M1

a) false; lines that start with one of {a, e, i, o, u, A, E, I, O, U}

b) true; the shortest possible matched line is of length 3; 2 from the {2, 3}; 1 from the [^0-9]

M2

a) false; one child is created after fork() == 0, and only it reaches that part of the code, because the return code of fork() is 0 in the child process and the child PID != 0 in the parent;

b) false; both the parent and the child created by fork() evaluate the condition, that’s why only the child enters the “if” branch

M3

5 - 2 = 3;

Rows with c3 == 2 or c3 == 5: 1, 3, 5, 6, 7, 8

Unique c1, c2 in these rows: (10, 5), (20, 7), (30, 2), (30, 3), (40, 4) => 5 results

Cod1: 1 on 3 rows; 2 on 3 rows; 3 on 2 rows => 2 results

M4

{A, B} -> {D, E} - false; A = B = 1 in both rows 1 and 6, but

row 1: D = 1, E = 1

row 6: D = 2, E = 3 (diff than row 1 => not a func dep)

{CodP, B} -> {D} - true; {CodP, B} are all unique except on rows 2, 3, but on those rows D has the same value (3), therefore the dependence is satisfied

M5

3 - 2 = 1;

Fk1 = 1 on the first 3 rows; fk1, fk2 are unique on all 3 => 3 results

Fk1 = 2 on the last 2 rows; fk1, fk2 are unique on both => 2 results

M6

A..D all find the (Name, ProcesVerbalId) who took part in the accident;

a) false; it doesn't check for the different owner

b) false; it finds all people who don't own any cars - but a person who owns a car can also cause the accident, just with someone else's car

c) true; it finds all the owners of the car involved in the accident and then checks that PersoanaId is not among them

d) ? the requirement is ambiguous: “the car has another owner” - is it *PersoanaId + another owner(s)* or just *another owner(s)*; (a car can have multiple owners according to the schema)

D checks if there exists another (!= PersoanaId) owner for the car involved in the accident, but that doesn’t mean that PersoanaId is not also an owner

e) false obvs

M7

i - current index; n = len(x); x = arr;

y = product of all numbers from x whose last 2 digits are equal

j = no. of numbers without that property

returns y \* (no. of numbers with property), or 0 if there are no such numbers

M8

the program won't compile, because there is no default constructor for Vehicle; a solution is to add a def constr to Vehicle; another sol is to use member initialization lists, sth like:

Scooter(double s): Vehicle(s) {}

If we fix this, the program will run; bc virtual functions are binded at runtime and regular functions at compile time => go() is called for Scooter, accelerate() is called for Vehicle;

PS: similar situation in AP61, however the base class is abstract => this error won’t occur

M9

x = arr, n = len(x), i = crt index;

S = sum of all even numbers from x;

k = no. of odds in x;

arithmetic mean of the even numbers in x, or 0 if there are no even numbers

M10

B()

B() - the D constructor calls the B constructor

D()

B.f

Function h - dd.h() is called from a D object bc of virtual void h(), so dd is binded to a D object;

D.f - B.h -> B.g calls the f() method from D, dd is still binded

I1

nothing; it starts with 2-3 vowels, but it needs to end with a non-digit;

\* not sure about escaping - if the curly brackets are escaped, it also needs to contain the seq "{2,3}"

PS: this is if the quotes are regular ("); the ones in the pdf are different tho

I2

instead of {2,} (at least 2), use {2} (exactly 2); {2,} would have also matched strings of odd lengths

+

replace [A-Z0-9] with [A-F0-9]; hexa digits are only A-F, not A-Z;

"^[A-F0-9]\{2\}\([A-F0-9]\{2\}\)\*$"

I3?

normally it doesn't work; exec is not called exec(args), those are execvp, execlp...

but assuming it behaves like execlp, it should run "ls" in /home (lists all the files / directories in the /home directory)

I4

Displays all the lines in a file, excluding those which start with at least one “#”

I5

normal case, bc it's just a regex (no grep / sed) and the curly braces are escaped:

one of {0, 1}

"{1,}"

one of {+, -, \*, /}

"{1,}"

so definitely no binary numbers

otherwise, if we assume the special use of curly brackets:

one or more of {0, 1}

one or more of {+, -, \*, /}

still no binary numbers, because those cannot start with a zero (and this pattern can)

“\” means escaping <=> special characters lose their meaning ("\+" no longer means one or more etc.); therefore, \{\} means that we are searching for the curly brackets themselves, not their special meaning

I6

4; fork() == 0 will create the first child process, lines 2-3 are no longer executed by the parent; before line 2 there is 1 child, which become 2 after it; both child procs execute the second fork() => 2 new child procs => 4 in total

I7

Assume the for is up to n => 2^n processes in total (including the parent); all of them will print sth after their child processes finish their execution; if a proc has no child procs, it exits the while loop instantly

I9

{C1, C2} -> {C4} - true;

rows 1, 5: C1 = C2 = 1, C4 = 4

rows 2, 3: C1 = 1, C2 = 2, C4 = 3

I11

BC bubblesort = O(n)

BC merge sort = O(n log n)

BC insertion = O(n)

BC selection = O(n ^ 2)

BC quicksort = O(n log n)

not sure what "greater" means tho; if "better" => insertion; if "larger" => all others

I12

Linear (sequential) search; cannot do binary bc the numbers are not ordered

I13 - is it the continuation of I12 ?

Assume length = m

n seq searches => n \* m, no extra memory

sort + n bin searches => m log m + n log m = (m + n) log m, O(1) extra memory if we use heapsort

I14

bool isPal(int nr) {

int rev = 0, nrCopy = nr;

while (nr > 0) {

rev = rev \* 10 + nr % 10;

nr /= 10;

}

return rev == nrCopy;

}

A multiple of 10 is never a palindrome (the leading digit cannot be 0, therefore it won’t match the last digit) => returns 0

I16+17

WC quicksort = O(n ^ 2); a vector with all elements equal; or if the pivot is always the min / max in the vector

I18

BC insertion sort = O(n); a vector that already sorted in the order that we want

I19

WC insertion sort = O(n ^ 2); a vector in reverse order

I20

~~It should be a reference to the derived obj, sth like:~~

~~DerivedObj d = DerivedObj();~~

~~BaseObj& b = d;~~

I21  
Abstract class - has at least 1 abstract (not implemented) method

I22

Friend class - can access private fields of its friends

I23

Copy constructor - called when doing Point x2 = x1; (x1 is already instantiated)

I25

First the constructor of the main class is called (the default constr - if using member initialization lists, we can specify another one to be called); then the derived constr is called

AP1 - same as I4

AP2

a) false; not all lines that start with A are matched; ex. AX

b) (I assume it's multiple of 2, like for I2) false; because {2,} means "at least two times", therefore hexa sequences of odd lengths (3, 5..) are accepted; moreover, there's a [A-Z0-9] instead of [A-F0-9] (hexa letter values are only A-F);

AP3

B

AP4

sed “/^$/d” d.sh

AP5

“Increases and decreases” ⇔ additions and subtractions ?

grep -E “^0|(1[0-1]{,2})[+-]0|(1[0-1]{,2})” a.txt

“Binary number < 8” ⇔ either 0 or a number of < 3 digits, the leading digit = 1

AP6

1. true;

2. true; it deletes all lines that contain “#”; this also means the lines that start with “#”

AP7 - same as AP3 (displays only the non-commented lines)

AP8 - same as AP5

AP9

1. false;

2. false; it can also contain additions, subtractions and divisions

AP10 - same as AP3 (displays only the non-commented lines)

AP11

1. false; besides A, lines can start with B..F, or a..f, or digits

2. true; it does print those lines, but many others as well

AP12

1. true?; tho numbers such as -0000001.5 are also accepted

2. false; it replaces just the numbers themselves (all of them, because of the “g” flag), not the entire line

AP13

If we only want lines that contain such a hexa no. somewhere inside them, then remove the anchors (^, $):

grep -i “[A-F0-9]\{2\}\([A-F0-9]\{2\}\)\*”

if we also need to ensure that the number starts with a non-zero:

grep -E -i “0|[A-F1-9][A-F0-9]([A-F0-9]{2})\*”

AP14

1. false; 16

2. false; all processes will print after their child procs are finished

AP15

1. no; up to 16 procs

2. It’s not executed at all, the parent gets stuck in the while loop bc wait() never returns 0

AP17

?Both the child process and the parent will do “ls”

AP18

fork() returns -1 in case of error, so the execl line is executed in the parent process and overwrites it;

Prints all the files + directories in the crt location (classic ls), but sorts them in ascending order of the last modification time

AP19

1. true;

2. true; even processes with no children will enter the wait, but will immediately exit it with error ECHILD ([manual](https://man7.org/linux/man-pages/man2/wait.2.html)); in that case, wait() will return -1

AP20

1. true;

2. true;

AP21

1. false; see AP27, all procs will be stuck in infinite loops

2. true;

AP22

1. false; 8

2. true; wait() returns after **any** of its child procs finishes exec, and it’s not in a while loop

AP23

1. true;

2. false;

AP24

1. false; 16

2. false;

AP25

1. true; 2 processes after i = 0; 4 after i = 1..; 16 after i = 3

2. true; see AP27

AP26

$# -ge 1 ⇔ no. of command-line args >= 1

-e $1 ⇔ file given by the first cl argument exists

\* file $1 - prints the type of the file given by arg 1

AP27

1. true; the last 4 children (i = 2) are created after the fork() is executed, therefore they never reach that line

2. false; wait(NULL) waits for 1 child to finish exec and returns its PID, or -1 if error; however, this while condition is “!= 0”, and wait() never returns 0 => every process will remain stuck in an infinite loop; change the cond to “> 0” and it waits for all children to finish;

AP28

1 row, the one with ID = 2; 1 column “NumeMagazin”

AP29

5; “\_” in patterns means 1 arbitrary character

AP30

P.Nume, C.Marca, C.Denumire

Lucescu Ana, NULL, NULL

AP31

Column B; tuples (10), (35), (30)

AP33 part 1

a) true; no repeating attributes

b) true; PK has just one attribute => no key has any proper subsets => every non-prime is fully functionally dep on every key

c) false; A -> BC is ok (A - superkey), but B -> D is not (B is not superkey, D is non-prime)

d) false; not 3NF => not BCNF

AP33 part 2 - same as AP42

AP34

PK, A, C, B - 4 columns

11 tuples (1^2 with c1, 3^2 with c2, 1^2 with c3)

AP35 - same as AP30

AP36

D; Query1 returns “C102”

AP37

MA -> ND; NrP -> NP

a) false;

non-primes: {ND}, {NrP}, {NP}, {CI}

keys: {MA, DP}

MA -> ND => ND depends on a subset of the key {MA, DP} => MA is not ffd on every key => not 2NF

b) false; rows 3, 4

c) false; not 2NF => not 3NF

d) true; no repeating attributes

e) false

AP38

a) false; only 2 columns are selected

b) true;

c) false; only 2 columns are selected

d) false; it selects NumeCategorie instead of Pret

e) false; B is true

AP39

a) false; it does cross-product between T and R - normally it needs to check sth like T.ID = R.ID

b) false; it does a natural join between a table of just T.Name and table S - they don't have any common columns => empty result

c) true; this one applies the projection on the result of the natural joins

d) false; intersection between a column of just names and table S; also there’s no “author” in Teatre \* Repertoriu

e) false; C is true

AP40

5 - 4 = 1

AP41

C

"a record r belongs to the R1/R2 if in R1 r is concatenated with every record in R2" => a Client is in the result if is concatenated with every Movie

AP42

A) 0

AP43 - same as AP33

AP44

key = {U, V}

nonprime = {X}, {Y}

V -> Y, X -> Y, X -> U

bc of V -> Y => Y is not ffd (it depends on a subset of a key - V) => not 2NF => remove it

AP45

"a candidate can take several exams with the same car and same cop" => (candidate, car, cop) is not unique => cannot be key

“a candidate can take just one exam on a day” => (candidate, date) is unique => can be key => anything that contains (candidate, date) can also be key

AP46

no, because B = 4 determines both D = 3 and D = 2 (rows 2, 3)

AP47

yes;

(A, B) = (1, 1) only determines (D, E) = (2, 2) (rows 1, 2)

(A, B) = (2, 2) .. (D, E) = (2, 2) (row 3)

(A, B) = (3, 4) .. (D, E) = (4, 5) (rows 4, 5)

AP48

C - though B also looks ok ??; not A bc not 1NF (repetitive)

AP49

A) false; k will never be > n/2 => doesn’t work for numbers that would be put in the second half

B) true; basic binary search

C) false; can end up with infinite loop

D) true; also binary search

E) false; j + anything goes past n from the start

AP50

n1 = 5 \* n => n1 = 80

n2 = 1 + log2(n) + 1 + 2 + 4 + 8 + 16 + .. x^2 <= n => n2 = 36

AP51

A) true

B) false

C) ?false; it does compute that mirror, but only as a partial result and it does not return it or anything

D) false

E) true; multiples of 10 cannot be palindromes, bc 0 cannot be a leading digit

AP52

Bubblesort in increasing order; those 3 lines inside the if ⇔ swap; the p is useless tho

AP53

A) false; n = 6 (20 in base 3), m = 8 (22 in base 3) => k = -1 => returns false, but it should have returned true

B) false; n = 6, m = 8 => k = -1 => returns false

C) true; the opposite of A

D) true; k doesn’t change if the crt digits are equal; if n = m => the crt digits are always equal => k remains 0 => returns true; the question doesn’t say anything about *returning false if the condition is not satisfied*, like A-C

AP54

B

AP55

Determines the “peak” of the mountain sequence ⇔ the maximum number in the array

AP56

A; tho I can’t really see the code, but it’s the one that compares plants[j] to plants[ind] s.t. ind becomes the index of the minimum in that range (it’s selection sort)

AP57

A

B::f - because A.f() is virtual; if we remove that, it prints A::f

AP58

C

AP59

B\* b = new D(new B())

AP60

B; there’s an extra call to the copy constructor when calling functionA(A a) - in order for “a” to be passed as value

AP61

A) false; Shape is abstract, so is Polygon => cannot be instantiated

B) true;

C) false; a Square can be instantiated if we comment out lines 1 and 2

D) false;

AP62

C; for x[1] it only enters cook() from Pizza

AP63 - same as AP59

POCU1

A; 6 - 1 = 5

POCU2

A - if the for was only up to n / 2, it would have reversed the array

POCU3

C is candidate key (only one not in a right-hand side of any dep)

A) false; C -> AD is ok; AC -> E is ok; E -> B is not ok (E is not superkey and B is non-prime)

B) false; not 3NF => not BCNF

C) false; D, E are true

D) true; no repetitive atts

E) true; the only key is made of 1 attr => no subsets => every non-prime is ffd

POCU4

A) true; speed is protected in the diagram, but private in the implementation => the implementation does **NOT** match

B) true; Helicopter implements FlyBehaviour, but does not implement the fly() method (no args) => Helicopter is abstract => can**NOT** be implemented

C) true;

D) false; there **is** an association between the 2 classes

POCU5

C - the virtual from class A is “trasmitted” all the way to class C, so the method is binded at runtime => it is executed as if from a C object;

POCU6

A) false; inner join ON id spectacol = id locatie, makes no sense

B) true; we need the no. of shows from every location on every date => we group by location and date; HOWEVER, in order to select an attr, we need it in group by => we also need the name in group by

C) false; ^

D) false; inner join ON id spectacol = id locatie

POCU7

C???; linked hash set preserves the insertion order; tree set orders the elements naturally (in increasing order); **???hash set apparently does not guarantee order, although we can probably assume the insertion order???**

POCU8

D; the query is only executed when needed ⇔ when calling .ToList() on it

POCU9

A; C; D

POCU10

B; E

POCU11

D

POCU12

(assume we need to show the dates + the max value, cause the latter is not specified)

A) true

B) false; sum(\*) would also sum the Date column

C) false; we need max, so >= ALL instead of ANY from the start (but also sum instead of count, see D)

D) false; using count(\*) we get the date when the most orders were made (the date that appears the most has the highest count), not the date with the most ordered pieces

POCU13

D

POCU14

A; [helpful](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/knowing-when-to-use-override-and-new-keywords)

a - there’s no *virtual* so the binding is done at compile time => it will attempt to call area() from Shape (which doesn’t have one), then Square => area from square

b - shape doesn’t have a concrete area(), so it’s just the one from square

c - again no *virtual* => area from square

d - type rectangle, value rectangle => area from rectangle

POCU15

B) 80 - when doing aggregate(distinct ..), NULL is ignored => distinct returns just {3, 4}

POCU16

A

POCU17

D - compiling error because Rectangle is abstract => cannot declare obj of that type

POCU18

A) true; all DataNasterii are unique

B) false; age 29 (rows 4, 5) determines different salaries (2200, 2500)

C) true; java determines only 2200, php only 2500, c# only 3000

D) false; Toma -> both Ana and Ioan

E) true; all (Nume, Prenume) are unique

POCU19

C; [helpful](https://www.tutorialsteacher.com/csharp/csharp-delegates)

the custom del will have 3 methods inside it: Hello (which prints that “Hello”), the anonymous one which receives x as arg (which does not print anything), then Goodbye, which prints the “Goodbye” and its return value = “Goodbye John” (“John” is the argument)

POCU20

A) false; a ticket motive cannot be "plangere" and "informatii" simultaneously

B) false; disjunction instead of conjunction

C) false; union instead of intersection

D) true;

POCU21

D; cc1.getInstance() creates a new (non-null) instance inside cc1 and attributes it to cc2 ⇔ cc2 = new CC(); cc2 itself is not null, but the instance inside it still is;

Print cc1: instance is not null => “CC object” + cc2.toString() = “CC object” + “null”

Print cc2: “null”

**Subiect 2019 EN**

DB

1.

PKs = PID, ProcID, (PacientID, ProcedureID, Date, DoctorLName, DoctorFName)

FK = PacientID - PID, ProcedureID - ProcID

2.

Date of birth -> age; (doctorLName, doctorFName) -> doctorSpeciality; (doctorLName, doctorFName) -> doctorSeniority

3.

a) yes; doctor speciality and seniority depend on doctor last name and first name, which is a subset of a key => they are not ffd => not 2NF

b) no;

c) yes; for age -> date of birth, age is not superkey, dob is not prime => not 3NF;

d) no;

e) no;

4.

SELECT LastName, FirstName

FROM PMR INNER JOIN PACIENTS ON PacientID = PID

WHERE DS = “Radiology”

INTERSECT

SELECT LastName, FirstName

FROM PMR INNER JOIN PACIENTS ON PacientID = PID

WHERE DS = “Cardiology”

5.

SELECT PMR.DoctorSpeciality as Speciality, COUNT(\*) as NumberOfProcedures, SUM(P.Price) as AmountOfMoney

FROM PMR INNER JOIN Procedures P ON P.ProcID = PMR.ProcedureID

GROUP BY PMR.DoctorSpeciality

WHERE SUM(P.Price) >= ALL (

SELECT SUM(P.Price)

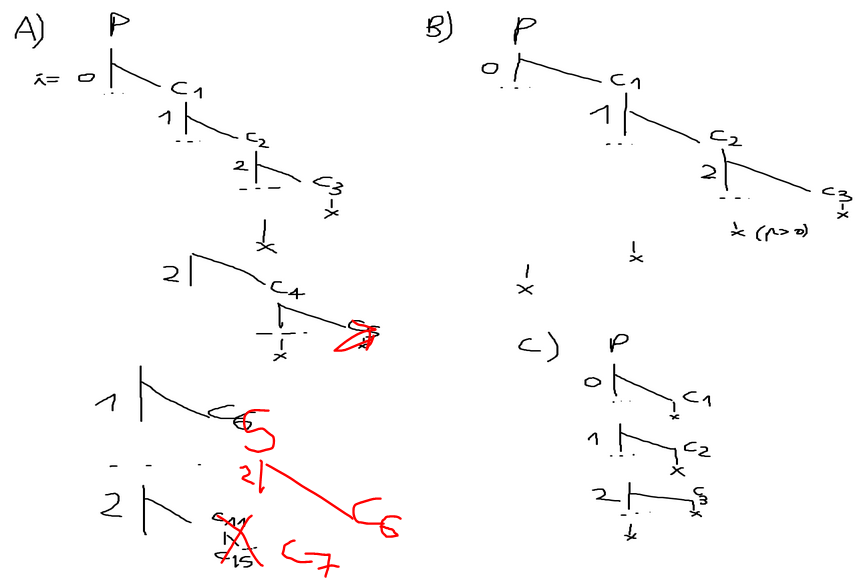
FROM PMR INNER JOIN Procedures P ON P.ProcID = PMR.ProcedureID

GROUP BY PMR.DoctorSpeciality

)

OS

1.



c)

for (int i = 0; i < 3; i++) {

if (fork() == 0) {

printf("child proc\n");

break;

}

else {

print("parent proc\n");

wait(0);

}

}

d) when called from process p, waits for ANY child process of p to finish; returns -1 in case of error (ex. no children to wait on), returns child PID (> 0) on success

2.

a) -n = non empty string; -d = exists and is directory

grep -v inverts the selection (select non-matching); grep -q does not print anything (quiet)

a.text is created; it contains the text from “a”, without the lines that start with “apples”; same for “b”

“Error: x.txt”

b.text is created, like ^, but without “pears”

b) the command “file” prints the file type (ex. “text file”); the output of this command is passed through a pipe (“|”) to a grep which selects all lines that don’t match that file type, without printing anything to the console; if the grep finds sth => the if condition is true => lines 6, 7 are executed

c) the $2 is no longer interpreted as the second command line argument => the sed does nothing; if we want to preserve that interpretation, add single quotes around $2 too

d) the program gets stuck in an infinite loop whenever the current cl argument is not a directory; in this example, the first cl argument is not a directory => the program remains stuck there

**Subject 2018 EN**

DB

1.

PKs = (TID), (PID), (T1ID, T2ID, Date)

FKs = TeamID -> TID; Team1ID -> TID; Team2ID -> TID

2.

Country -> continent;

Stadium -> no. of seats

Stadium -> city

Stadium -> country

3.

a) yes; continent (non prime) depends on country (not a superkey) => not 3NF;

b) no;

c) yes; for ex. city (non prime) depends on stadium (not a superkey) => not 3NF;

d) no;

4.

SELECT Name

FROM Teams T INNER JOIN Players P ON TID = TeamID

WHERE T.Continent = “Asia” AND P.NumberOfMatches > 100

5.

SELECT T.Country, COUNT(\*) as NoMatches

FROM Teams T INNER JOIN Matches M ON (TID IN (TeamID1, TeamID2))

GROUP BY TID, T.Country

OS

1.

a) K child processes

b) nothing is printed; the (only) child process gets stuck when opening the pipe for reading, because nobody opened it for writing on the other side “*if the FIFO is opened for reading, the process will "block" until some other process opens it for writing. This action works vice-versa as well. If this behavior is undesirable, the O\_NONBLOCK flag can be used*”

c) prints “x” and “y” in any order; bc now there are 2 processes, each with its own reading / writing fifos, they no longer get stuck and send to each other the values “x” and “y”, which get printed; after the fifos are closed, the child processes are killed; we can’t determine the order in which they are printed; theoretically both will try to write to / read from each other at the same time

e) nothing; there’s a deadlock; C1 attempts to open “p” for reading; C2 attempts to open “q” for reading; both get stuck waiting for the other proc to open sth for writing

**Subject 2017 EN**

OS

1.

a) 1 parent and 1 child

b)

Result:\n - P

exam passed\n - C

c) 15

d) the child process will remain stuck trying to read from the pipe (line 7); the parent will also wait for the child to finish (line 21) => deadlock

e) child proc is stuck^; but now the parent no longer waits on it => the parent proc finishes => its pipe end is closed automatically => child gets unstuck => it finishes as well

2.

a) 0; it checks whether the name + path of a file is the same as a folder’s, which cannot happen

b) a list of all the files (with their paths) inside the crt dir

c) a list of all the directories (with their paths) inside the crt dir

d) the name of a file from the list $f

e) the name of a directory from the list $d

DB

Venues(VID, Name, Address)

Movies(MID, Title, Year)

Genres(GID, Name, Desc)

MovieGenreRelation(MID, GID)

Actors(AID, Name)

MovieActorRelation(MID, AID)

Screenings(SID, VID, DateTime, MID)

SoldTickets(SID, RowNr, SeatNr)

No func dep lol

a)

SELECT V.Name, V.Address

FROM Venues V INNER JOIN Screenings S ON S.VID = V.VID

WHERE “comedy” IN (

SELECT G.Name

FROM Genre G INNER JOIN MovieGenreRelation MGR ON MGR.GID = G.GID

WHERE MGR.MID = S.MID

)

INTERSECT

SELECT V.Name, V.Address

FROM Venues V INNER JOIN Screenings S ON S.VID = V.VID

WHERE “drama” IN (

SELECT G.Name

FROM Genre G INNER JOIN MovieGenreRelation MGR ON MGR.GID = G.GID

WHERE MGR.MID = S.MID

)

b)

SELECT COUNT(\*) AS TicketsSold

FROM SoldTickets ST INNER JOIN Screenings S ON S.SID = ST.SID

INNER JOIN MovieActorRelation MAR ON MAR.MID = S.MID

GROUP BY ST.SID, ST. RowNr, ST.SeatNr -- PK for ST

WHERE S.VID = (

SELECT VID  
 FROM Venues V

WHERE V.Name = “Piata Unirii”

) AND MAR.AID = (

SELECT AID

FROM Actors A

WHERE A.Name = “Alain Delon”

)

c)

SELECT M.Title, M.Year

FROM Movies M INNER JOIN Screenings S ON S.MID = M.MID

INNER JOIN SoldTickets ST ON ST.SID = S.SID

GROUP BY M.MID

WHERE COUNT(\*) >= ALL (

SELECT COUNT(\*)

FROM Movies M INNER JOIN Screenings S ON S.MID = M.MID

INNER JOIN SoldTickets ST ON ST.SID = S.SID

)

**Subject 2016 EN**

DB

Tourists (TID, Name, Email, CityCode, CountryCode)

Accommodation (AID, Name, CityCode, CountryCode, TypeCode, Stars, Rating, Price)

City (Code, Name)

Country (Code, Name)

Type (Code, Name)

Booking (TID, AID, StartDate, Nights)

1.

SELECT T.Name, T.Email

FROM Tourists T INNER JOIN Booking B ON B.TID = T.TID

INNER JOIN Accommodation A ON A.AID = B.AID

WHERE A.Rating > 9 AND T.Name = “pension”

MINUS

SELECT T.Name, T.Email

FROM Tourists T INNER JOIN Booking B ON B.TID = T.TID

INNER JOIN Accommodation A ON A.AID = B.AID

WHERE A.Rating < 9 AND T.Name = “hotel” AND A.Stars = 3

2.

SELECT COUNT(\*)  
FROM Booking B INNER JOIN Tourists T ON T.TID = B.TID

INNER JOIN Accommodation A ON A.AID = B.AID

INNER JOIN City C1 ON C1.Code = T.CityCode -- tourist city

INNER JOIN City C2 ON C2.Code = A.CityCode -- accommodation city

WHERE B.StartDate = 2015 AND C1.Name = “Cluj” AND C2.Name = “Paris” AND A.Stars = 5

3.

SELECT A.Name, C.Name, T.Name, A.Stars

FROM Accommodation A INNER JOIN City C ON A.CityCode = C.Code

INNER JOIN Type T ON A.TypeCode = T.Code

INNER JOIN Bookings B ON B.AID = A.AID

GROUP BY A.AID, A.Name, C.Name, T.Name, A.Stars

WHERE COUNT(\*) >= ALL (

SELECT COUNT(\*)

FROM Booking B

WHERE B.AID = A.AID AND B.StartDate >= 2017 -- assume we are in 2022

)

OS

1.

2.

a) it matches the lines that contain at least 1 digit and nothing else

b) it will wait for input from stdin

c) “cut -c1” only gets the first char

New file: “a” {abc aa}

New file: “f3.nr” {74}

New file: “b” {b2}

Prints n = 3

d) abc bcd cde def

De la germana

DB1\_1

E Nr -- first part

e1 2

e2 1

e3 1

E Nr -- second part

e1 4

e2 4

e3 4

=> result = first part

DB1\_2

1 Lufthansa

2 Wizzair

It determines the companies with the most flights

AP1\_1

8 2 4 3 7 6 5 1 9 2 1 6

AP2\_1

The program doesn’t compile

Line 1 - Shape is abstract

Line 2 - Polygon is abstract

Line 3 - creates a Square instance with length = 5

Line 4 - prints “”Polygon:Square 25”

AP2

new D(new B());

b is an obj of type D; f() is virtual in B => it is dynamically binded => b->f() is called as if from a D obj;

BD2

“An employee cannot have 2 reservations at the same time” => (employee, date) - unique

“A client cannot have 2 reservations at the same time” => (client ,date) - unique

Obvs (employee, client, date) - unique

“Candidate key” = set of atts that uniquely determine the tuples in a table ⇔ any unique set of atts can be a candidate key => all 3 can be candidate keys

SO2

a) true; fork returns -1 < 0 if the child cannot be created

b) true; the child does not exist => only the parent can enter that if branch

BD1

Yes; Q2 does a cross join + condition on PasID and CAID ⇔ condition join (inner join);

Nume in (“lufthansa”, “wizzair”) ⇔ nume = lufthansa or nume = wizzair

DB1

1 - 3 = -2

SO1

a) false; maybe there are no cl argos

b) false; the output of the while loop is printed to a file “f”

OS1

a) false; ex. “15” is not found bc it has to start with a minus and to have a decimal point

b) false; ex. “-15” is not found bc it does not have a decimal point

AP1

“][)(}{” - the no. of opening / closing parentheses for each type is ok, but their order is not

OS2 - same as SO2