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

Schriftliche Abschlussprüfung
Fachgebiet Informatik in deutscher Sprache

VARIANTE 1

BEMERKUNG.

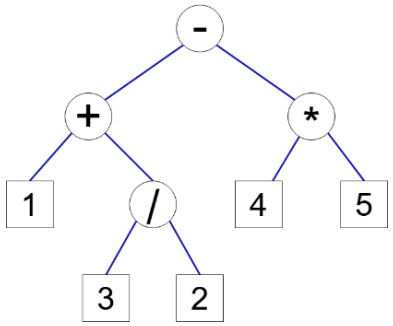
- Alle Prüfungsthemen sind verpflichtend. Bei allen Prüfungsthemen müssen vollständige Lösungen angegeben werden.
- Die Mindestnote für das Bestehen der Abschlussprüfung ist 5.00.
- Die Arbeitszeit beträgt 3 Stunden.

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#).**

Ein arithmetischer Ausdruck kann auf folgende Weise als Binärbaum dargestellt werden. Wenn der Ausdruck aus einem einzigen Operanden besteht, hat der entsprechende Binärbaum einen einzigen Knoten (den Wurzelknoten), der diesen Operanden als Information enthält. Hat der Ausdruck die Form **E1** *op* **E2**, wobei *op* ein binärer Operator ist und **E1** und **E2** arithmetische Ausdrücke sind, wird er mit einem Binärbaum assoziiert, dessen Wurzelknoten mit dem Operator *op* bezeichnet ist, der linke Teilbaum ist der mit dem Ausdruck **E1** assoziierte Binärbaum und der rechte Teilbaum ist der mit dem Ausdruck **E2** assoziierte Binärbaum.
Durch Postorder-Traversal des mit einem arithmetischen Ausdruck assoziierten Binärbaums erhält man die *Postfix-Notation* des Ausdrucks. **Zum Beispiel** für den Ausdruck (1+3/2)-4*5

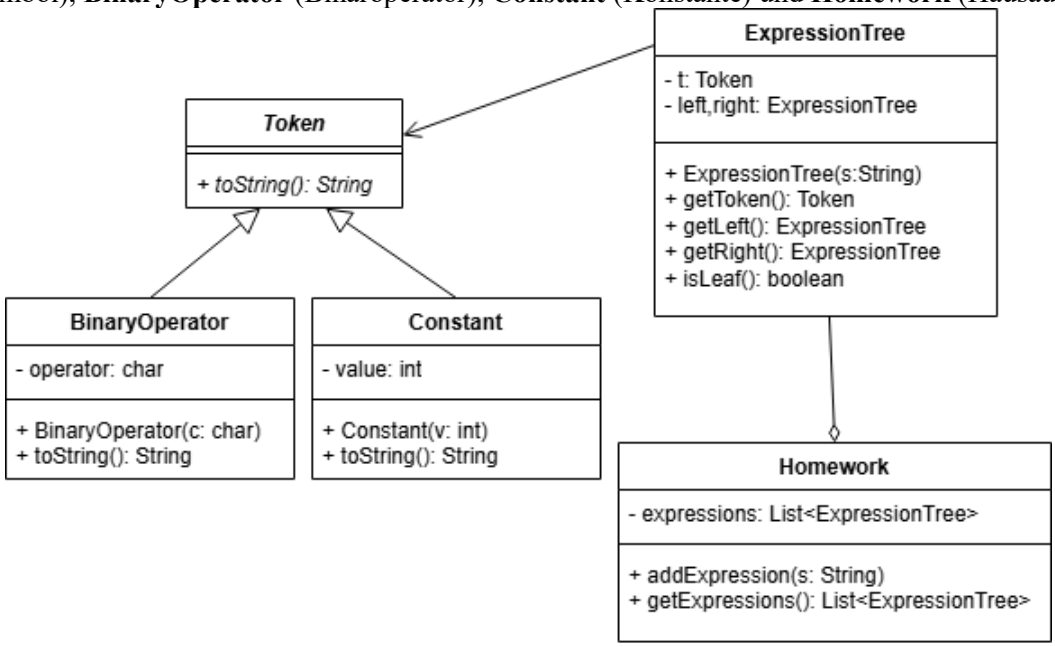
- Der entsprechende Binärbaum ist



- die Postfix- Notation ist '132/+45*-'.

1. **(3.5 Punkte)** Schreiben Sie eine Funktion in einer der Programmiersprachen **Python, C++, Java** oder **C#**, die als Parameter eine Zeichenkette *s* mit bis zu 100 Zeichen erhält, die die **korrekte** *Postfix-Notation* eines arithmetischen Ausdrucks darstellt. Der Ausdruck enthält die binären Operatoren ‚+' (Addition), ‚-' (Subtraktion), ‚*' (Multiplikation), ‚/' (Division) und als Operanden einstellige natürliche Zahlen ungleich Null. Die Funktion wird den Wert des arithmetischen Ausdrucks zurückgeben. Der Algorithmus muss die Zeitkomplexität θ(n), haben, wobei *n* die Länge der Zeichenkette *s* ist. Lösungen, die die geforderte Komplexität nicht erfüllen, werden teilweise bewertet. **Beispiel:** für die Zeichenkette '132/+45*-' wird der Wert **-17,5** zurückgegeben. Hilfsfunktionen können implementiert werden.
2. **(1 Punkt)** Geben Sie die Zeitkomplexität im *besten*, *mittleren* und *schlechtesten* Fall für die Funktion aus **Aufgabe 1.** an. Begründen Sie Ihre Antwort.

3. **(4.5 Punkte)** Gegeben sei das folgende UML-Diagramm, das die folgenden Klassen enthält, **ExpressionTree** (stellt den Binärbaum dar, der mit einem korrekten arithmetischen Ausdruck assoziiert ist), ***Token*** (Symbol), **BinaryOperator** (Binäroperator), **Constant** (Konstante) und **Homework** (Hausaufgabe).



- Die Klasse **ExpressionTree** stellt den Binärbaum dar, der mit einem arithmetischen Ausdruck assoziiert ist, dessen Operanden Konstanten (natürliche Zahlen ungleich Null) und Operatoren binär (+, -, *, /) sind.
    - Der Konstruktor der Klasse **ExpressionTree** nimmt als Parameter eine nicht-Null Zeichenkette *s*, die einen korrekten arithmetischen Ausdruck mit Klammern, binären Operatoren und Konstanten als Operanden darstellt, und erzeugt den entsprechenden binären Baum. D*ieser Konstruktor muss nicht implementiert werden.*
    - Die Methode **isLeaf()** gibt true zurück, wenn der Baum aus einem einzigen Knoten besteht (entspricht einem arithmetischen Ausdruck, der einen einzigen Operanden enthält), und andernfalls false.
    - Die Methode **getToken()** gibt die in der Wurzel des Binärbaums **A** gespeicherten Info zurück, die mit einem nicht leeren Ausdruck assoziiert ist, und die Methoden **getLeft()** und **getRight()** geben die linken bzw. rechten Teilbäume des Baums **A** zurück.
- Die abstrakte Klasse **Token** repräsentiert die Information (binärer Operator oder konstanter Operand), die in einem Baumknoten gespeichert werden können. Die Klasse stellt eine abstrakte **toString()**-Methode bereit, die eine mit dem Symbol assoziierte Zeichenkette zurückgibt: ist das Symbol ein binärer Operator, wird das Zeichen zurückgegeben, das den Operator repräsentiert (z. B. '+'); ist das Symbol eine Konstante, wird der in eine Zeichenkette umgewandelte Wert der Konstante zurückgegeben (z. B. '123').
- Die Klasse **Homework** speichert eine Liste *expressions* mit arithmetischen Ausdrücken, die ein Schüler als Hausaufgabe erhalten hat. Die Methode **addExpression(s: String)** der Klasse **Homework** nimmt als Parameter eine Zeichenkette *s*, die einen korrekten arithmetischen Ausdruck repräsentiert, und fügt den aus *s* konstruierten Binärbaum an das Ende der Liste *expressions* an. Die Methode **getExpressions()** gibt die Liste der Binärbäume zurück, die mit den als Hausaufgabe erhaltenen arithmetischen Ausdrücken verbunden sind.

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) Konstruktor der Klasse **BinaryOperator**. Das Attribut *operator* muss einen binären arithmetischen Operator repräsentieren (+, -, *, /). Der Konstruktor muss die Bedingung durchsetzen.
a2) Die Methode **isLeaf()** der Klasse **ExpressionTree**.
a3) Die Methode **addExpression(s: String)** der Klasse **Homework**.

b) Implementieren Sie eine Funktion, die als Parameter ein Objekt *e* vom Typ **ExpressionTree** annimmt und die Zeichenkette zurückgibt, die die Postfix-Notation des arithmetischen Ausdrucks darstellt, der mit dem in *e* gespeicherten Baum assoziiert ist.

c) Erzeugen Sie ein Objekt *h* vom Typ **Hausaufgabe**, in das die folgenden auszuwertenden arithmetischen Ausdrücke eingetragen werden: ' (5-3)+(2*6) ', '5-(3*4+5)/(6-2)' und '3*(5/(3+2)-4)+6'. Für jeden arithmetischen Ausdruck, der in *h* gespeichert ist, wird seine Postfix-Notation mit Hilfe der Funktion aus **b)** ausgegeben.

THEMA Datenbanken **VARIANTE 1**

Aufgabe 1 (4 Punkte)

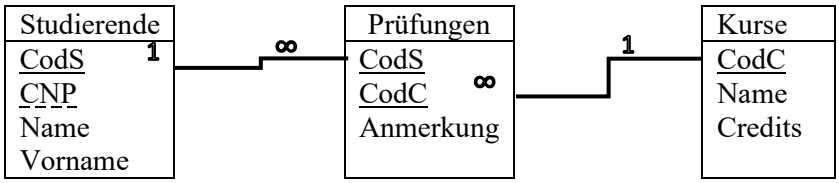
Eine Firma, die Handtücher herstellt, speichert Informationen über Kunden, Handtücher und Kundenrezensionen zu Handtüchern in einer relationalen Datenbank:

- Ein Kunde hat einen Code, einen Namen, eine E-Mail-Adresse (eindeutig für jeden Kunden) und ein Geburtsdatum.
- Ein Handtuch hat einen Code, einen Namen, eine Beschreibung, ein Muster, eine Länge und eine Breite.
- Ein Material hat einen Code, einen Namen und eine Beschreibung. Ein Material kann für mehr als ein Handtuch verwendet werden, und ein Handtuch kann aus mehr als einem Material hergestellt werden. Jedes Material kann mit jedem Handtuch höchstens einmal assoziiert werden. Für jedes Paar, das aus einem Handtuch und einem Material besteht (z. B. das Paar aus Handtuch p1 und Material m1), wird der prozentuale Anteil des Materials im Handtuch gespeichert.
- Ein Kunde kann mehreren Handtüchern Noten geben, und ein Handtuch kann Noten von mehreren Kunden erhalten. Jeder Kunde kann jedem Handtuch höchstens eine Note geben.

Realisieren Sie ein relationales BCNF-Schema für die Datenbank, wobei Sie Primärschlüssel, Kandidatenschlüssel und Fremdschlüssel genau hervorheben. 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 Studenten).



V2.
Studenten [CodS, CNP, Name, Vorname]
Kurse [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 ein Fremdschlüssel in Prüfungen und verweist auf {CodC} in Kurse.

Aufgabe 2 (5 Punkte)

Betrachten Sie die folgenden Beziehungen in einer Datenbank über eine Firma, die Brettspiele zur Miete anbietet:

Kategorien(CodeKategorie, Name)
BoardGames(CodeBoardGame, Name, *CodeKategorie*, Erscheinungsjahr, NrMinSpieler, NrMaxSpieler, PreisProTag)
Person(PersonCode, Name, TelefonNr)
Vermietungen(VermietungsCode, *FirmenSpielCode*, *PersonenCode*, VermietungsDatum, AnzahlTage, BetragBezahlt)

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

a. Schreiben Sie eine SQL-Abfrage, die den Code und den Namen der Brettspiele zurückgibt, die einzeln spielbar sind (siehe Attribut NrMinSpieler), im Jahr 2024 mindestens einmal ausgeliehen wurden und ein Erscheinungsjahr haben, das dem neuesten Jahr aller Brettspielveröffentlichungen entspricht, die die aufgeführten Bedingungen erfüllen (siehe Attribut Erscheinungsjahr).

b. Geben Sie die folgenden Instanzen der Beziehungen BoardGames, Kategorien, Personen und Vermietungen an:

Personen:

Code Person	Name	Telefon
1	P1	1111111111
2	P2	2222222222
3	P3	3333333333

Kategorien:

Code Kategorie	Bezeichnung
1	Strategiespiel
2	Familienspiel
3	Kooperationsspiel

Gesellschaftsspiele:

Gesellschaft SpielCode	Name	Code Kategorie	Erscheinungsjahr	NrMin Spieler	NrMax Spieler	Preis Protag
1	JS1	1	2020	3	6	15
2	JS2	2	2016	2	4	5
3	JS3	1	2020	2	5	12
4	JS4	3	2024	3	10	10

Vermietungen:

VermietungCode	GesellschaftsspielCode	Code Person	Datum Vermietung	Tage	Betrag Bezahlt
1	1	1	15.05.2025	3	45
2	2	1	18.05.2025	5	25
3	3	2	10.05.2025	2	24
4	3	2	02.06.2025	3	36
5	1	3	25.05.2025	1	15
6	2	3	03.06.2025	4	20
7	3	3	10.06.2025	2	24
8	4	2	20.06.2025	3	30

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

```
SELECT P.PersonCode, P.Name, MIN(BetragBezahlt) MinSum
FROM Personen P
    INNER JOIN Vermietungen I ON P.PersonCode = I.PersonCode
    INNER JOIN JS GesellschaftsSpiel ON I.GesellschaftsSpielCode = JS. GesellschaftsSpielCode
    INNER JOIN Kategorien C ON JS.CodeKategorie = C.CodeKategorie
WHERE C.Name = ' Strategiespiel '
GROUP BY P.CodePerson, P.Name
HAVING COUNT(DISTINCT I.GesellschaftsSpielCode) =
    (SELECT COUNT(*)
     FROM JS2 GesellschaftsSpiele
            INNER JOIN Kategorien C2 ON JS2.KategorienCode = C2.KategorienCode
     WHERE C2.Name = 'Strategiespiel'
     )
```

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

- {VermietungCode} → {GesellschaftsSpielCode}
- {CodePerson} → {DatumVermietung}.

BAREM INFORMATICĂ
VARIANTA 1

Subiect Algoritmică și Programare

Oficiu – **1p**

Cerința **1**. **– 3.5p**
- signatura – **0.2p**
- implementare având complexitate timp $\theta(n)$ – **3.2p**
  * soluție având complexitate timp $O(n^2)$ – **2**p
- returnare rezultat – **0.1p**

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

Cerința **3.a) – 2.25p**

Definirea clasei Token – **0.2p** din care
- clasă abstractă – 0.1
- metoda **toString** – 0.1

Definirea clasei BinaryOperator– **0.65p** din care
- relația de moștenire – 0.15
- atribut – 0.1
- **constructor (a1)** – 0.3
- metoda **toString** – 0.1

Definirea clasei Constant– **0.45p** din care
- relația de moștenire – 0.15
- atribut – 0.1
- constructor – 0.1
- metoda **toString** – 0.1

Definirea clasei ExpressionTree – **0.55p** din care
- atribute– 0.1
- constructor – 0.1
- metode **getToken, getLeft, getRight** – 0.15
- metoda **isLeaf (a2) –** 0.2

Definirea clasei Homework – **0.4p** din care
- atribut – 0.1
- metoda **getExpressions** – 0.1p
- metoda **addExpression (a3)** – 0.2p

Funcția **3.b)** – **1.75p**
- signatura – **0.1p**
- implementare parcurgere postordine – **1.55p**
- returnare rezultat – **0.1p**

Funcția principală **3.c)** – **0.5p**
- construire obiect **h** – 0.05p
- adăugare expresii aritmetice în **h** - 0.25p
- parcurgere listă expresii memorate în **h** și afișare formă postfixată – 0.2p din care
  apel funcție b) pentru expresie– 0.1p
  afișare formă postfixată expresie - 0.1 p

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:

CodPersoana	Nume	MinSum
3	P3	15

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

- **b2 -** {CodInchiriere} → {CodJocDeSocietate} este satisfăcută – **0.25p**; **0.25p** explicație
  - {CodPersoana} → {DataInchirierii} 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 1

SUBIECT Sisteme de operare

Barem:

1p – oficiu

Problema 1 (5p)

1p – a) Se rulează `fork` şi dacă eşuează condiţia va fi adevărată. Dacă `fork` nu eşuează, se creează un proces fiu şi condiţia va fi evaluată ca falsă şi de procesul părinte şi de procesul fiu.

1p – b) Câtă vreme `wait` returnează succes, se tipăreşte PID-ul procesului fiu care tocmai s-a încheiat.

1p – c) Se va afişa `abc` pentru că procesul iniţial va intra în primul if, dar nu va intra în `while` pentru că neavând procese fiu, `wait` va returna eroare.

1p – d) Nu se va afişa nimic, pentru că procesul iniţial şi primul proces fiu se încheie la linia 8, al doilea proces fiu şi procesul nepot se incheie la linia 6

1p – e) Afişează PID-urile celor două procese fiu pe care le creează la liniile 3 şi 5.

Problema 2 (4p)

1p – a) Scriptul Shell se relansează în execuţie, cu primul argument decrementat, al doilea argument directorul tocmai creat şi restul argumentelor identice cu cele rămase după shift.

1p – b) Va lua valorile `b`, `c` şi `d`

1p – c) Se va afişa: `x/a/a  x/a/b  x/b/a  x/b/b`

1p – d) Se va afişa cifra 8