View->Web Layout – For the testCases to appear better

EN

# 1.

In the C programming language, write a structure named "Movie" which represents a movie within a streaming platform. The Movie structure should contain the following fields: **(5 points)**

* Title (array of 100 characters)
* Rating (real number)
* Genre (array of 40 characters)

Write a structure named "StreamingService" that contains the following fields: **(5 points)**

* Name of the streaming platform (array of 50 characters)
* Array of available movies (an array of maximum of 30 elements of the Movie structure)
* Number of elements in the array (integer)

Create a function named "print" that prints the information for a given streaming platform passed as a parameter, in the following format: **(10 points)**

[Name of streaming platform]

[Title1] [Rating1]

[Title2] [Rating2]

[Title3] [Rating3]

...

Create a function named "showBestStreamingService" that takes an array of variables of type StreamingService and a variable named "genre" as parameters. This function should print the name of the streaming platform that offers the most movies of the given genre (depending on the "genre" parameter). If multiple streaming platforms have the same maximum number of movies of the given genre, the first one should be printed. **(15 points)**

Complete the main function **(5 points)**

# 2.

To enable precise control and records in an airport service, it is necessary to implement a module with the following classes and functionalities:

Implement an Airplane class that will store information about:

* Registration mark (an array of up to 12 characters)
* Name of the airline that owns the airplane (dynamically allocated array of chars)
* Airline home airport (an array of up to 20 characters)
* Status (the status of the airplane that can be: NOT\_WORKING, REPAIRING or WORKING)
* Total number of flight hours (integer)

For the class implement the necessary constructors, = operator, and methods for the solution to work. **(10 points).** Additionally, implement a void print() method to print the airplane information (according to the test examples). **(5 points)**

Implement an AirportService class that will store information about:

* Name of the airport where the service is located (a string of up to 20 characters)
* List of airplanes currently in service (a dynamically allocated array of objects of the Airplane class, initially with no elements)
* Number of airplanes in service (number of elements in the array - integer, initially 0)

For the class implement the necessary constructors and methods in order for the solution to work. **(5 points)**Additionally in the AirportService class also implement:

* method void addAirplane (Airplane plane) - for adding a new plane to the service **(5 points).** An airplane can only be added to the service if it is in NOT\_WORKING status. Additionally, if there is already an airplane from the same airline in the service as the airplane we want to add, only the airplane with a greater number of flight hours should remain in the service. When an airplane is added to the service, its status changes to REPAIRING **(10 points).**
* method Airplane serviceOldestAirplane() - to perform a service on the oldest airplane. The oldest plane is considered the plane from the list with the most flight hours. Servicing an aircraft involves:
  + The aircraft needs to be removed from the list of aircraft in the service
  + The status of the serviced aircraft should transition from REPAIRING state to WORKING state and should be returned as a result
  + If there are several airplanes with the same number of flight hours when determining the oldest plane, then it is necessary to service the plane that has the same home airport as the airport where the service is located **(15 points)**
* method void print() - for printing information about all the airplanes that are currently being repaired in the service (format in test examples)**(10 points)**

| * **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 1 | TESTING CONSTRUCTOR FOR AIRPLANE  TEST FOR DEFAULT CONSTRUCTOR PASSED  TEST FOR CONSTRUCTOR WITH 3 ARGUMENTS PASSED  TEST FOR CONSTRUCTOR WITH 4 ARGUMENTS PASSED | TESTING CONSTRUCTOR FOR AIRPLANE  TEST FOR DEFAULT CONSTRUCTOR PASSED  TEST FOR CONSTRUCTOR WITH 3 ARGUMENTS PASSED  TEST FOR CONSTRUCTOR WITH 4 ARGUMENTS PASSED |  |
|  | 2 | TESTING COPY-CONSTRUCTOR AND OPERATOR = (ASSIGNMENT) FOR AIRPLANE  TEST FOR COPY CONSTRUCTOR PASSED  TEST FOR OPERATOR = (ASSIGNMENT) PASSED | TESTING COPY-CONSTRUCTOR AND OPERATOR = (ASSIGNMENT) FOR AIRPLANE  TEST FOR COPY CONSTRUCTOR PASSED  TEST FOR OPERATOR = (ASSIGNMENT) PASSED |  |
|  | 3  ZHN-96-TA  FINKI-Airline  TMF  13  0 | TESTING PRINT() FOR AIRPLANE  ID: ZHN-96-TA - NOT\_WORKING FINKI-Airline, 13, TMF | TESTING PRINT() FOR AIRPLANE  ID: ZHN-96-TA - NOT\_WORKING FINKI-Airline, 13, TMF |  |
|  | 3  ZHN-96-RT  FINKI-Airline  MF  50  2 | TESTING PRINT() FOR AIRPLANE  ID: ZHN-96-RT - WORKING FINKI-Airline, 50, MF | TESTING PRINT() FOR AIRPLANE  ID: ZHN-96-RT - WORKING FINKI-Airline, 50, MF |  |
|  | 4 | TESTING CONSTRUCTOR FOR AIRPORTSERVICE  TEST PASSED | TESTING CONSTRUCTOR FOR AIRPORTSERVICE  TEST PASSED |  |
|  | 5  2  ZHN-96-RT  FINKI-Airline  MF  50  0  ZHN-96-RA  F-Airline  TMF  50  0 | TESTING ADD() AND PRINT() FOR AIRPORTSERVICE  FINKI  ID: ZHN-96-RT - REPAIRING FINKI-Airline, 50, MF  ID: ZHN-96-RA - REPAIRING F-Airline, 50, TMF | TESTING ADD() AND PRINT() FOR AIRPORTSERVICE  FINKI  ID: ZHN-96-RT - REPAIRING FINKI-Airline, 50, MF  ID: ZHN-96-RA - REPAIRING F-Airline, 50, TMF |  |
|  | 5  2  ZHN-96-RT  FINKI-Airline  MF  50  0  ZHN-96-RA  F-Airline  TMF  50  1 | TESTING ADD() AND PRINT() FOR AIRPORTSERVICE  FINKI  ID: ZHN-96-RT - REPAIRING FINKI-Airline, 50, MF | TESTING ADD() AND PRINT() FOR AIRPORTSERVICE  FINKI  ID: ZHN-96-RT - REPAIRING FINKI-Airline, 50, MF |  |
|  | 5  3  ZHN-96-RT  FINKI-Airline  MF  40  0  ZHN-96-TT  F-Airline  FR  120  0  ZHN-96-RA  FINKI-Airline  TMF  50  0 | TESTING ADD() AND PRINT() FOR AIRPORTSERVICE  FINKI  ID: ZHN-96-RA - REPAIRING FINKI-Airline, 50, TMF  ID: ZHN-96-TT - REPAIRING F-Airline, 120, FR | TESTING ADD() AND PRINT() FOR AIRPORTSERVICE  FINKI  ID: ZHN-96-RA - REPAIRING FINKI-Airline, 50, TMF  ID: ZHN-96-TT - REPAIRING F-Airline, 120, FR |  |
|  | 6  3  ZHN-96-RT  FINKI-Airline  MF  40  0  ZHN-96-TT  F-Airline  FR  120  0  ZHN-96-RA  FINKI-Airline  TMF  50  0 | TESTING serviceOldestAirplane() AND PRINT() FOR AIRPORTSERVICE  Removed plane:  ID: ZHN-96-TT - WORKING F-Airline, 120, FR  -----------------  FINKI  ID: ZHN-96-RA - REPAIRING FINKI-Airline, 50, TMF | TESTING serviceOldestAirplane() AND PRINT() FOR AIRPORTSERVICE  Removed plane:  ID: ZHN-96-TT - WORKING F-Airline, 120, FR  -----------------  FINKI  ID: ZHN-96-RA - REPAIRING FINKI-Airline, 50, TMF |  |
|  | 6  3  ZHN-96-RT  FINKI-Airline  MF  40  0  ZHN-96-TT  F-Airline  FR  120  0  ZHN-96-RA  FINKI-Airline  FINKI  120  0 | TESTING serviceOldestAirplane() AND PRINT() FOR AIRPORTSERVICE  Removed plane:  ID: ZHN-96-RA - WORKING FINKI-Airline, 120, FINKI  -----------------  FINKI  ID: ZHN-96-TT - REPAIRING F-Airline, 120, FR | TESTING serviceOldestAirplane() AND PRINT() FOR AIRPORTSERVICE  Removed plane:  ID: ZHN-96-RA - WORKING FINKI-Airline, 120, FINKI  -----------------  FINKI  ID: ZHN-96-TT - REPAIRING F-Airline, 120, FR |  |
|  | 7 | TESTING COPY CONSTRUCTOR AND OPERATOR = FOR AIRPORTSERVICE  FINKI  ID: ZHN-96-TY - REPAIRING FINKI-Airline, 13, TMF  FINKI  ID: ZHN-96-TY - REPAIRING FINKI-Airline, 13, TMF | TESTING COPY CONSTRUCTOR AND OPERATOR = FOR AIRPORTSERVICE  FINKI  ID: ZHN-96-TY - REPAIRING FINKI-Airline, 13, TMF  FINKI  ID: ZHN-96-TY - REPAIRING FINKI-Airline, 13, TMF |  |

Passed

MK

# 1.

Во програмскиот јазик C да се креира структура UmetnickoDelo за опишување на уметничко дело во дадена галерија. Структурата треба да содржи: **(5 поени)**

* Автор (низа од 100 знаци)
* Цена(реален број)
* Тип на уметничко дело (цел број со вредност 1 за слика, 2 за скулптура)

Да се креира структура Galerija, што содржи: **(5 поени)**

* Име на галеријата (низа од 50 знаци)
* Низа од достапни уметнички дела (низа од максимум 30 елементи од структурата UmetnickoDelo)
* Број на елементи во низата (цел број)

Да се креира глобална функција print за печатење на информациите за дадена галерија во следниот формат: **(10 поени)**

[Ime na galerijata]

[Avtor1] [Cena1] [Tip1]

[Avtor2] [Cena2] [Tip2]

[Avtor3] [Cena3] [Tip3]

...

Да се креира функција najdobra\_ponuda, што прима низа од променливи од типот Galerija, и променлива tipNaDelo. Оваа функција треба да го испечати името на галеријата која ја нуди најевтината уметничка слика (за променлива со вредност 1) и името на галеријата која ја нуди најевтината скулптура (за променлива со вредност 2). Најевтино уметничко дело е уметничкото дело кое има најнска цена. Ако има повеќе такви галерии со најевтино уметничко дело, се печати последната.**(15 поени)**

**For example:**

| **Input** | **Result** |
| --- | --- |
| 2  Galerija1 3  Pablo\_Picasso 5000 1  Leonardo\_da\_Vinci 8000 1  Michelangelo 4500 2  Galerija2 2  Vincent\_van\_Gogh 6000 1  Salvador\_Dali 3000 2  2 | Galerija1  Pablo\_Picasso 5000.00 1  Leonardo\_da\_Vinci 8000.00 1  Michelangelo 4500.00 2  Galerija2  Vincent\_van\_Gogh 6000.00 1  Salvador\_Dali 3000.00 2  Najevtino umetnicko delo tip skulptura go nudi galerijata: Galerija2 so cena: 3000.00 |
| 2  Galerija1 3  Pablo\_Picasso 5000 1  Leonardo\_da\_Vinci 8000 1  Michelangelo 4500 2  Galerija2 2  Vincent\_van\_Gogh 6000 1  Salvador\_Dali 3000 2  1 | Galerija1  Pablo\_Picasso 5000.00 1  Leonardo\_da\_Vinci 8000.00 1  Michelangelo 4500.00 2  Galerija2  Vincent\_van\_Gogh 6000.00 1  Salvador\_Dali 3000.00 2  Najevtino umetnicko delo tip slika go nudi galerijata: Galerija1 so cena: 5000.00 |
| 1  ArtMGM 3  Picasso 10000 1  Rodin 20000 1  Van\_Gogh 12000 1  2 | ArtMGM  Picasso 10000.00 1  Rodin 20000.00 1  Van\_Gogh 12000.00 1  Ne e pronajdeno umetnicko delo od tip skulptura |

# 2.

За потребите на апликациите за нарачка и доставување на храна (пр. Корпа, Кликни Јади и сл.) потребно е да се имплементира модул за управување со доставувачите.

Да се имплементира класа DeliveryPerson во која ќе се чуваат информации за:

* ИД на доставувачот (низа од најмногу 5 знаци)
* Име на доставувачот (динамички алоцирана низа од знаци)
* Локација - реон во кој моментално се наоѓа доставувачот (цел број)
* Број на извршени достави (цел број, иницијално е 0)

За класата да се имплементираат потребните конструктори, оператор = и методи за да функционира решението. **(10 поени)**. Дополнително за класата да се имплементира метод void print() за печатење на информациите за доставувачот. **(5 поени)**

Да се имплементира класа DeliveryApp во која ќе се чуваат информации за:

* Име на апликацијата (низа од најмногу 20 знаци)
* Листа на доставувачи кои работат за оваа компанија (динимички алоцирана низа од објекти од класата DeliveryPerson, иницијално без елементи)
* Број на доставувачи (број на елементи во низата од доставувачи - цел број, иницијално е 0)

За класата да се имплементираат потребните конструктори и методи со цел да функционира решението. **(5 поени)** Дополнително во класата да се имплементираат:

* метод void addDeliveryPerson (const DeliveryPerson & person) - за додавање нов доставувач во апликацијата. **(10 поени)** Да не се дозволи додавање на доставувач со веќе постоечко ИД. **(5 поени)**
* метод void assignDelivery (int restaurantArea, int customerArea) - за доделување на нарачка на доставувачот кој е најблиску до реонот на ресторанот restaurantArea од кој треба да се подигне нарачката. При доделување на нарачка на доставувач важат следните правила:
  + Просторот на кој функционира апликацијата е поделен на 10 соседни реони. Пример доколку ресторанот е во реон 5, прво се проверува дали има достапен доставувач во самиот тој реон (5), па потоа дали има доставувачи во првите соседни реони (4 и 6), па во вторите соседни реони (3 и 7) итн.
  + Доколку има повеќе доставувачи во реонот што е најблиску до посакуваниот реон, се избира доставувачот кој има извршено најмалку доставувања досега.
  + Откако нарачката е доделена на доставувачот, се менува неговата локација на реонот во кој се доставува нарачката (customerArea) и се зголемува бројот на извршени достави за доставувачот. **(15 поени)**
* метод void print() - за печатење на информации за сите доставувачи кои работат за компанијата (формат во тест примери). **(10 поени)**

**Објаснување за вториот тест пример:**

Апликацијата има четири доставувачи: Стефан (лоциран во 2), Влатко (лоциран во 3), Стојменски (лоциран во 8) и Тенев (лоциран во 1). 

* Првата нарачка е од реон 2 во 4. Стефан е оддалечен 0 реони од реон 2, Влатко е оддалечен 1 реон, Стојменски е оддалечен 6 реони и Тенев е оддалечен 1 реон. Нарачката ќе му биде доделена на Стефан бидејќи е најблиску и неговата локација ќе биде променета во 4.
* Втората нарачка е од реон 3 во 5. Стефан е оддалечен 1 реон, Влатко 0 реони, Стојменски 5 реони и Тенев 2 реони. Нарачката ќе му биде доделена на Влатко бидејќи е најблиску и неговата локација ќе биде променета во 5.
* Третата нарачка е од реон 2 во 9. Најблиску до ова нарачка е Тенев (оддалечен само 1 реон до реонот 2). Нарачката ќе му биде доделена на Тенев бидејќи е најблиску и неговата локација ќе биде променета во 9.
* Четвртата нарачка е од реон 8 во реон 10. Најблиску до ова нарачка е Стојменски (оддалечен 0 реони до ресторанот), па тој ќе ја подигне нарачката.

Во овој пример, сите доставувачи подигнуваат точно по една нарачка.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  Korpa  4  11111 Stefan 2  11112 Vlatko 3  11113 Stojmenski 8  11114 Tenev 8 | DeliveryApp addDeliveryPerson test  Korpa  ID: 11111 Name: Stefan Current location: 2 # of deliveries: 0  ID: 11112 Name: Vlatko Current location: 3 # of deliveries: 0  ID: 11113 Name: Stojmenski Current location: 8 # of deliveries: 0  ID: 11114 Name: Tenev Current location: 8 # of deliveries: 0 |
| 6  Korpa  4  11111 Stefan 2  11112 Vlatko 3  11113 Stojmenski 8  11114 Tenev 1  4  2 4  3 5  2 9  8 10 | DeliveryApp addDeliveryPerson test  Korpa  ID: 11111 Name: Stefan Current location: 4 # of deliveries: 1  ID: 11112 Name: Vlatko Current location: 5 # of deliveries: 1  ID: 11113 Name: Stojmenski Current location: 10 # of deliveries: 1  ID: 11114 Name: Tenev Current location: 9 # of deliveries: 1 |