BABEŞ-BOLYAI UNIVERSITY
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Bachelor Degree Written Exam 2026
Computer Science – English

VARIANT 2

REMARKS

- All subjects are compulsory and full solutions are requested.
- The minimum passing grade is 5.00.
- The working time is 3 hours.

SUBJECT Algorithms and Programming

- **The programming language that is used must be indicated.**
- **The implementations are not required to deallocate dynamically allocated memory areas.**
- **Lack of appropriate programming style (suggestive variable names, indentation of code, comments, if necessary, readability of code) will result in a 10% deduction from the subject's score.**
- **Do not add additional attributes or methods, other than those mentioned in the statement, except for constructors and destructors, if applicable. Do not change the visibility of attributes specified in the statement.**
- **Do not use sorted containers, predefined sort and search operations.**
- **Existing libraries (from C++, Java, C#, Python) may be used for data types.**

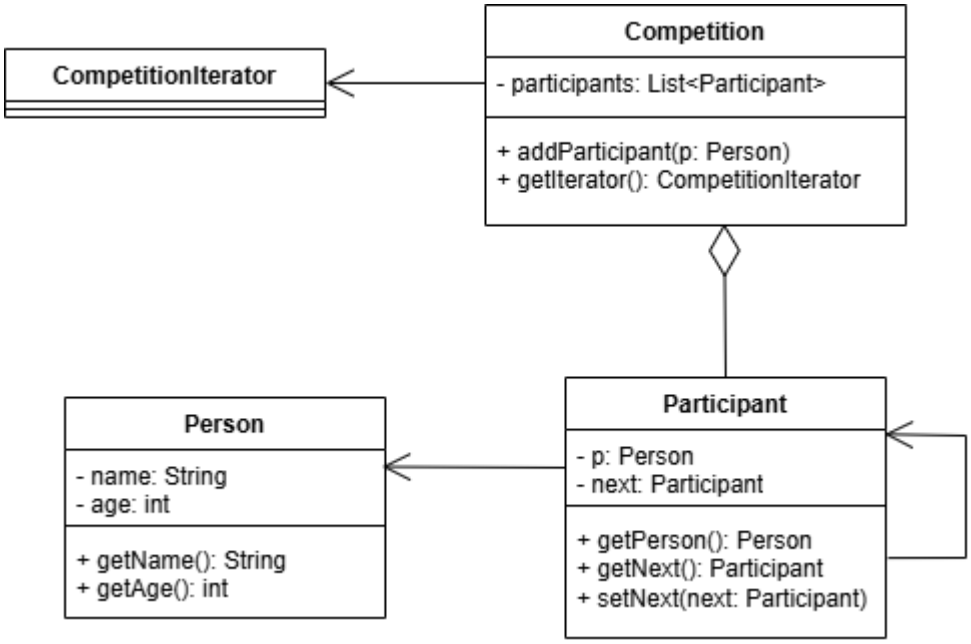
Participants from 7 age categories are registered for a competition: c_0 (ages 5-9), c_1 (ages 10-19), c_2 (ages 20-29), c_3 (ages 30-39), c_4 (ages 40-49), c_5 (ages 50-59) and c_6 (ages 60-69). Participants will be stored by age category, in the order c_0, c_1, …, c_6; within in each age category participants will be stored in alphabetical order.

1. **(2.5 points)** Write a function in one of the programming languages **Python, C++, Java** or **C#**, which receives as parameters a list *lst* of 7 lists of alphabetically ordered strings (the names of the participants in age group c_i are stored in alphabetical order at position i in the list *lst*), a natural number *age* and a string *name* (representing the age and name of a participant) and inserts the participant's name into the list corresponding to the age category, ensuring that this list remains alphabetically ordered. ***Do not use predefined library functions to insert an element at a specific position within the list.***

2. **(1 point)** Indicate the *best-case*, *average-case*, and *worst-case* time complexity for the function from item **1**. Justify your answer.

3. **(5.5 points)** Consider the following UML diagram, which contains the classes **Competition**, **Participant**, **Person**, **CompetitionIterator**.

Note ***The class constructors are not shown in the diagram.***



- The class **Competition** has an attribute *participants*, which is a list having as size the number of age categories. The participants in age category c_i are stored at position i in the *participants* list. The class has the following methods:
  - **addParticipant(p: Person)**, which adds person **p** to the list associated with their age group.
  - **getIterator()**, which returns an iterator for iterating through the competition participants.
- The class **Participant** represents a person in a list of persons participating in the competition in a specific age category. Participants in each age category are stored in alphabetical order by name.
- The class **Person** stores the *name* and *age* of a person participating in the competition. The class has methods for returning its attributes. A person's age is a natural number in the range [5–69].
- The class **CompetitionIterator** defines an iterator of persons participating in the competition, sorted by their age categories (c_0, c_1, …, c_6), and within each age category, sorted alphabetically by name.

Write a program that implements the following requirements, using one of the **C++, Java**, or **C#** programming languages:

a) Declare the classes **Competition**, **Person**, **Participant**, their attributes and methods as per the diagram above. Implement only the following methods:
  a1) Constructor of the **Competition** class.
  a2) Method **setNext**(**p**: **Participant**) in class **Participant**.

b) Define the class **CompetitionIterator**, including the attributes and methods specific to an iterator. Implement all the methods of the **CompetitionIterator** class. The attributes and implemented methods must use $\theta(1)$ memory space. The methods must have time complexity $\theta(1)$.

c) Define a function that receives as parameter an object **c** of type **Competition** and, using the class defined at **b)**, prints the competition participants, in the order of their age categories and within each age category, in alphabetical order, by name.

d) Define a function that receives as parameters an object **c** of type **Competition** and a string *name* and returns the age category in which the person having the name equal to *name* is competing, or an empty string, if there is no participant having the name equal to *name*. If there are multiple participants with the same name *name*, the first age group in which a participant with the given name is found will be returned.

e) Create an object **o** of type **Competition**; add in this object 2 age categories and 4 participants (2 in each age category). Call the function from **c)**. For one of the participants call the function from **d)**.

SUBJECT Databases **VARIANT 2**

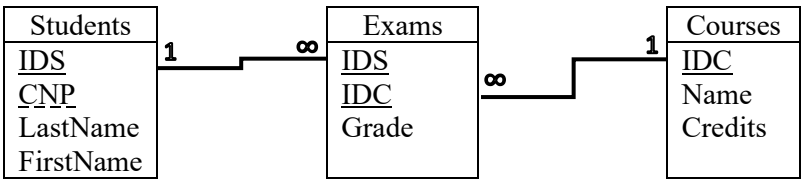
Problem 1. (4 points)

A media company uses a relational database to manage its activity.

- Each newspaper has an ID, a name (unique for each newspaper), a founding year, and a publishing frequency (it can be a simple character string attribute; e.g., *daily*, *weekly*, *monthly*); each newspaper is managed by a single central office, but one central office can manage multiple newspapers.
- A central office has an ID and a phone number (unique for each central office).
- Each journalist has an ID, last name, first name, date of birth, gender, and debut year. A journalist can have collaboration contracts with multiple newspapers. A newspaper can collaborate with multiple journalists. A journalist can have only one collaboration contract with a certain newspaper. For each collaboration contract, the following are stored: the newspaper, the journalist, the contract start date, the contract duration (an integer number representing the number of months), and the remuneration.
- An article has an ID, a title, and is written by a journalist for one of the newspapers with which the journalist collaborates.
- A topic has an ID, a name (unique for each topic; e.g., *Economy*, *Sport*), and a description. An article can cover multiple topics, and a topic can appear in multiple articles.

Create a BCNF relational schema for the database, rigorously highlighting the primary keys, candidate keys, and foreign keys. Create the schema in one of the ways indicated in the example below:
* example for tables Students, Courses, and Exams:

V1. Diagram with tables, primary keys underlined with a solid line, candidate keys underlined with a dashed line, and relationships drawn directly between foreign keys and the corresponding primary / candidate keys (for instance, the relationship drawn between column IDS in Exams and column IDS in Students).



V2.
Students[IDS, CNP, LastName, FirstName]
Courses[IDC, Name, Credits]
Exams[IDS, IDC, Grade]
Primary keys are underlined with a solid line, and candidate keys are underlined with a dashed line.
{IDS} in Exams is a foreign key that references {IDS} in Students. {IDC} in Exams is a foreign key that references {IDC} in Courses.

Problem 2. (5 points)

Consider the following relational schemas:
ContentCreators[CreatorID, Name, Country, BirthYear]
Videos[VideoID, Title, Category, Duration, *CreatorID*]
Platforms[PlatformID, Name]
VideosPlatforms[*VideoID, PlatformID*, Monetization, PublishingYear]

Primary keys are underlined. Foreign keys are written in italics and have the same names as the columns they reference.

a. Write an SQL query that returns the ID and name of each content creator who satisfies the following two conditions:
1. he / she published at least 2 videos in the same year and
2. he / she has at least one video published on the "StreamNow" platform.
A video is considered *published* if it appears on a platform.

Example for condition 1:
- creator C1 published videos: V1 in 2020, V2 in 2020, V3 in 2021; C1 satisfies condition 1.
- creator C2 published videos: V4 in 2020, V5 in 2021, V6 in 2022; C2 does not satisfy condition 1.

b. Consider the following instances of relations ContentCreators, Videos, Platforms, and VideosPlatforms:

ContentCreators

Creator ID	Name	Country	BirthYear
1	n1	Romania	2005
2	n2	Romania	2005
3	n3	France	2005

Platforms

Platform ID	Name
1	p1
2	p2
3	p3

Videos

Video ID	Title	Category	Duration	Creator ID
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

VideosPlatforms

Video ID	Platform ID	Monetization	Publishing Year
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. Write the result of evaluating the query below on the given instances. Provide only the values of the tuple(s) and the names of the columns in the result, without describing all the steps involved in evaluating the query.

```
SELECT vp.PlatformID, MIN(vp.PublishingYear) MinYear
FROM ContentCreators c INNER JOIN Videos v ON c.CreatorID = v.CreatorID
        INNER JOIN VideosPlatforms vp ON v.VideoID = vp.VideoID
WHERE c.Country = 'Romania'
GROUP BY vp.PlatformID
HAVING COUNT(vp.VideoID) >
        (SELECT COUNT(*)
        FROM VideosPlatforms vp2 INNER JOIN Platforms p2
           ON vp2.PlatformID = p2.PlatformID
        WHERE p2.Name = 'p2')
```

b2. Explain whether the following functional dependencies are satisfied or not by the data in the Videos instance:

- $\{Category\} \rightarrow \{Title\}$
- $\{CreatorID\} \rightarrow \{VideoID\}$

SUBJECT Operating Systems

Problema 1. (5 points) Answer the following questions about the execution of the program `./a.out`, compiled from the source code below, assuming that all necessary includes are present and that the `fork` and `write` system calls are executed successfully. Take note, that the `WEXITSTATUS(st)` macro extracts the exit code returned by the child process through the `exit` call (an 8-bit integer, between 0–255).

```
 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) What will the execution `./a.out ab cde f` display? Justify your answer.

b) How many child processes are created by `./a.out ab cde f` ? How about `./a.out` (without arguments) and what is displayed in this case? Justify your answer.

c) Explain in detail the functioning of line `6` and how is the value sent by `exit` reaching the parent process?

d) Explain in detail the functioning of lines `8-9`.

e) What will display `./a.out` if called with an argument consisting of 260 characters? Justify your answer.

Problem 2. (4 points) Answer the following questions about the execution of the `./a.sh` UNIX Shell script below, assuming that it uses the given `d.txt` file. Note: the line numbers are NOT part of the file content.

```
 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) Specify what does the script display for line `1` respectively line `2` of file `d.txt`. Justify your answer.
b) Explain the functioning of command `sed` on line 8.
c) Specify the numerical values that will be displayed. Justify your answer.
d) Explain in detail the functioning of lines `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