BABEȘ-BOLYAI UNIVERSITY
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Bachelor Degree Written Exam, July 3, 2023
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

Remarks

- Lack of a proper programming style (suggestive names for variables, code indentation, comments if necessary, code readability) leads to losing 10% of the subject score.
- In Pseudocode the following statements can be used exclusively: *assign*, *for*, *if*, *while*, *repeat*, *display*, and *return*.
- Do not add any additional attributes or methods, other than those mentioned in the statement, except for constructors and destructors, if necessary. Do not change the visibility of the attributes specified in the statement.

Problem 1. (1.5 points)

Let us consider the Pseudocode subalgorithms defined below. The subalgorithm **transformare** has as input data the array x with n integer elements ($x[1]$, …, $x[n]$) and the array y with m integer elements ($y[1]$, …, $y[m]$) and as output data the array z with k integer elements ($z[1]$, …, $z[k]$). The subalgorithm **g** has as input data the array x with n integer elements and a natural number i ($1 \leq i \leq n$) and as output data the array y with n elements. What will the array z contain after the call **transformare**(x, n, y, 0, z, k)? Justify your answer.

```
Subalgorithm g(x, n, y, i)
  If i ≤ n then
    y[i] ← x[i]
    g(x, n, y, i + 1)
  EndIf
EndSubalgorithm

Subalgorithm transformare(x, n, y, m, z, k)
  If n = 0 then
      g(y, m, z, 1)
      k ← m
   else
      y[m +1] ← x[n]
      transformare(x, n - 1, y, m + 1, z, k)
  EndIf
EndSubalgorithm
```

Problem 2. (3 points)

Given an array $a[1]$, …, $a[n]$ ($4 \leq n \leq 10000$) having integer elements in the interval $[-10^{15}, 10^{15}]$, write a program in the C++ programming language, having the worst-case time complexity $O(n^3)$, which displays on the screen four natural numbers x, y, z, t ($x, y, z, t \in [1, n]$, $x < y < z < t$), so that $a[x] + a[y] + a[z] + a[t] = 0$. If there are no four numbers on distinct positions in the array having the sum 0, the program will display the value -1 on the screen. **Remark**. *Solutions that do not fall within the indicated complexity class will receive a partial score. Algorithms and containers from the STL can be used.*

Problem 3. (2.25 points)

Implement in C++ the classes **MyObject, MyInteger, MyString, MyObjectList** and **MyListIterator** so that the following C++ function's output is the one mentioned in the comment and the memory is correctly managed.

```
void function()
{
	MyObjectList list{};
	list.add(new MyInteger{ 2 }).add(new MyString{ "Hi" });
	MyString* s = new MyString{ "Bye" };
	list.add(s).add(new MyString{ "5" });

	MyListIterator i{ list };
	while (i.isValid()) {
		MyObject* o = i.element();
		o->print();
		i.next();
	} // prints: 2 Hi Bye 5
}
```

Problem 4. (2.25 points)

Implement in C++ the class **Person,** which has the private attributes: *surname* (string), *firstname* (string), *age* (integer), a constructor that initializes all attributes, and accessor methods for all attributes of the class. Write a function that sorts a list of persons. The function will receive as input parameters the list of persons and a function representing the sort criteria. The time complexity of the sorting function, in the worst case, must be $\Theta(n \log_2 n)$, where n is the number of elements in the list.

Call the function to sort a list of objects of type **Person** in ascending order:

- by the attribute *name*
- by the attribute *age*
- by *age* and *name* (if they have the same *age*, they are compared by *name*)

Observation. Do not use any predefined sort operations.

SUBJECT Databases

Problem 1. (4 points)

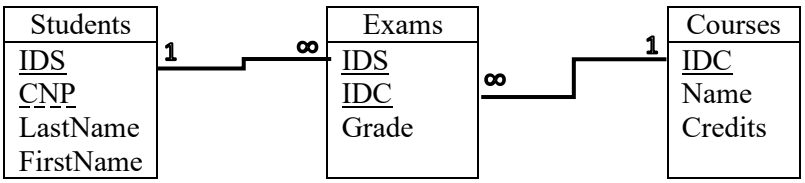
The lineup for a figure skating championship includes competitors who can register for the championship events. Competitors participating in events are evaluated by judges. The data is stored in a relational database.

- A competitor has an ID, last name, first name, and date of birth.
- An event has an ID, name, description, and date.
- A competitor can register for multiple events. Multiple competitors can register for an event.
- A judge has an ID, last name, and first name.
- An evaluation is made by a judge for a competitor who is participating in a certain event and involves giving an integer grade from 1 to 10 to the competitor in the event. A judge can evaluate multiple competitors who participate in the same event, as well as multiple events for a certain competitor. A judge can evaluate at most once a certain competitor's participation in an event. A competitor can be evaluated in an event only if they have registered for that event.

Create a relational BCNF 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, 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)

Consider the following relational schemas:

Newspapers[NewspaperID, NewspaperName]
Journalists[JournalistID, LastName, FirstName, *NewspaperID*]
Categories[CategoryID, Name]
Articles[ArticleID, Title, PublicationDate, Text, *JournalistID*, *CategoryID*]
Primary keys are underlined. Foreign keys are written in italics and have the same name as the columns they reference.

a. Write an SQL query that returns the total number of articles in the category named *Economics* published by journalists working at the newspaper named *Universe*.

b. Consider the following instances for relations Categories and Articles:

Categories

CategoryID	Name
1	c1
2	c2
3	c3

Articles

ArticleID	Title	PublicationDate	Text	JournalistID	CategoryID
1	a1	1-1-2022	t1	1	2
2	a2	1-2-2022	t2	2	3
3	a3	1-1-2022	t3	3	1
4	a4	1-5-2022	t4	1	2
5	a5	1-3-2022	t5	2	1
7	a7	1-7-2022	t7	2	2
8	a8	1-5-2022	t8	4	2

b1. Write the result of evaluating the query below on the given instances. Give only the values of the tuple(s) and the names of the columns in the result without describing all the steps of evaluating the query.

```
SELECT A.JournalistID, COUNT(*) No
FROM Articles A INNER JOIN Categories C ON A.CategoryID = C.CategoryID
WHERE C.Name = 'c2'
GROUP BY A.JournalistID
HAVING 2 < (SELECT COUNT(*)
      FROM Articles A2
      WHERE A2.JournalistID = A.JournalistID)
```

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

- $\{JournalistID\} \rightarrow \{Title\}$
- $\{ArticleID\} \rightarrow \{Text\}$

SUBJECT Operating Systems

Problem 1 (5p)
Answer the following questions about the execution of the code fragment below, considering that the pipes and the process are created successfully.

```
 1  int main(int argc, char** argv) {
 2      int a[2], b[2], c[2], i;
 3      char val=0, x=0;
 4
 5      pipe(a); pipe(b); pipe(c);
 6      if(fork() == 0) {
 7          close(b[0]); close(c[1]);
 8          for(i=0; i<3; i++) {
 9              write(a[1], &val, 1);
10              write(b[1], &x, 1);
11              read(c[0], &x, 1);
12              read(a[0], &val, 1);
13          }
14          printf("%d\n", val);
15          close(a[0]);close(a[1]);
16          close(b[1]);close(c[0]);
17          exit(0);
18      }
19      close(b[1]); close(c[0]);
20      for(i=0; i<3; i++) {
21          read(b[0], &x, 1);
22          read(a[0], &val, 1);
23          val++;
24          write(a[1], &val, 1);
25          write(c[1], &x, 1);
26      }
27      close(a[0]);close(a[1]);
28      close(b[0]);close(c[1]);
29      printf("%d\n", x);
30
31      wait(NULL);
32      return 0;
33  }
```

a) What is the role of pipe `c`?
b) What will be displayed in the console?
c) What will happen if lines 11 and 12 are swapped?
d) How is the process execution affected if line 7 is moved between lines 14-15 and line 19 is moved between lines 26-27?
e) What happens if line 17 is eliminated?

Problem 2 (4p)
Answer the following questions about the execution of the UNIX Shell script below.

```
1  #!/bin/bash
2
3  for F in `find /usr/sbin -type f`; do
4    if [ ! -x "$F" ] && ls -l "$F" | grep -q -E "^-(.{2}x|.{5}x|.{8}x)"
5    then
6      echo "$F:" `ps -e -f | grep -E "$F" | grep -E -v "grep" | wc -l`
7    fi
8  done
```

a) What will be the values of variable `F`?
b) Explain the regular expression on line 4.
c) What files will satisfy the first part of the condition on line 4 (before `&&`)?
d) What files will satisfy the second part of the condition on line 4 (after `&&`)?
e) What is the role of the second `grep` on line 6?

BAREM INFORMATICĂ
VARIANTA 1

Subiect Algoritmică şi Programare

Oficiu – 1p

Problema 1. Punctaj - 1.5p

- Răspuns corect (șirul *z* va conține elementele șirului *x* în ordine inversă): **0.75 p**
- Justificare: **0.75p**

Problema 2. Punctaj - 3p

- Soluție având complexitatea timp în caz defavorabil $O(n^3)$ - 3p
- Soluție având complexitatea timp în caz mediu $\theta(n^3)$ - 1.75p
- Soluție având complexitatea timp $O(n^3 \log_2 n)$ - 1.25p
- Soluție având complexitatea timp $O(n^4)$ - 0.5p

Problema 3. Punctaj - 2.25p

- Clasa **MyObject** (**0.2p**): abstractă, funcția abstractă **print,** virtual destructor
- Clasa **MyInteger** (**0.3p):** mostenire, camp privat de tip *int* + constructor + funcția **print**
- Clasa **MyString (0.3p)**: mostenire, camp privat de tip *string* + constructor, funcția **print**
- Clasa **MyObjectList (0.85p)**:
  - vector de elemente *Object* (pointers) (0.2), funcția **add** (0.2), funcția **length** (0.1), destructor (0.15), funcție care accesează elementul de pe o poziția dată *sau* funcție **getObjects** *sau* friend class MyListIterator (0.2).
- Clasa My**ListIterator (0.6p)**:
  - câmpuri private de tip *MyObjectList* și *int* (elementul curent) + constructor (0.25), funcția **isValid** (0.1), funcția **next** (0.1), funcția **element** (0.15).

Problema 4. Punctaj - 2.25p

- Clasa Persoana - **0.3p**
- Signatura funcției: lista/array + parametru de tip funcție (de comparare sau funcție care returnează cheia după care sortăm) - **0.3p**
- Implementare **Merge Sort** (partea recursivă/interclasare) - 0.6+0.75 = **1.35p**
- Implementare funcții de comparare/cheie (sau funcții lambda) și apel corect pentru cele 3 variante de sortare- **0.3p**

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:

CodJurnalist	Nr
2	1

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

- **b2** - {CodJurnalist}→{Titlu} nu este satisfăcută – **0.25p**; **0.25p** explicație
  - {CodArticol}→{Text} este satisfăcută – **0.25p**; **0.25p** explicație

Notă: La specializările Informatică engleză și Informatică maghiară se iau în considerare versiunile traduse în limbile corespunzătoare.

Subiect Sisteme de operare

Oficiu – 1p

Problema 1. Punctaj – 5p

a) Previne procesul fiu de la a citi din `a` ceea ce a scris el însuşi – 1p
b) Se va afişa 0 şi 3 în orice ordine – 1p
c) Dacă procesul fiu citeşte din `a` ce a scris chiar el, se blochează, altfel functionează ca la a) – 1p
d) Funcţionarea nu este afectată – 1p
e) Procesul fiu îşi continua execuţia cu instricţiunile de după IF – 1p

Problema 2. Punctaj – 4p

a) Numele fişierelor normale din directorul /usr/bin găsite recursiv şi în subdirectoare – 0.5p
b) La început de linie, minus urmat de orice grup de caractere de lungime 2, 5 sau 8 urmat de x – 1p
c) Fişierele care nu pot fi executate de utilizatorul care rulează scriptul – 0.5p
d) Fişierele care au drept de execuţie pentru owner, group sau others – 1p
e) Elimină din lista de procese cele care conţin în linia de comandă `$F` pe cele care conţin cuvântul `grep` – 1p