BABEŞ-BOLYAI UNIVERSITY
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Bachelor Degree Written Exam 2026
Computer Science – English

VARIANT 1

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

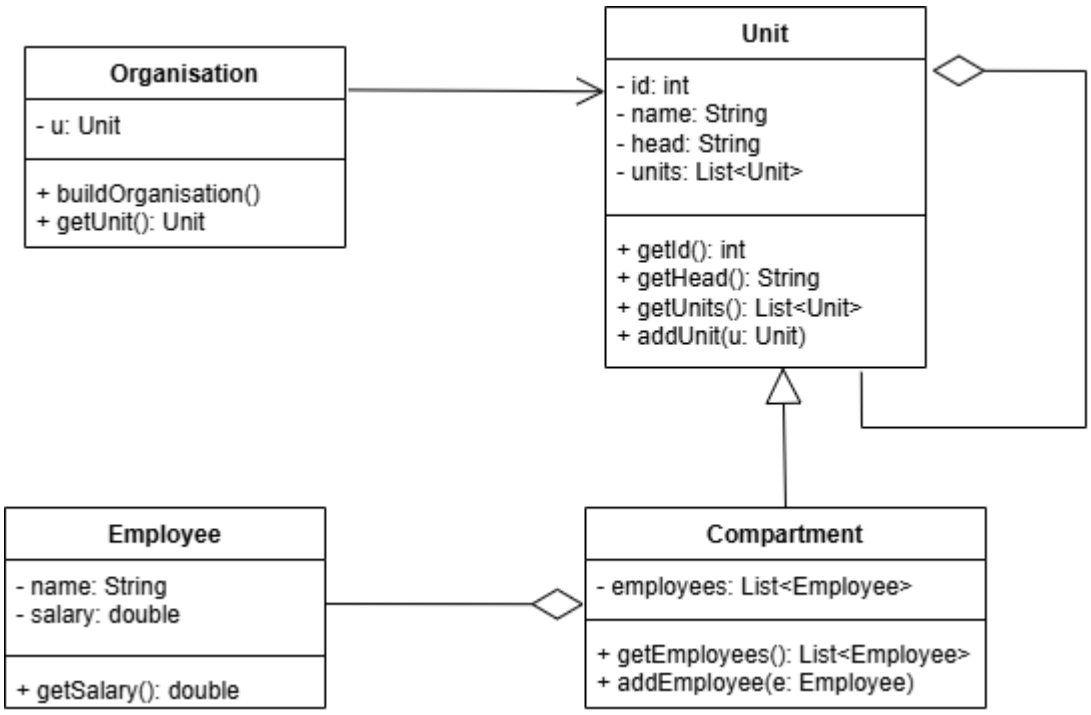
- **The programming language that is used must be indicated.**
- **The implementations are not required to deallocate dynamically allocated memory areas.**
- **Lack of appropriate programming style (suggestive variable names, indentation of code, comments, if necessary, readability of code) will result 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 or search operations.**
- **Existing libraries (from C++, Java, C#, Python) may be used for data types.**

1. **(3 points)** Write a function in one of the programming languages **Python, C++, Java** or **C#**, which receives as parameters an array *v* containing real numbers sorted in descending order (representing the salaries of employees in a compartment) and a real number *salary* (representing the salary of a new employee). The function will insert *salary* into the array *v* so that the array remains sorted in descending order of salaries. Use an auxiliary function, with an iterative implementation, which returns, in $O(\log_2 n)$ time complexity, the position at which the *salary* should be inserted into the array *v*, where *n* is the length of the array *v*. *Recursive solutions will be partially scored.*
   Note. Inserting an element at position *k* into the array *v* involves shifting the elements at positions *k, k+1, ..., n* one position to the right, after which the length of the array increases by 1. ***Do not use predefined insertion functions from libraries to insert an element at a specific position within the array.***

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 points)** Consider the following UML diagram, which contains the classes **Organisation** (represents the hierarchical structure of an organization), **Unit** (Organisational unit), **Compartment**, **Employee**.

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



- The class **Organisation** represents an organization (represented as a hierarchical structure of organizational units). The **getUnit()** method returns attribute *u* of the class.
  - The method **buildOrganisation**() creates the hierarchical structure of the organisation. *Implementation of this method is not required.*
- The class **Unit** represents the hierarchical structure of an organizational unit. It has an identifier (*id*), a name (*name*), and a director (whose name is stored in the attribute *head*). The unit has several (sub)units under its direct authority (stored in the attribute *units*). The class has methods for accessing *id*, *head* and *units* and a method **addUnit(*u*: Unit)** that adds unit *u* to the end of the *units* list.
- The class **Compartment** represents a unit that has no other (sub)units under its direct authority. It stores a list of employees (**Employee**) in the *employees* attribute. The list *employees* stores the employees in descending order of their salaries. The class has a method **addEmployee**(*e*: **Employee**) that adds an employee to the sorted *employees* list and a method **getEmployees()** that returns the list of employees in the compartment.
- The class **Employee** represents an employee identified by a *name* and a *salary*. The class has a method **getSalary()** that returns the *salary* attribute.

Note: *The units (***Unit***) within the organization (***Organisation***) have distinct identifiers (id).*

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

a) Declare all classes, attributes and methods as per the diagram above. Implement only the constructor of the class **Compartment**.

b) Define a function that receives as parameter an object *o* of type **Organisation** and an integer number *n* and returns a list containing all employees under the authority (direct or indirect) of the organisation's unit having the *id* equal to *n*. The returned list will be empty if the organization does not have a unit with an *id* equal to *n*.

c) Define a function that receives as parameter an object *o* of type **Organisation** and returns a list of compartment heads within the organisation *o*.

d) Create an object *o* of type **Organisation**, call the method **buildOrganisation()**,then call the functions from **b)** and **c)** for the created organisation.

SUBJECT Databases **VARIANT 1**

Problem 1. (4 points)

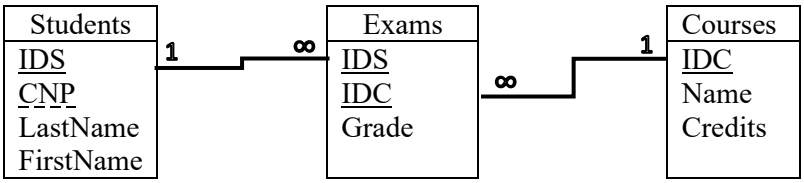
A research institute stores information about herbaria used in its research activities. The herbaria are stored in different storage spaces within the institute's building and are used by the institute's employees. To manage information about employees, storage spaces, herbaria, plants, and geographical regions, a relational database is used:

- An employee of the research institute has an id, a name, an email address (unique for each employee), date of birth, the position held within the research institute, and salary.
- A storage space has an id, a name, a description, and an employee responsible for its administration. Each employee can be responsible for administering multiple storage spaces, but a storage space can be administered by only one employee.
- A geographical region has an id, a name, a description, and a country.
- An herbarium has an id, a title, a description, a date when it was taken into the institute's management and is stored in a specific storage space within the institute. Each storage space can store multiple herbaria, but an herbarium is stored in a single storage space.
- A plant has an id, a scientific name, a common name, class, order, and a value indicating whether it is protected or not. An herbarium can contain multiple plants, and a plant can appear in multiple herbaria. Each plant can be associated with each herbarium only once. For each pair consisting of a plant and an herbarium (for example, the pair composed of plant p1 and herbarium h1), the date when the plant was added to the herbarium will be stored, together with the geographical region from which the plant was collected.

Construct a BCNF relational schema for the database, rigorously highlighting primary keys, candidate keys, and foreign keys. Construct 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, 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 referencing {IDS} in Students. {IDC} in Exams is a foreign key referencing {IDC} in Courses.

Problem 2. (5 points)

We consider the following relations from a database that stores information about artworks and temporary exhibitions of a museum:

Exhibitions[ExhibitionId, Title, Description, OpeningDate, RoomNumber]
ArtisticPeriods[PeriodId, PeriodName]
Artists[ArtistId, Name, Birthdate, *PeriodId*]
Artworks[ArtworkId, Title, CreationYear, *ArtistId, ExhibitionId*, ArtworkType]
Primary keys are underlined. Foreign keys are written in italics and have the same names as the referenced columns.

a. Write an SQL query that returns all exhibitions (exhibition title, opening date, room number) that contain at least one artwork created by the artist with the name "Pablo Picasso", but do not include any artwork created by the artist named "Claude Monet". Remove duplicates.

b. The following instances of the relations ArtisticPeriods, Artists, and Artworks are given:

ArtisticPeriods:

PeriodId	PeriodName
1	Cubism
2	Impressionism
3	Postimpressionism

Artists:

ArtistId	Name	Birthdate	PeriodId
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

Artworks:

Artwork Id	Title	Creation Year	Artist Id	Exhibition Id	Artwork Type
1	Guernica	1937	1	1	Painting
2	Jacqueline with Flowers	1954	1	1	Painting
3	Water lilies	1919	2	2	Painting
4	The Little Fourteen-Year-Old Dancer	1880	4	2	Sculpture
5	The Ballet Class	1874	4	2	Painting
6	Sunflowers	1888	3	3	Painting
7	The Starry Night	1889	3	2	Painting
8	Cafe Terrace at Night	1888	3	1	Painting

b1. Specify the result of evaluating the query below on the given instances. Mention strictly the values of the tuple(s) and the column names in the result, without presenting all steps of the query evaluation.

```
SELECT A.Name, PA.PeriodName, COUNT(O.ArtworkId) as Nr
FROM Artists A
INNER JOIN ArtisticPeriods PA ON A.PeriodId = PA.PeriodId
INNER JOIN Artworks O ON O.ArtistId = A.ArtistId
GROUP BY A.ArtistId, A.Name, PA.PeriodName
HAVING COUNT(O.ArtworkId) >=
  (SELECT AVG(T.Nr)
   FROM
     (SELECT A1.ArtistId, COUNT(*) AS Nr
      FROM Artists A1
      INNER JOIN ArtisticPeriods PA1 ON A1.PeriodId = PA1.PeriodId
      INNER JOIN Artworks O1 ON O1.ArtistId = A1.ArtistId
      GROUP BY A1.ArtistId
     ) T
  )
```

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

- { ArtworkId } → { ArtistId }
- { ExhibitionId } → { ArtworkId }.

SUBJECT Operating Systems

Problem 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.

```
 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) How many times is the line `END` displayed by the execution `./a.out`? Justify your answer.

b) How many times will the `L0`, `L1`, and `L2` lines appear in file `out`, after the execution `./a.out > out`? Justify your answer.

c) How many processes are created in total during the execution (except for the initial process). Justify your answer.

d) Explain in detail the functioning of the loop on lines `10-11` and specify how many direct child prcesses are waited for by the initial process. Justify your answer.

e) How many times are the `L` lines displayed `(L0+L1+L2)`, during the `./a.out` execution, if line `6` is moved immediately after line `7`? 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 `f.txt` file. Note: the line numbers are NOT part of the file content.

```
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) What does display the `while` loop (lines `2-6`, before the first `| sort`)? Justify your answer.
b) What is the effect of the expression `sort | uniq -c`? Justify your answer.
c) What does the script display overall? Justify your answer.
d) How is the result changed if the `sort` command before `uniq -c` is removed? Justify your answer.

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