BABEŞ-BOLYAI UNIVERSITY
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Bachelor Degree Written Exam, September 5, 2023
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

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 array $a[1], a[2]....a[n]$ ($n \geq 1$) with distinct natural numbers sorted in descending order and a natural number x. Write a subalgorithm in Pseudocode, having the time complexity $O(\log_2 n)$, which returns the position *pos* of x in the array a (if x is in the array), or the position where x should be inserted in the array a so that the elements of the array remain sorted in descending order. Justify the complexity. **Observation.** Inserting an element at position k in the array involves shifting the elements at positions k, $k+1$, ..., n one position to the right and incrementing the array's length.

Example. If $n=4$, the array a is (14, 12, 9, 5) then:

- for $x=15$ the subalgorithm will return *pos*=1, as the array obtained by inserting 15 is (**15**, 14, 12, 9, 5);
- for $x=11$ the subalgorithm will return *pos*=3, as the array obtained by inserting 11 is (14, 12, **11**, 9, 5);
- for $x=12$ the subalgorithm will return *pos*=2, as the element **12** is already in the array at position 2.

Problem 2. (1.5 points)

Write a subalgorithm in Pseudocode that, given as input parameters a text represented as an array of characters $x[1]$, $x[2]....x[n]$ ($n \geq 1$) and an integer number k, $0 < k \leq n$, determines an array of characters $y[1], y[2]....y[n]$ representing the circular permutation of the array x, k positions to the right. Specify the time complexity of the subalgorithm.

Example. If $n=6$ and the text is S, T, R, I, N, G, then for $k=2$, the array y will be N, G, S, T, R, I.

Problem 3. (1.5 points)

Let us consider an array a having n elements ($a[1], ..., a[n]$), which represents a permutation of the elements $\{1, ..., n\}$ ($n>0$). Write an algorithm in Pseudocode, with worst-case time complexity $\theta(n)$, that determines whether the array a can represent the inorder traversal of a binary search tree with n nodes labeled from 1 to n. If so, display value 0, otherwise display a list of swaps (interchanges) that can produce an array that satisfies the condition. Any solution that does not fall within the indicated complexity class will receive a partial score.

Example. If $n = 3$, the array a is (3, 2, 1), then the swap 1 3 will be displayed (elements on positions 1 and 3 must be swapped, thus a possible tree whose inorder traversal is (1, 2, 3) has 2 as a root node, 1 as a left child node and 3 as a right child node).

Problem 4. (2.25 points)

Given the following (incomplete) program in C++, fill in the implementation for classes **Computer** and **SorterByTotalPrice** so that the output displayed by the **main** function is "HC85, 165". The method **sort** in class **SorterByTotalPrice** will receive as input a list of products and will sort them in descending order of total price. Sorting

will be achieved by calling the function **std::sort** from the C++ standard library. The total price of a computer is computed by adding a tax to the base price.

```
#include <string>
#include <vector>
#include <iostream>
#include <algorithm>

class Product
{
   private:
       std::string name;
       int baseprice;

   public:
       Product(const std::string& name, int baseprice):
                    name{name},
                    baseprice{baseprice} {}

       virtual int total_price() {
              return this->baseprice;
       }
       virtual std::string toString() {
              return this->name;
       }
      virtual ~Product() {}
};

class Computer: public Product {
   private:
       int tax;
       //A: missing code
};

class SorterByTotalPrice {
       //B: missing code
};

int main()
{
       std::vector<Product*> computers{new Computer{"HC90", 140, 10},
                                       new Computer{"HC91", 100, 12},
                                       new Computer{"HC85", 150, 15}};
       SorterByTotalPrice::sort(computers);
       std::cout << computers[0]->toString();
       for (auto c: computers)
           delete c;
       return 0;
}
```

Problem 5. (2.25 points)

Implement in C++ the classes required at items 1-3 and then the function required at item 4.

1. The class ***Contact*** is abstract, having a private attribute *name* of type string, an abstract method (pure virtual) **sendMessage**, which receives as input parameter a *message* of type string, and a method **getName**, which returns the value of the attribute *name*.
2. The class **Person** extends the class **Contact**, has another private attribute *number* of type string, and the method **sendMessage** receives as input parameter a message of type string and prints the person's name, number and the given message.
3. The class **Group** extends the class **Contact** and it can contain zero or more contacts. The method **addContact** receives as input parameter a contact and it adds it in the contact list. The method **sendMessage** prints the names and numbers of all contacts in the contact list, as well as the message given as input parameter.
4. Create the following objects (use any values for the attribute *number*): 2 persons named "Mother" and "Father", 2 persons named "Jane" and "John"; the group "Parents" that contains the persons "Mother" and "Father", the group "Family" that contains the group "Parents" and the person "Jane". Using the function **sendMessage**, send the following message to the group "Family" and to the person "John": "You are invited to my birthday party next week!". The memory must be correctly managed.

SUBJECT Databases

Problem 1. (4 points)

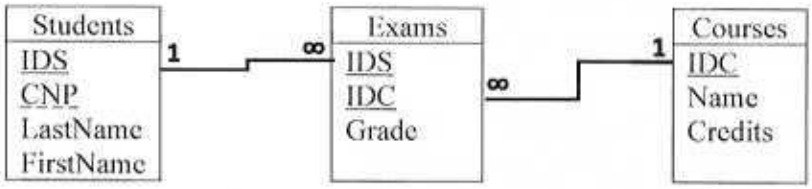
A rental bike shop stores the following data in a relational database:

- A bicycle company has an ID, a name, and a country of origin.
- A bicycle belongs to a company, has an ID, a model, a size (a number, e.g., 56), and a daily rental price.
- A customer has a CNP (Personal Identification Number), a name, a phone number, and an e-mail address.
- When a customer rents bicycles, a rental form is created. A customer can rent multiple bicycles at once. A rental form belongs to a customer, has a date on which the form was made and a list of rented bicycles, each with a start and end date for the rental period. If a customer returns on the same day to rent additional bicycles, these are added to the existing form for that day. A customer does not have multiple rental forms for the same day. If a customer comes on another day to rent bicycles, a new rental form will be created.

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:

Customers[CustomerID, CustomerName, PhoneNumber]
Drivers[DriverID, DriverName, Salary]
Cars[CarID, LicensePlateNumber, CarMake, *DriverID*]
Rides[RideID, *CustomerID*, *CarID*, RideDate, DepartureTime, RideDuration, DeparturePlace, Destination, PaymentAmount]

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 ID, name, and salary of the drivers who have the highest salary among drivers who have had at least one ride with destination „Iulius Mall Cluj" with a „Ford" make car. Eliminate duplicates.

b. Consider the following instances for relations Cars and Rides:

Cars:

CarID	LicensePlateNumber	CarMake	DriverID
1	CJ11AAA	Ford	1
2	CJ22BBB	Opel	2
3	CJ33CCC	Volkswagen	3

Rides:

Ride ID	Customer ID	Car ID	RideDate	Departure Time	Ride Duration	Departure Place	Destination	Payment Amount
1	1	1	2023.06.25	07:30	15	LP1	D1	20
2	1	2	2023.06.26	08:30	20	LP2	D3	30
3	2	1	2023.06.26	12:30	10	LP4	D4	15
4	1	1	2023.06.30	10:15	5	LP1	D2	10
5	3	3	2023.07.01	15:05	40	LP1	D4	50
6	3	2	2023.07.01	20:00	12	LP2	D2	15

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 c.DriverID, COUNT(DISTINCT r.CustomerID) No
FROM  Cars c INNER JOIN Rides r ON c.CarID = r.CarID
WHERE r.RideDate BETWEEN '2023.01.01' AND '2023.06.30'
GROUP BY c.DriverID
HAVING COUNT(DISTINCT r.CustomerID) >= ALL
        (SELECT COUNT(DISTINCT r.CustomerID)
        FROM Cars c INNER JOIN Rides r ON c.CarID = r.CarID
        WHERE r.RideDate BETWEEN '2023.01.01' AND '2023.06.30'
        GROUP BY c.DriverID)
```

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

- {RideID} → {CarID}
- {CarID} → {DeparturePlace}.

SUBJECT Operating Systems

Problem 1 (4p)
Answer the following questions about the execution of the code fragment below, knowing that function `mkfifo` creates a FIFO file, and function `unlink` deletes it.

```
1  int main(int argc, char** argv) {
2      int f; char c = 'a';
3
4      mkfifo("fifo", 0600);
5      if(fork() == 0) {
6          f = open("fifo", O_RDONLY);
7          read(f, &c, 1);
8          printf("%c\n", c);
9          close(f);
10         exit(0);
11     }
12
13     f = open("fifo", O_WRONLY);
14     write(f, "xy", 2);
15     close(f);
16
17     wait(NULL);
18     unlink("fifo");
19     return 0;
20 }
```

a) What will be printed in the console?

b) What will be printed in the console if line 18 is moved between lines 9-10?

c) How is the program execution affected if line 4 is moved between lines 5-6?

d) In what situation will the program display 'a'?

e) Explain the role of the arguments of function `mkfifo` on line 4.

Problem 2 (5p)
Answer the following questions about the execution of the UNIX Shell script below.

```
1  #!/bin/bash
2
3  for A in $*; do
4      for F in x/*; do
5          if ! grep -E -q "^$A(_+$A)*$" $F; then
6              echo "$A" >> $F
7          elif [ `grep -E "^$A(_+$A)*$" $F | wc -l` -gt 1 ]; then
8              grep -E -v "^$A(_{1,}$A){0,}$" $F > $F.aux
9              echo "$A" >> $F.aux
10             mv $F.aux $F
11         fi
12     done
13 done
```

a)	What will be the content of file `x/a.txt` seen in the column on the right, after running the script with argument `abc`?	`abc` `def`

b)	What will be the content of file `x/b.txt` seen in the column on the right, after running the script with argument `abc`?	`abc`
		`def`
		`abc`

c)	What will be the content of file `x/c.txt` seen in the column on the right, after running the script with arguments `abc def`?	`def_def`
		`abc`
		`abc_def`
		`abc_abc`
		`def_def_def`

d)	Explain in detail the condition on line 7.	
e)	Explain in detail the regular expression on line 8.	

BAREM INFORMATICĂ
VARIANTA 2

Subiect Algoritmică şi Programare

Oficiu – 1p

Problema 1. Punctaj - 1.5p

- Subalgoritm corect: **1p**
  - signatura corectă: 0.1p
  - implementare căutare binară: 0.9p
- Justificarea complexității (căutare binară): **0.5p**

Problema 2. Punctaj - 1.5p

- Subalgoritm corect implementat: **1.25p**
  - signatura corectă: 0.1p
  - implementare: 1.15p
- Complexitate $\theta(n)$: **0.25p**

Problema 3. Punctaj - 1.5p

Soluție: *șirul trebuie transformat în permutarea identică*.

- Soluție având complexitatea timp în caz defavorabil $\theta(n)$ - 1.5p
- Soluție având complexitatea timp în caz defavorabil $\theta(n \cdot \log_2 n)$ - 1p
- Soluție având complexitatea timp în caz defavorabil $\theta(n^2)$ - 0.75p
- Alte soluții care nu se încadrează în clasele de complexități de mai sus - 0.25p

Problema 4. Punctaj - 2.25p

- Implementarea clasei **SorterByTotalPrice** cu metoda statică **sort()** și folosirea funcției **std::sort**, cu o funcție lambda (**0.75p)**
  - pentru folosire **std::sort**, cu operator <
    bool operator<(Product*, Product*) 0.6p
  - pentru folosire **std::sort**, cu o functie standalone
    bool nume_functie(Product*, Product*) 0.75p
  - pentru funcții de sortare implementate (fără **std::sort**) corect 0.25p
- Implementarea metodei total_price in clasa **Computer** (**0.5p**)
- Implementarea constructorului în clasa **Computer** (**0.5p**)
- Implementarea metodei **toString()** in clasa **Computer** (**0.5p**)

Problema 5. Punctaj - 2.25p

- clasa ***Contact*** **(0.4p)**:
  - câmp privat *name*, constructor (0.1), metoda **getName()** (0.05), metoda abstractă **sendMessage** (0.15), destructor virtual (0.1).
- clasa **Person (0.35p):**
  - moștenire (0.05), câmp privat **number**, constructor (0.15), metoda **sendMessage** (0.15)
- clasa **Group (0.85p)**:
  - moștenire (0.05), câmp privat ***lista de contacte*** (0.2), constructor (0.1), metoda **addContact** (0.2), metoda **sendMessage** (0.3), destructor (a se vedea punctajul de mai jos pentru distrugerea corecta a obiectelor)
- main **(0.65p)**:
  - construire corectă obiecte **Person** (0.15), construire corectă obiecte **Group** + adăugare de contacte (0.3), apel **sendMessage** la Group si la Person (0.1), distrugere corecta obiecte (0.1)

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:

CodSofer	Nr
1	2

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

- **b2** - {CodCursa}→{CodAuto} este satisfăcută – **0.25p**; **0.25p** explicație
  - {CodAuto}→{LocPlecare} 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.

Subiect Sisteme de operare

Oficiu – 1p

Problema 1. Punctaj – 4p

a) Se va tipări 'x' – 0.5p
b) Se va tipări 'x' – 1p
c) Dacă părintele execută open înainte ca fiul să creeze fifo, fiul se blochează la open şi părintele la wait – 1p
d) Niciodată – 1p
e) Primul este numele fifo-ului de creat şi al doilea sunt permisiunile de acces – 0.5p

Problema 2. Punctaj – 5p

a) Conţinutul va fi acelaşi – 1p
b) Conţinutul va fi `def` urmat de `abc` pe altă linie – 1p
c) Conţinutul va fi `abc_def` urmat de `abc` pe altă linie urmat de `def` pe altă linie – 1p
d) Fişierul `$F` conţine cel puţin 2 linii formate exclusiv din una sau mai multe valori `$A` separate prin cel puţin câte un _ - 1p
e) Întreaga linie e o secvenţă care începe cu valoarea `$A` urmată de 0 sau mai multe secvenţe formate din unul sau mai multe _ urmate de valoarea `$A` – 1p