BABEŞ-BOLYAI UNIVERSITY
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Bachelor Degree Written Exam 2026
Artificial Intelligence in 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.**

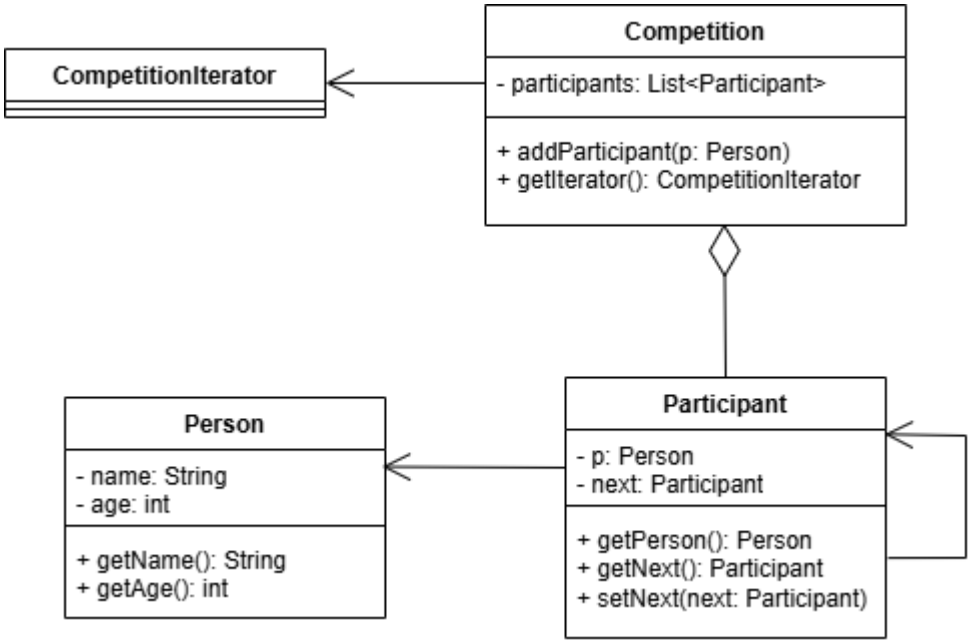
Participants from 7 age categories are registered for a competition: c_0 (ages 5-9), c_1 (ages 10-19), c_2 (ages 20-29), c_3 (ages 30-39), c_4 (ages 40-49), c_5 (ages 50-59) and c_6 (ages 60-69). Participants will be stored by age category, in the order c_0, c_1, …, c_6; within in each age category participants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

VARIANT 2

SUBJECT Metaheuristics (3 points)

A factory has one machine and must process **n** jobs. Each job takes a certain processing time (in hours) to complete. Given that the machine can only do one job at a time, find the order of the jobs that minimizes the total completion time. For each of the following requirements, provide a clear solution and explain it.

a) Identify a permutation-based representation for a solution to the problem. Define and justify the size of the resulting search space.

b) Define a neighbourhood function for the selected representation and specify the size of the neighbourhood. Given a permutation solution (1 2 3) for n=3 jobs, specify the list of neighbours determined by the selected neighbourhood function.

c) Define a crossover operator that can be used for this problem under the selected representation in an evolutionary algorithm. Give an example of its application for n=5 jobs and two parents randomly selected.

VARIANT 2

SUBJECT Machine Learning (3 points)

The Cookie Quality Inspector

A bakery is testing two automated systems to help classify cookies before packaging. The goal is simple:

- *yummy* = perfect cookie (well-baked, great texture)
- *yucky* = imperfect cookie (burnt, undercooked, or misshapen)

A human expert tastes and evaluates a small batch of cookies and records the ground truth for 8 cookies:

[*yummy*, *yucky*, *yummy, yummy, yucky, yummy, yummy, yucky*].

A virtual master chef M1 makes final decisions for each cookie:

[*yummy, yummy, yummy, yucky, yucky, yummy, yummy, yucky*].

A more careful virtual master chef M2 assigns a probability that a cookie is perfect:

[0.92, 0.69, 0.45, 0.72, 0.65, 0.62, 0.93, 0.20].

For example: a cookie with 0.92 is very likely perfect, a cookie with 0.20 is likely imperfect. The bakery wants to evaluate how well M1 performs in correctly identifying perfect cookies and whether M2 might actually be better, even though it doesn't directly give final label

Tasks:

a) For M1, construct the confusion matrix and compute accuracy, precision and recall for the yummy class.

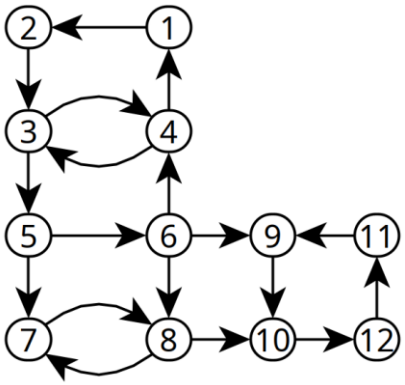
b) Describe the setup when the model M2 can be considered more performant than the model M1.

VARIANT 2

SUBJECT Graphs

Problem 1 (1 point)

Consider the following graph and answer the requirement:



Is this graph bipartite? Justify your answer.

Problem 2 (2 points)

We have a set of comic books. All the comic books have a separate story but some of them share a character, meaning they happen in the same universe. We know which comic book shares a character with what other comic books. We want to group together all comic books that appear in the same universe (e.g. if comic book A has a character in common with comic book B, and comic book B has a different character in common with comic book C, that means that A, B and C all happen in the same universe and should be grouped together; on the other hand if neither of them share a character with comic book X, then X happens in a separate universe and should be in a different group).

For each of the following requirements, provide a clear solution and explain it.

a) Describe how you would model this problem as a graph and what type of graph you would use.

b) How do you find the groups using such a graph? Describe an algorithm that solves the problem.

c) Give an example with at least 6 comic books and 3 different universes. Draw the graph associated with your example. Show what would happen if we applied on this graph the algorithm that you described at b).

BAREM INTELIGENȚĂ ARTIFICIALĂ
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 INTELIGENȚĂ ARTIFICIALĂ
VARIANTA 2

Subiect Inteligență Artificială

Oficiu – 1p

Subiect Metaeuristici – 3p

Cerința a) – 1 p

Definire reprezentare prin permutări – 0.5 p

Definire şi justificare mărime spaţiu de căutare – 0.5 p

Cerința b) – 1 p

Definire funcţie de vecinătate – 0.5 p

Specificare corectă a mărimii vecinătăţii – 0.25 p

Specificare corectă a listei de vecini pentru exemplul cerut – 0.25 p

Cerința c) – 1 p

Definire operator încrucişare – 0.75 p

Exemplu de aplicare a operatorului de încrucişare – 0.25 p

Subiect Machine Learning – 3p

Cerința a) – 1.75p

matricea de confuzie [[4, 1], [1, 2]] – 0.25p

Acc = (TP + TN) / N = 6 / 8 = 0.75 – 0.5p

Precision = TP / (TP + FP) = 4 / 5 = 0.8 – 0.5p

Recall = TP / (TP + FN) = 4 / 5 = 0.8 – 0.5p

Cerința b) – 1.25p

discutie pe baza

- threshold-ului – 0.75p
- metrica luata in calcul (F1) – 0.5p

Subiect Grafe – 3p

Problema 1 – 1p

Explică ce înseamnă graf bipartit – 0.5p

Justifică corect de ce acest graf e bipartit (e.g. dă partițiile corecte) – 0.5p

Problema 2 – 2p

Cerința a) Justifică corect că trebuie folosit graf neorientat care nu e conectat – 0.5p

Cerința b) – 0.75p

Determină că fiecare grup e o componentă conexă – 0.25p

Descrie un algoritm corect de găsire a componentelor conexe – 0.5p

Cerința c)

Dă un exemplu și îl desenează corect – 0.25p

Exemplifică corect cum ar funcționa algoritmul de la b) – 0.5p