
 8         echo "$L" | sed -E "s/^(..).*$/\1/g"
 9    else
10         echo "$N"
11    fi
12  done < d.txt
```

d.txt

```
1  hello
2  ab
3  sun
4  code
```

a) Geben Sie an, was das Skript für Zeile `1` bzw. `2` der Datei `d.txt` ausgibt. Begründen Sie Ihre Antwort.

b) Erläutern Sie die Funktionsweise des Befehls `sed` in Zeile 8.

c) Geben Sie an, welche Zahlenwerte ausgegeben werden. Begründen Sie Ihre Antwort.

d) Erläutern Sie ausführlich die Funktionsweise der Zeilen `3-6`.

# BAREM INFORMATICĂ
# VARIANTA 2

## Subiect Algoritmică și Programare

Oficiu – **1p**

Cerința **1. – 2.5p**
- signatura – 0.2p
- implementare – **2.3p** din care
  - căutarea listei corespunzătoare vârstei *varsta* – 0.3p
  - adaugarea numelui *nume* în lista identificată – 2p

Cerința **2. – 1p**
- caz favorabil **0.2p** din care
  - complexitate – 0.1
  - justificare – 0.1
- caz mediu **0.4p** din care
  - complexitate – 0.2
  - justificare – 0.2
- caz defavorabil **0.4p** din care
  - complexitate – 0.2
  - justificare – 0.2

Cerința **3.a) – 1p**

Definirea clasei Competition – **0.4p** din care
- atribut – 0.1
- **constructor (a1)** – 0.2p
- metode **addParticipant**, **getIterator** – 0.1

Definirea clasei Participant – **0.3p** din care
- atribute – 0.1
- metode **getPerson, getNext** – 0.1
- **metoda setNext (a2)** – 0.1p

Definirea clasei Person– **0.3p** din care
- atribute – 0.1
- constructor – 0.1
- metode **getName, getAge** – 0.1

Cerința **3.b) – 2.5p**

Definirea clasei CompetitionIterator
- atribute – 0.5
- constructor – 0.5
- metode specifice - 1.5

Funcția **3.c)** – **0.7p**
- signatura – **0.1p**
- parcurgere – **0.6p**

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

Funcția principală **3.e)** – **0.3p**
- construire obiect **o** – 0.1p
- adăugare 2 categorii de vârstă și 4 participanți – 0.1p
- apel funcții c) și d) – 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:

| CodPlatforma | AnMin |
|---|---|
| 3 | 2020 |

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

- **b2** - {Categorie}→{Titlu} nu este satisfăcută – **0.25p**; **0.25p** explicație
  - {CodCreator}→{CodVideo} 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 2

## Sisteme de Operare

**1p** – oficiu

Problema 1 (5p)

**1p** – a) 6 = strlen("ab") + strlen("cde") + strlen("f") = 2 + 3 + 1

**1p** – b) 3 fii pentru ./a.out ab cde f; 0 fii pentru ./a.out fără argumente, și se afișează 0.

**1p** – c) Linia 5 termină procesul fiu cu un cod de ieșire egal cu lungimea argumentului argv[i]; valoarea ajunge la părinte prin starea de terminare citită de wait.

**1p** – d) `wait(&st)` aşteaptă terminarea fiului curent şi pune starea lui în st; `WEXITSTATUS(st)` extrage codul de ieşire (lungimea), care se adună în `total`.

**1p** – e) Afişează 4; nu va afișa 260, deoarece codul de ieşire ocupă doar 8 biţi (0–255);

Problema 2 (4p)

**1p** – a) (linia 1: hello) → he, (linia 2: ab) → 2. …+ justificare

**1p** – b) Comanda sed înlocuiește fiecare linie (aici vom avea una singură) cu primele două caractere ale ei, ignorând restul textului.

**1p** – c) se va afișa 2 și 3 (pentru liniile 2 și 3 din `d.txt`)

**1p** – d) Liniile 3-6 calculează în N **lungimea** liniei curente