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

LAB 1

# 1.

Use the **C** programming language to solve this exercise.  
  
Write a program which from standard input receives the integer **N** (Number of transactions), followed by information about **N transactions**, and outputs the identification number and total sum (with commission) of all the transactions paid with a credit card. In edge cases where there are no transactions which have been paid with a credit card, output "No credit card transaction".  
  
The structure Transaction must have the following information:  
- Identification number (int)  
- Paid amount (int)  
- Commission (int)  
- Was the transaction paid with a debit or credit card (0 - debit, 1 - credit)  
  
Input limit: 0 < N < 100  
  
Input format:  
N  
id1 sum1 commission1 type1  
...  
idn sumn Commissionn typen

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 3  1 100 10 0  2 202 20 1  3 10 1 1 | 2 222  3 11 | 2 222  3 11 |  |
|  | 5  1 10 213 0  2 20 123 0  3 30 587 0  4 40 342 0  5 50 342 0 | No credit card transaction | No credit card transaction |  |
|  | 4  1 10 100 1  20 23 200 0  300 33 300 0  4000 23 420 0 | 1 110 | 1 110 |  |
|  | 5  10 12334 1920 1  22 23894 2202 0  23 23421 2022 0  58 10000 1000 1  223 982345 10 0 | 10 14254  58 11000 | 10 14254  58 11000 |  |
|  | 10  1 209 13 0  2 123 2313 1  3 123 1 0  4 12381 12312 1  5 12317 191 0  6 67898 123 1  7 67485 2189 0  8 786123 101 1  9 81792 10123 0  10 12893 10213 1 | 2 2436  4 24693  6 68021  8 786224  10 23106 | 2 2436  4 24693  6 68021  8 786224  10 23106 |  |

Passed

# 2.

Use the programming language C to solve this exercise.

Write a program which from standard input receives an integer n (number of Circles), where 0<n<100. In the next n lines the program receives 3 numbers x, y and radius, where x and y represent the centre of the circle, and radius is the radius of the circle. The program should print out the area and centre of the largest circle (the circle with the largest area).

In an edge case where there are more circles with the same largest area only print out the first one.

The structure Point must have the following information:

-x coordinate (int)

-y coordinate(int)

The structure Circle must have the following information:

-Centre (Point)

-Radius(int)

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 3  1 2 3  4 5 6  3 2 3 | Largest circle has area of: 113.04 with centre: (4,5) | Largest circle has area of: 113.04 with centre: (4,5) |  |
|  | 10  1 2 3  4 5 3  6 7 3  8 9 3  2 1 3  5 4 3  7 6 3  9 8 3  10 10 3  11 11 3 | Largest circle has area of: 28.26 with centre: (1,2) | Largest circle has area of: 28.26 with centre: (1,2) |  |
|  | 6  1 1 1  2 3 10  9 8 7  33 23 10  11 11 1  9 9 9 | Largest circle has area of: 314.00 with centre: (2,3) | Largest circle has area of: 314.00 with centre: (2,3) |  |
|  | 2  1 2 6  4 3 3 | Largest circle has area of: 113.04 with centre: (1,2) | Largest circle has area of: 113.04 with centre: (1,2) |  |
|  | 7  1 2 3  1 2 6  1 2 10  1 2 4  3 3 10  3 2 2  88 190 10 | Largest circle has area of: 314.00 with centre: (1,2) | Largest circle has area of: 314.00 with centre: (1,2) |  |

Passed

# 3.

Define a structure **Employee**with fields: name (char array), identification number (ID) of the employee (char array), name of the department in which he works (char array) and salary (integer).

Then, define a structure **Company**with fields: name of the company (char array), number of employees (integer) and an array of employees.

Define a function:

**void printEmployeesBelowAverageSalary(Company \*comp, int numCompanies)**

which for each company will print all employees whose salary is lower than the average salary of all employees in that company.

Define a function:

**void printHighestSalaryEmployee(Company \*comp, int numCompanies, char \*department)**

which for each company will print the employee with the highest salary in the given department.

**Note**: First, the number of companies is read, then the companies (name and number of employees), then their employee data respectively, and finally the department that will be used in the **printHighestSalaryEmployee** function is read.

|  | **Input** | **Expected** | **Got** |
| --- | --- | --- | --- |
|  | 2  Apple  3  Vlatko 111111 IT 150000  Nikola 222222 IT 125000  Marko 333333 Accounting 50000  Google  2  Stefan 444444 IT 150000  John 555555 HR 60000  IT | Employees with salaries below the average:  Marko 333333 Accounting 50000  John 555555 HR 60000  Employees with highest salaries for given department:  Vlatko 111111 IT 150000  Stefan 444444 IT 150000 | Employees with salaries below the average:  Marko 333333 Accounting 50000  John 555555 HR 60000  Employees with highest salaries for given department:  Vlatko 111111 IT 150000  Stefan 444444 IT 150000 |  |
|  | 3  Facebook  3  Milan 111111 IT 120000  Jack 222222 HR 50000  Nick 333333 HR 70000  IBM  2  Phil 444444 Finance 45000  Hugh 555555 IT 135000  Apple  4  Steve 666666 HR 50000  Karen 777777 HR 120000  Anna 888888 IT 180000  Lucy 999999 Accounting 60000  HR | Employees with salaries below the average:  Jack 222222 HR 50000  Nick 333333 HR 70000  Phil 444444 Finance 45000  Steve 666666 HR 50000  Lucy 999999 Accounting 60000  Employees with highest salaries for given department:  Nick 333333 HR 70000  Nick 333333 HR 70000  Karen 777777 HR 120000 | Employees with salaries below the average:  Jack 222222 HR 50000  Nick 333333 HR 70000  Phil 444444 Finance 45000  Steve 666666 HR 50000  Lucy 999999 Accounting 60000  Employees with highest salaries for given department:  Nick 333333 HR 70000  Jack 222222 HR 50000  Karen 777777 HR 120000 |  |
|  | 3  Amazon  4  Petar 111111 IT 150000  Kiril 222222 IT 160000  Stephen 125345 Accounting 60000  Jim 543164 Accounting 80000  Microsoft  3  Dave 333333 Accounting 70000  Seth 444444 Accounting 75000  Dan 555555 IT 130000  Google  3  Steve 666666 Accounting 120000  Mark 777777 Accounting 125000  Connor 646346 IT 175000  Accounting | Employees with salaries below the average:  Stephen 125345 Accounting 60000  Jim 543164 Accounting 80000  Dave 333333 Accounting 70000  Seth 444444 Accounting 75000  Steve 666666 Accounting 120000  Mark 777777 Accounting 125000  Employees with highest salaries for given department:  Jim 543164 Accounting 80000  Seth 444444 Accounting 75000  Mark 777777 Accounting 125000 | Employees with salaries below the average:  Stephen 125345 Accounting 60000  Jim 543164 Accounting 80000  Dave 333333 Accounting 70000  Seth 444444 Accounting 75000  Steve 666666 Accounting 120000  Mark 777777 Accounting 125000  Employees with highest salaries for given department:  Jim 543164 Accounting 80000  Seth 444444 Accounting 75000  Mark 777777 Accounting 125000 |  |

LAB 2

# 1.

Create a class BasketballPlayer which has the following information:

* name (array of characters not longer than 20 characters)
* surname (array of characters not longer than 20 characters)
* jersey number (integer)
* points on first match (integer)
* points on second match (integer)
* points on third match (integer)

Write the needed methods for the class, and write a method for printing details about the basketball player in the following format "Player: **name surname** with number: **number** has **avg\_pts** points on average".

The program should receive the name, surname, jersey number and points from 3 matches on standard input.

On standard output just print out the details about the basketball player using the created method.

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | Marko  Petrov  10  13  14  15 | Player: Marko Petrov with number: 10 has 14 points on average | Player: Marko Petrov with number: 10 has 14 points on average |  |
|  | Ema  Pandilova  4  17  16  8 | Player: Ema Pandilova with number: 4 has 13.6667 points on average | Player: Ema Pandilova with number: 4 has 13.6667 points on average |  |
|  | Milan  Todorovikj  6  20  15  11 | Player: Milan Todorovikj with number: 6 has 15.3333 points on average | Player: Milan Todorovikj with number: 6 has 15.3333 points on average |  |
|  | Dobrinka  Sotirova  15  17  13  8 | Player: Dobrinka Sotirova with number: 15 has 12.6667 points on average | Player: Dobrinka Sotirova with number: 15 has 12.6667 points on average |  |
|  | Nikola  Nikolovski  11  20  19  15 | Player: Nikola Nikolovski with number: 11 has 18 points on average | Player: Nikola Nikolovski with number: 11 has 18 points on average |  |

Passed

# 2.

Define a class named **Film**that will store the following information:

* **name**(an array of up to 100 characters) - the name of the film
* **rating**(a decimal number) - the film's rating
* **revenue**(a decimal number) - how much the film earned in millions of dollars

All variables inside the class should be private. The following should be defined within the class:

* **constructor (with and without parameters)**
* **destructor**
* **set methods**
* **get methods**

The following methods should be defined in the class:

* **display()**- which will print the account information in the following format:

Name of film: …

Rating: …

Revenue: …

This method should be defined outside the class:

* **printMostPopularFilm(Film films, int n)** - which will find the film with the highest revenue.
* **DO NOT CHANGE THE MAIN FUNCTION**

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 10  Inception 8.8 828.3  Titanic 7.8 2187.5  Avatar 7.8 2787.9  Jaws 8.0 470.7  Rocky 8.1 225.0  Grease 7.2 394.5  Alien 8.4 1043.4  Jumanji 7.0 962.1  Gladiator 8.5 460.5  Memento 8.4 39.7 | -->Testing set methods and display()  Name of film: Memento  Rating: 8.4  Revenue: 39.7  -->Testing printMostPopularFilm()  Name of film: Avatar  Rating: 7.8  Revenue: 2787.9 | -->Testing set methods and display()  Name of film: Memento  Rating: 8.4  Revenue: 39.7  -->Testing printMostPopularFilm()  Name of film: Avatar  Rating: 7.8  Revenue: 2787.9 |  |
|  | 10  Scarface 8.3 65.9  Babe 6.8 254.1  Shrek 7.8 484.4  Nemo 8.1 940.3  Superman 7.3 300.2  Jurassic 8.1 1671.7  Matrix 8.7 463.5  Spiderman 7.3 890.9  Frozen 7.4 1276.5  Moana 7.6 690.1 | -->Testing set methods and display()  Name of film: Moana  Rating: 7.6  Revenue: 690.1  -->Testing printMostPopularFilm()  Name of film: Jurassic  Rating: 8.1  Revenue: 1671.7 | -->Testing set methods and display()  Name of film: Moana  Rating: 7.6  Revenue: 690.1  -->Testing printMostPopularFilm()  Name of film: Jurassic  Rating: 8.1  Revenue: 1671.7 |  |
|  | 10  Tangled 7.7 591.8  Zootopia 8.0 1023.8  Coco 8.4 807.8  Ratatouille 8.0 623.7  Up 8.2 735.1  Cars 7.1 462.2  Toy 8.3 1066.9  Pirates 8.0 794.9  Scream 7.2 173.0  Halloween 7.8 255.5 | -->Testing set methods and display()  Name of film: Halloween  Rating: 7.8  Revenue: 255.5  -->Testing printMostPopularFilm()  Name of film: Toy  Rating: 8.3  Revenue: 1066.9 | -->Testing set methods and display()  Name of film: Halloween  Rating: 7.8  Revenue: 255.5  -->Testing printMostPopularFilm()  Name of film: Toy  Rating: 8.3  Revenue: 1066.9 |  |

Passed

# 3.

Define a class called **Country**which will store the following information:

* **name**(a string of maximum 100 characters) representing the name of the country
* **capital**(a string of maximum 100 characters) representing the capital city
* **area**(a decimal number) representing the area in thousands of square kilometers
* **population**(a decimal number) representing the population in millions.

All variables inside the class should be private. The following should be defined within the class:

* **constructors (with and without parameters)**
* **destructor**
* **set methods**
* **get methods**

The following methods should be defined within the class:

* **display()**, which will print the information about the country in the following format:

Country: Guyana

Capital: Georgetown

Area: 214

Population: 0.8

Outside the class:

* **sortCountries(Country \*countries, int n)**, which will sort the countries in ascending order by their area.

**DO NOT MODIFY THE MAIN FUNCTION.**

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 5  Brazil Brasília 8515 213.3  Chile Santiago 756 19.1  Cyprus Nicosia 9 1.2  Gambia Banjul 11 2.3  Greece Athens 131957 10.7 | -->Testing set methods and display()  Country: Greece  Capital: Athens  Area: 131957  Population: 10.7  -->Testing sortCountries()  Country: Cyprus  Capital: Nicosia  Area: 9  Population: 1.2  Country: Gambia  Capital: Banjul  Area: 11  Population: 2.3  Country: Chile  Capital: Santiago  Area: 756  Population: 19.1  Country: Brazil  Capital: Brasília  Area: 8515  Population: 213.3  Country: Greece  Capital: Athens  Area: 131957  Population: 10.7 | -->Testing set methods and display()  Country: Greece  Capital: Athens  Area: 131957  Population: 10.7  -->Testing sortCountries()  Country: Cyprus  Capital: Nicosia  Area: 9  Population: 1.2  Country: Gambia  Capital: Banjul  Area: 11  Population: 2.3  Country: Chile  Capital: Santiago  Area: 756  Population: 19.1  Country: Brazil  Capital: Brasília  Area: 8515  Population: 213.3  Country: Greece  Capital: Athens  Area: 131957  Population: 10.7 |  |
|  | 10  Guinea Conakry 245 13.2  Guyana Georgetown 214 0.8  Haiti Port-au-Prince 27 11.4  Israel Jerusalem 22 9.4  Jordan Amman 89 10.1  Laos Vientiane 236 7.2  Lebanon Beirut 10 6.9  Maldives Male 30 0.5  Mali Bamako 1240 19.1  Malta Valletta 0.32 0.5 | -->Testing set methods and display()  Country: Malta  Capital: Valletta  Area: 0.32  Population: 0.5  -->Testing sortCountries()  Country: Malta  Capital: Valletta  Area: 0.32  Population: 0.5  Country: Lebanon  Capital: Beirut  Area: 10  Population: 6.9  Country: Israel  Capital: Jerusalem  Area: 22  Population: 9.4  Country: Haiti  Capital: Port-au-Prince  Area: 27  Population: 11.4  Country: Maldives  Capital: Male  Area: 30  Population: 0.5  Country: Jordan  Capital: Amman  Area: 89  Population: 10.1  Country: Guyana  Capital: Georgetown  Area: 214  Population: 0.8  Country: Laos  Capital: Vientiane  Area: 236  Population: 7.2  Country: Guinea  Capital: Conakry  Area: 245  Population: 13.2  Country: Mali  Capital: Bamako  Area: 1240  Population: 19.1 | -->Testing set methods and display()  Country: Malta  Capital: Valletta  Area: 0.32  Population: 0.5  -->Testing sortCountries()  Country: Malta  Capital: Valletta  Area: 0.32  Population: 0.5  Country: Lebanon  Capital: Beirut  Area: 10  Population: 6.9  Country: Israel  Capital: Jerusalem  Area: 22  Population: 9.4  Country: Haiti  Capital: Port-au-Prince  Area: 27  Population: 11.4  Country: Maldives  Capital: Male  Area: 30  Population: 0.5  Country: Jordan  Capital: Amman  Area: 89  Population: 10.1  Country: Guyana  Capital: Georgetown  Area: 214  Population: 0.8  Country: Laos  Capital: Vientiane  Area: 236  Population: 7.2  Country: Guinea  Capital: Conakry  Area: 245  Population: 13.2  Country: Mali  Capital: Bamako  Area: 1240  Population: 19.1 |  |
|  | 12  Monaco Monaco 0.2 0.04  Namibia Windhoek 825 2.5  Niger Niamey 1270 25.2  Qatar Doha 11 2.8  Romania Bucharest 238391 19.5  Rwanda Kigali 26 12.7  Samoa Apia 2 0.2  Somalia Mogadishu 637 15.9  Tonga Nuku'alofa 0.75 0.1  Turkey Ankara 783562 83.6  Yemen Sana'a 527 29.2  Zambia Lusaka 752 18.9 | -->Testing set methods and display()  Country: Zambia  Capital: Lusaka  Area: 752  Population: 18.9  -->Testing sortCountries()  Country: Monaco  Capital: Monaco  Area: 0.2  Population: 0.04  Country: Tonga  Capital: Nuku'alofa  Area: 0.75  Population: 0.1  Country: Samoa  Capital: Apia  Area: 2  Population: 0.2  Country: Qatar  Capital: Doha  Area: 11  Population: 2.8  Country: Rwanda  Capital: Kigali  Area: 26  Population: 12.7  Country: Yemen  Capital: Sana'a  Area: 527  Population: 29.2  Country: Somalia  Capital: Mogadishu  Area: 637  Population: 15.9  Country: Zambia  Capital: Lusaka  Area: 752  Population: 18.9  Country: Namibia  Capital: Windhoek  Area: 825  Population: 2.5  Country: Niger  Capital: Niamey  Area: 1270  Population: 25.2  Country: Romania  Capital: Bucharest  Area: 238391  Population: 19.5  Country: Turkey  Capital: Ankara  Area: 783562  Population: 83.6 | -->Testing set methods and display()  Country: Zambia  Capital: Lusaka  Area: 752  Population: 18.9  -->Testing sortCountries()  Country: Monaco  Capital: Monaco  Area: 0.2  Population: 0.04  Country: Tonga  Capital: Nuku'alofa  Area: 0.75  Population: 0.1  Country: Samoa  Capital: Apia  Area: 2  Population: 0.2  Country: Qatar  Capital: Doha  Area: 11  Population: 2.8  Country: Rwanda  Capital: Kigali  Area: 26  Population: 12.7  Country: Yemen  Capital: Sana'a  Area: 527  Population: 29.2  Country: Somalia  Capital: Mogadishu  Area: 637  Population: 15.9  Country: Zambia  Capital: Lusaka  Area: 752  Population: 18.9  Country: Namibia  Capital: Windhoek  Area: 825  Population: 2.5  Country: Niger  Capital: Niamey  Area: 1270  Population: 25.2  Country: Romania  Capital: Bucharest  Area: 238391  Population: 19.5  Country: Turkey  Capital: Ankara  Area: 783562  Population: 83.6 |  |

Passed

LAB 3

# 1.

Define a class **MobilePhone**in which there will be information about a mobile phone:

* model (array of characters not longer than 20 characters)
* model number (integer)
* release year (integer)

Define the default constructor, constructor with arguments, copy constructor, destructor and get methods if needed. Additionally define a print function which will print the information about the mobile phone in this format: "**(model) (model number)** release year: **(release year)**".

Than define a class **Owner**in which there will be information for:

* name (array of characters not longer than 20 characters)
* surname (array of characters not longer than 20 characters)
* mobile phone (object of class **MobilePhone**)

Define the default constructor, constructor with arguments, destructor and get methods if needed. Additionally define a print function which will print the information about the owner in this format:

„**(name)** **(surname)**has a mobile phone:

**(information about the mobile phone)"**

**DO NOT CHANGE THE MAIN FUNCTION!!!**

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 1  Iphone  12  2020  Marko  Petrov | Marko Petrov has a mobile phone:  Iphone 12 release year: 2020 | Marko Petrov has a mobile phone:  Iphone 12 release year: 2020 |  |
|  | 2  Xiamoi  12  2021  Dobrinka  Sotirova | Dobrinka Sotirova has a mobile phone:  Xiamoi 12 release year: 2021 | Dobrinka Sotirova has a mobile phone:  Xiamoi 12 release year: 2021 |  |
|  | 2  Iphone  11  2019  Milan  Todorovikj | Milan Todorovikj has a mobile phone:  Iphone 11 release year: 2019 | Milan Todorovikj has a mobile phone:  Iphone 11 release year: 2019 |  |
|  | 1  Pixel  7  2022  Ema  Pandilova | Ema Pandilova has a mobile phone:  Pixel 7 release year: 2022 | Ema Pandilova has a mobile phone:  Pixel 7 release year: 2022 |  |
|  | 2  Galaxy  23  2023  Bojan  Bojanovski | Bojan Bojanovski has a mobile phone:  Galaxy 23 release year: 2023 | Bojan Bojanovski has a mobile phone:  Galaxy 23 release year: 2023 |  |

Passed

# 2.

Define a class **SpaceObject**with attributes:

* **name**(array of maximum 100 characters) - name of space object
* **type**(array of maximum 100 characters) - type of space object
* **distanceFromEarth**(integer) - distance from Earth

Functions that should be implemented in the class:

* **print()**

- Prints in format:  
"[**name**] ( [**type**] ) - distance: [**distanceFromEarth**] light years away from Earth".

Define a class **Alien**with attributes:

* **name**(array of maximum 100 characters) - name
* **age**(integer) - age
* **homePlanet**(array of maximum 100 characters) - home planet of the alien
* **numObj**(integer) - number of favourite space objects
* **spaceObjects**(array of SpaceObject objects) - favourite space objects

Functions that should be implemented in the class:

* **SpaceObject getObjectClosestToEarth()**

- Returns the object which is one of their favorites and has the smallest distance from Earth.

* **print()**

- Prints in format:

Alien name: ...

Alien age: ...

Alien homePlanet: ...

Favourite space object closest to earth: ....

All the variables inside the class must be **private**. Accordingly, in the classes you will need to define:

* **constructor (with and without parameters)**
* **destructor**
* **get/set methods**
* **copy constructor**

**DO NOT CHANGE THE main() FUNCTION.**

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | Zorblax 537 Xarkon 3  Gamma3 SpaceStation 9864000  Epsilon1 CargoVessel 1989200  Vulcanus Starship 20900000 | Alien name: Zorblax  Alien age: 537  Alien home planet: Xarkon  Favourite space object closest to earth: Epsilon1 (CargoVessel) - distance: 1989200 light years away from Earth | Alien name: Zorblax  Alien age: 537  Alien home planet: Xarkon  Favourite space object closest to earth: Epsilon1 (CargoVessel) - distance: 1989200 light years away from Earth |  |
|  | Zargoth 257 Lululand 4  Andromeda1 SpaceStation 2537000  Nebula2 Probe 2020000  Comet12 Comet 367900000  Orion6 Satellite 1350000000 | Alien name: Zargoth  Alien age: 257  Alien home planet: Lululand  Favourite space object closest to earth: Nebula2 (Probe) - distance: 2020000 light years away from Earth | Alien name: Zargoth  Alien age: 257  Alien home planet: Lululand  Favourite space object closest to earth: Nebula2 (Probe) - distance: 2020000 light years away from Earth |  |
|  | Krynn 3233 Zolara 4  AndromedaX2 Starship 7900000  Quasar1 Satellite 1850000  Nebula12 Probe 10200000  Nexus3 SpaceStation 8000000 | Alien name: Krynn  Alien age: 3233  Alien home planet: Zolara  Favourite space object closest to earth: Quasar1 (Satellite) - distance: 1850000 light years away from Earth | Alien name: Krynn  Alien age: 3233  Alien home planet: Zolara  Favourite space object closest to earth: Quasar1 (Satellite) - distance: 1850000 light years away from Earth |  |

Passed

# 3.

Define a class **Employee**which will store information about:

**name**(array of chars maximum 100 characters) - the employee's name

**surname**(array of chars maximum 100 characters) - the employee's surname

**salary**(integer) - how much the employee earns

The methods that the class should have are:

* **print()**

- Prints in the format:  
Employee name:  
Employee surname:  
Employee salary:

Define a class **TechCompany**which will store information about:

**name**(string of maximum 100 characters) - the name of the company

**employees**(array of Employee objects)

**numOfEmployees**(integer) - how many employees the company has

The methods that the class should have are:

* **print()**

-Prints in the format: Tech company name: Number of employees:

* **double getAverageOfEmployeeSalary()**

-Returns the average value of salaries for each employee in the company.

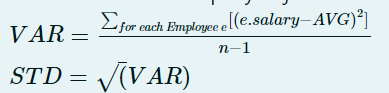
All variables inside the classes should be private. Accordingly, constructors (with and without parameters), destructors, set methods, get methods, and copy constructors should be defined within the classes.

Functions outside of the class:

* **printCompanyWithHighestAverageSalary(TechCompany companies[], int numCompanies)**

-Returns the TechCompany object that has the highest average salary for its employees.

* **printCompanyWithHighestStdSalary(TechCompany companies[], int numCompanies)**

-Returns the TechCompany object that has the highest standard deviation for its employees' salaries.  
  
-Note: AVG is the average value of salaries for all employees in the company, STD is the standard deviation. First square the expression, then add it to the sum!

* **isSameCompany(TechCompany company1, TechCompany company2)**

-Returns a boolean value: two companies are the same if they have the same name. Only the names need to be checked!

DO NOT CHANGE THE MAIN FUNCTION.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 3  CompanyA  5  John Doe 50000  Jane Doe 70000  Bob Smith 60000  Anna Williams 80000  Michael Brown 55000  CompanyB  4  Alice Brown 80000  David Green 70000  Eve White 50000  Frank Black 65000  CompanyC  6  Laura Lee 100000  Benjamin Johnson 65000  Emma Davis 75000  Oliver Wilson 82000  Sophie Martinez 60000  William Clark 72000 | -->Testing get and set methods for one object and copy constructor  copy  -->Testing addEmployee function  Number of employees in copy: 6  -->Testing copy of first employee  Employee name: John  Employee surname: Doe  Employee salary: 50000  -->Testing methods  Tech company with the highest average salary: CompanyC  Tech company with the highest standard deviation for salaries: CompanyC  The tech company: CompanyC has the highest standard deviation and highest average salary |

LAB 4

# 1.

As a successful student at FCSE, you have been called to make improvements to the IKnow system. As a part of the improvements there should be a new component that deals with the disciplinary actions, so you, a member of the team, have to implement the class **DisciplinaryAction**which will contain the following information:

* Name of the student (dynamic allocation of a char array)
* Index of the student (int)
* Reason for the disciplinary action (dynamic allocation of a char array)

The class should also contain the following methods, operator and constructors:

* **Default**constructor
* **Copy**constructor
* **Constructor**with **arguments**
* **Destructor**
* **Set**и **Get**method for the index
* void **print()**method
* **Operator =**

The Print() method should print according to the following template:  
Student: **[Name]**  
Student's index:**[Index]**  
Reason: **[Reason]**

Change the values of **[]** with the values stored in the class

**Do NOT make changes in the main() function !!!**

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 1  Stefan 151020 Prepis | -- Testing operator = & print() --  Student: Stefan  Student's index: 151020  Reason: Prepis  -- Testing copy constructor & set index --  -------------  Source:  -------------  Student: Stefan  Student's index: 151020  Reason: Prepis  -------------  Copied and edited:  -------------  Student: Stefan  Student's index: 112233  Reason: Prepis  -- Testing if array was inputted correctly --  Student: Stefan  Student's index: 151020  Reason: Prepis | -- Testing operator = & print() --  Student: Stefan  Student's index: 151020  Reason: Prepis  -- Testing copy constructor & set index --  -------------  Source:  -------------  Student: Stefan  Student's index: 151020  Reason: Prepis  -------------  Copied and edited:  -------------  Student: Stefan  Student's index: 112233  Reason: Prepis  -- Testing if array was inputted correctly --  Student: Stefan  Student's index: 151020  Reason: Prepis |  |
|  | 3  Nikola 216051 Prepis  Darijan 216055 Osteta\_na\_oprema  Ime 123456 Nedolicno\_odnesuvanje | -- Testing operator = & print() --  Student: Nikola  Student's index: 216051  Reason: Prepis  -- Testing copy constructor & set index --  -------------  Source:  -------------  Student: Nikola  Student's index: 216051  Reason: Prepis  -------------  Copied and edited:  -------------  Student: Nikola  Student's index: 112233  Reason: Prepis  -- Testing if array was inputted correctly --  Student: Nikola  Student's index: 216051  Reason: Prepis  Student: Darijan  Student's index: 216055  Reason: Osteta\_na\_oprema  Student: Ime  Student's index: 123456  Reason: Nedolicno\_odnesuvanje | -- Testing operator = & print() --  Student: Nikola  Student's index: 216051  Reason: Prepis  -- Testing copy constructor & set index --  -------------  Source:  -------------  Student: Nikola  Student's index: 216051  Reason: Prepis  -------------  Copied and edited:  -------------  Student: Nikola  Student's index: 112233  Reason: Prepis  -- Testing if array was inputted correctly --  Student: Nikola  Student's index: 216051  Reason: Prepis  Student: Darijan  Student's index: 216055  Reason: Osteta\_na\_oprema  Student: Ime  Student's index: 123456  Reason: Nedolicno\_odnesuvanje |  |
|  | 5  Damjan 123042 Prepis  Iskra 227700 Prepis  Jakov 217070 Nedolicno\_odnesuvanje  Vladislav 232323 Prepis  Jovan 210000 Nedolicno\_odnesuvanje | -- Testing operator = & print() --  Student: Damjan  Student's index: 123042  Reason: Prepis  -- Testing copy constructor & set index --  -------------  Source:  -------------  Student: Damjan  Student's index: 123042  Reason: Prepis  -------------  Copied and edited:  -------------  Student: Damjan  Student's index: 112233  Reason: Prepis  -- Testing if array was inputted correctly --  Student: Damjan  Student's index: 123042  Reason: Prepis  Student: Iskra  Student's index: 227700  Reason: Prepis  Student: Jakov  Student's index: 217070  Reason: Nedolicno\_odnesuvanje  Student: Vladislav  Student's index: 232323  Reason: Prepis  Student: Jovan  Student's index: 210000  Reason: Nedolicno\_odnesuvanje | -- Testing operator = & print() --  Student: Damjan  Student's index: 123042  Reason: Prepis  -- Testing copy constructor & set index --  -------------  Source:  -------------  Student: Damjan  Student's index: 123042  Reason: Prepis  -------------  Copied and edited:  -------------  Student: Damjan  Student's index: 112233  Reason: Prepis  -- Testing if array was inputted correctly --  Student: Damjan  Student's index: 123042  Reason: Prepis  Student: Iskra  Student's index: 227700  Reason: Prepis  Student: Jakov  Student's index: 217070  Reason: Nedolicno\_odnesuvanje  Student: Vladislav  Student's index: 232323  Reason: Prepis  Student: Jovan  Student's index: 210000  Reason: Nedolicno\_odnesuvanje |  |

Passed

# 2.

Define a class called Student which stores:

* **name**(a character array, dynamically allocated) - their name
* **age**(an integer) - their age
* **major**(a character array, dynamically allocated) - their major

Define a default constructor, constructor with arguments, and copy constructor for the class.

Then define a class called Classroom which stores:

* **students**(a dynamically allocated array of Students)
* **numStudents**(integer)
* **capacity**(integer)

Define a default constructor and a constructor with arguments for the class.

The following methods need to be defined for the Classroom class:

* **add**- to add a new student
* **remove** - to remove a student with a given name
* **findMedianAge**- finds the median age of the students in the given Classroom.

Example: 2 4 5**6** 7 8 10

Median: 6

Example: 2 4 5 **7 8** 10 12 20

Median: (7 + 8)/2

The ages must be in ascending order!

Get and set methods can be created as needed.

**All students have a unique name. Do not modify the main function.**

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 2  3  Alice 21 Math  Bob 22 English  Charlie 20 History  2  Dave 23 Math  Eve 20 English  1  2  Dave  Eve  1 | Classroom 0  Alice (21, Math)  Bob (22, English)  Charlie (20, History)  Classroom 1  Dave (23, Math)  Eve (20, English)  The median age in classroom 1 is: 21.5  After removing the elements:  Classroom 0  Alice (21, Math)  Bob (22, English)  Charlie (20, History)  Classroom 1 | Classroom 0  Alice (21, Math)  Bob (22, English)  Charlie (20, History)  Classroom 1  Dave (23, Math)  Eve (20, English)  The median age in classroom 1 is: 21.5  After removing the elements:  Classroom 0  Alice (21, Math)  Bob (22, English)  Charlie (20, History)  Classroom 1 |  |
|  | 2  2  Alice 19 Biology  Bob 20 Physics  3  Charlie 22 Chemistry  Dave 21 Math  Eve 20 English  1  3  Bob  Charlie  Dave | Classroom 0  Alice (19, Biology)  Bob (20, Physics)  Classroom 1  Charlie (22, Chemistry)  Dave (21, Math)  Eve (20, English)  The median age in classroom 1 is: 21  After removing the elements:  Classroom 0  Alice (19, Biology)  Classroom 1  Eve (20, English) | Classroom 0  Alice (19, Biology)  Bob (20, Physics)  Classroom 1  Charlie (22, Chemistry)  Dave (21, Math)  Eve (20, English)  The median age in classroom 1 is: 21  After removing the elements:  Classroom 0  Alice (19, Biology)  Classroom 1  Eve (20, English) |  |
|  | 9  15  Alice 18 Math  Bob 19 English  Charlie 21 History  Dave 22 Biology  Eve 25 Physics  Frank 26 Chemistry  George 27 Economics  Hannah 30 Psychology  Isaac 32 Sociology  Jack 33 Anthropology  Katie 34 Geology  Liam 35 Philosophy  Molly 36 Linguistics  Nathan 38 Literature  Olivia 40 Art  15  Peter 18 Math  Quinn 20 English  Rachel 21 History  Sam 23 Biology  Tom 25 Physics  Ursula 26 Chemistry  Victor 28 Economics  Wendy 30 Psychology  Xander 31 Sociology  Yara 33 Anthropology  Zoe 34 Geology  Andrew 36 Philosophy  Bella 38 Linguistics  Caroline 39 Literature  David 40 Art  10  Maggie 20 Math  Nora 21 English  Oscar 23 History  Penny 24 Biology  Quentin 27 Physics  Rose 28 Chemistry  Samantha 30 Economics  Trevor 31 Psychology  Uma 33 Sociology  Violet 35 Anthropology  5  Wesley 18 Math  Xavier 19 English  Yasmine 20 History  Zack 21 Biology  Abby 22 Physics  10  Ben 18 Math  Cathy 19 English  Doug 20 History  Ellie 22 Biology  Fred 23 Physics  Grace 25 Chemistry  Henry 27 Economics  Ivy 28 Psychology  Joey 29 Sociology  Kendra 31 Anthropology  5  Lenny 18 Math  Mandy 19 English  Nate 20 History  Oliver 21 Biology  Pam 23 Physics  10  Quincy 19 Math  Rachel 20 English  Stephen 22 History  Tina 24 Biology  Ulysses 25 Physics  Victoria 26 Chemistry  Wally 28 Economics  Xandra 30 Psychology  Yasmin 31 Sociology  Zane 33 Anthropology  8  Zelda 18 Math  Adam 19 English  Brian 21 History  Chloe 23 Biology  Dawn 25 Physics  Eric 26 Chemistry  Faith 28 Economics  Gus 30 Psychology  4  Heather 18 Math  Ian 19 English  Jane 20 History  Kelly 22 Biology  8  6  Alice  Heather  Brian  Yasmin  Nathan  Bob | Classroom 0  Alice (18, Math)  Bob (19, English)  Charlie (21, History)  Dave (22, Biology)  Eve (25, Physics)  Frank (26, Chemistry)  George (27, Economics)  Hannah (30, Psychology)  Isaac (32, Sociology)  Jack (33, Anthropology)  Katie (34, Geology)  Liam (35, Philosophy)  Molly (36, Linguistics)  Nathan (38, Literature)  Olivia (40, Art)  Classroom 1  Peter (18, Math)  Quinn (20, English)  Rachel (21, History)  Sam (23, Biology)  Tom (25, Physics)  Ursula (26, Chemistry)  Victor (28, Economics)  Wendy (30, Psychology)  Xander (31, Sociology)  Yara (33, Anthropology)  Zoe (34, Geology)  Andrew (36, Philosophy)  Bella (38, Linguistics)  Caroline (39, Literature)  David (40, Art)  Classroom 2  Maggie (20, Math)  Nora (21, English)  Oscar (23, History)  Penny (24, Biology)  Quentin (27, Physics)  Rose (28, Chemistry)  Samantha (30, Economics)  Trevor (31, Psychology)  Uma (33, Sociology)  Violet (35, Anthropology)  Classroom 3  Wesley (18, Math)  Xavier (19, English)  Yasmine (20, History)  Zack (21, Biology)  Abby (22, Physics)  Classroom 4  Ben (18, Math)  Cathy (19, English)  Doug (20, History)  Ellie (22, Biology)  Fred (23, Physics)  Grace (25, Chemistry)  Henry (27, Economics)  Ivy (28, Psychology)  Joey (29, Sociology)  Kendra (31, Anthropology)  Classroom 5  Lenny (18, Math)  Mandy (19, English)  Nate (20, History)  Oliver (21, Biology)  Pam (23, Physics)  Classroom 6  Quincy (19, Math)  Rachel (20, English)  Stephen (22, History)  Tina (24, Biology)  Ulysses (25, Physics)  Victoria (26, Chemistry)  Wally (28, Economics)  Xandra (30, Psychology)  Yasmin (31, Sociology)  Zane (33, Anthropology)  Classroom 7  Zelda (18, Math)  Adam (19, English)  Brian (21, History)  Chloe (23, Biology)  Dawn (25, Physics)  Eric (26, Chemistry)  Faith (28, Economics)  Gus (30, Psychology)  Classroom 8  Heather (18, Math)  Ian (19, English)  Jane (20, History)  Kelly (22, Biology)  The median age in classroom 8 is: 19.5  After removing the elements:  Classroom 0  Charlie (21, History)  Dave (22, Biology)  Eve (25, Physics)  Frank (26, Chemistry)  George (27, Economics)  Hannah (30, Psychology)  Isaac (32, Sociology)  Jack (33, Anthropology)  Katie (34, Geology)  Liam (35, Philosophy)  Molly (36, Linguistics)  Olivia (40, Art)  Classroom 1  Peter (18, Math)  Quinn (20, English)  Rachel (21, History)  Sam (23, Biology)  Tom (25, Physics)  Ursula (26, Chemistry)  Victor (28, Economics)  Wendy (30, Psychology)  Xander (31, Sociology)  Yara (33, Anthropology)  Zoe (34, Geology)  Andrew (36, Philosophy)  Bella (38, Linguistics)  Caroline (39, Literature)  David (40, Art)  Classroom 2  Maggie (20, Math)  Nora (21, English)  Oscar (23, History)  Penny (24, Biology)  Quentin (27, Physics)  Rose (28, Chemistry)  Samantha (30, Economics)  Trevor (31, Psychology)  Uma (33, Sociology)  Violet (35, Anthropology)  Classroom 3  Wesley (18, Math)  Xavier (19, English)  Yasmine (20, History)  Zack (21, Biology)  Abby (22, Physics)  Classroom 4  Ben (18, Math)  Cathy (19, English)  Doug (20, History)  Ellie (22, Biology)  Fred (23, Physics)  Grace (25, Chemistry)  Henry (27, Economics)  Ivy (28, Psychology)  Joey (29, Sociology)  Kendra (31, Anthropology)  Classroom 5  Lenny (18, Math)  Mandy (19, English)  Nate (20, History)  Oliver (21, Biology)  Pam (23, Physics)  Classroom 6  Quincy (19, Math)  Rachel (20, English)  Stephen (22, History)  Tina (24, Biology)  Ulysses (25, Physics)  Victoria (26, Chemistry)  Wally (28, Economics)  Xandra (30, Psychology)  Zane (33, Anthropology)  Classroom 7  Zelda (18, Math)  Adam (19, English)  Chloe (23, Biology)  Dawn (25, Physics)  Eric (26, Chemistry)  Faith (28, Economics)  Gus (30, Psychology)  Classroom 8  Ian (19, English)  Jane (20, History)  Kelly (22, Biology) | Classroom 0  Alice (18, Math)  Bob (19, English)  Charlie (21, History)  Dave (22, Biology)  Eve (25, Physics)  Frank (26, Chemistry)  George (27, Economics)  Hannah (30, Psychology)  Isaac (32, Sociology)  Jack (33, Anthropology)  Katie (34, Geology)  Liam (35, Philosophy)  Molly (36, Linguistics)  Nathan (38, Literature)  Olivia (40, Art)  Classroom 1  Peter (18, Math)  Quinn (20, English)  Rachel (21, History)  Sam (23, Biology)  Tom (25, Physics)  Ursula (26, Chemistry)  Victor (28, Economics)  Wendy (30, Psychology)  Xander (31, Sociology)  Yara (33, Anthropology)  Zoe (34, Geology)  Andrew (36, Philosophy)  Bella (38, Linguistics)  Caroline (39, Literature)  David (40, Art)  Classroom 2  Maggie (20, Math)  Nora (21, English)  Oscar (23, History)  Penny (24, Biology)  Quentin (27, Physics)  Rose (28, Chemistry)  Samantha (30, Economics)  Trevor (31, Psychology)  Uma (33, Sociology)  Violet (35, Anthropology)  Classroom 3  Wesley (18, Math)  Xavier (19, English)  Yasmine (20, History)  Zack (21, Biology)  Abby (22, Physics)  Classroom 4  Ben (18, Math)  Cathy (19, English)  Doug (20, History)  Ellie (22, Biology)  Fred (23, Physics)  Grace (25, Chemistry)  Henry (27, Economics)  Ivy (28, Psychology)  Joey (29, Sociology)  Kendra (31, Anthropology)  Classroom 5  Lenny (18, Math)  Mandy (19, English)  Nate (20, History)  Oliver (21, Biology)  Pam (23, Physics)  Classroom 6  Quincy (19, Math)  Rachel (20, English)  Stephen (22, History)  Tina (24, Biology)  Ulysses (25, Physics)  Victoria (26, Chemistry)  Wally (28, Economics)  Xandra (30, Psychology)  Yasmin (31, Sociology)  Zane (33, Anthropology)  Classroom 7  Zelda (18, Math)  Adam (19, English)  Brian (21, History)  Chloe (23, Biology)  Dawn (25, Physics)  Eric (26, Chemistry)  Faith (28, Economics)  Gus (30, Psychology)  Classroom 8  Heather (18, Math)  Ian (19, English)  Jane (20, History)  Kelly (22, Biology)  The median age in classroom 8 is: 19.5  After removing the elements:  Classroom 0  Charlie (21, History)  Dave (22, Biology)  Eve (25, Physics)  Frank (26, Chemistry)  George (27, Economics)  Hannah (30, Psychology)  Isaac (32, Sociology)  Jack (33, Anthropology)  Katie (34, Geology)  Liam (35, Philosophy)  Molly (36, Linguistics)  Olivia (40, Art)  Classroom 1  Peter (18, Math)  Quinn (20, English)  Rachel (21, History)  Sam (23, Biology)  Tom (25, Physics)  Ursula (26, Chemistry)  Victor (28, Economics)  Wendy (30, Psychology)  Xander (31, Sociology)  Yara (33, Anthropology)  Zoe (34, Geology)  Andrew (36, Philosophy)  Bella (38, Linguistics)  Caroline (39, Literature)  David (40, Art)  Classroom 2  Maggie (20, Math)  Nora (21, English)  Oscar (23, History)  Penny (24, Biology)  Quentin (27, Physics)  Rose (28, Chemistry)  Samantha (30, Economics)  Trevor (31, Psychology)  Uma (33, Sociology)  Violet (35, Anthropology)  Classroom 3  Wesley (18, Math)  Xavier (19, English)  Yasmine (20, History)  Zack (21, Biology)  Abby (22, Physics)  Classroom 4  Ben (18, Math)  Cathy (19, English)  Doug (20, History)  Ellie (22, Biology)  Fred (23, Physics)  Grace (25, Chemistry)  Henry (27, Economics)  Ivy (28, Psychology)  Joey (29, Sociology)  Kendra (31, Anthropology)  Classroom 5  Lenny (18, Math)  Mandy (19, English)  Nate (20, History)  Oliver (21, Biology)  Pam (23, Physics)  Classroom 6  Quincy (19, Math)  Rachel (20, English)  Stephen (22, History)  Tina (24, Biology)  Ulysses (25, Physics)  Victoria (26, Chemistry)  Wally (28, Economics)  Xandra (30, Psychology)  Zane (33, Anthropology)  Classroom 7  Zelda (18, Math)  Adam (19, English)  Chloe (23, Biology)  Dawn (25, Physics)  Eric (26, Chemistry)  Faith (28, Economics)  Gus (30, Psychology)  Classroom 8  Ian (19, English)  Jane (20, History)  Kelly (22, Biology) |  |

Passed

LAB 5

# 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

LAB 6

# 1.

As a successful student at FCSE, you have been called to make improvements to the IKnow system. As a part of the improvements there should be a new component that deals with the disciplinary actions, so you, a member of the team, have to implement the class **DisciplinaryAction**which will contain the following information:

* Name of the student (dynamic allocation of a char array)
* Index of the student (int)
* Reason for the disciplinary action (dynamic allocation of a char array)
* number of sessions in which the student can't take exams (integer)

The class should also contain the following methods, operator and constructors:

* **Default**constructor
* **Copy**constructor
* **Constructor**with **arguments**
* **Destructor**
* **Set**и **Get**method for the index
* void **print()**method
* **Operator =**
* **operator <<**for printing the information about the disciplinary action
* **operator ++** for incrementation of the number of sessions in which the student can't take any exams
* **operator >=** for comparison of two disciplinary actions based on the number of sessions in which the student can't take any exam

The printing should be done according to the following template:  
Student: **[Name]**  
Student's index:**[Index]**  
Reason: **[Reason]**Sessions: **[Sessions]**

Change the values of **[]** with the values stored in the class

**Do NOT make changes in the main() function !!!**

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 1  Stefan 151020 Prepis 1 | -- Testing operator = & operator << --  Student: Stefan  Student's index: 151020  Reason: Prepis  Sessions: 1  -- Testing copy constructor & set index --  -------------  Source:  -------------  Student: Stefan  Student's index: 151020  Reason: Prepis  Sessions: 1  -------------  Copied and edited:  -------------  Student: Stefan  Student's index: 112233  Reason: Prepis  Sessions: 1  -- Testing if array was inputted correctly --  Student: Stefan  Student's index: 151020  Reason: Prepis  Sessions: 1  Testing operator ++ and <<  Student: Stefan  Student's index: 151020  Reason: Prepis  Sessions: 2  Testing operator >= | -- Testing operator = & operator << --  Student: Stefan  Student's index: 151020  Reason: Prepis  Sessions: 1  -- Testing copy constructor & set index --  -------------  Source:  -------------  Student: Stefan  Student's index: 151020  Reason: Prepis  Sessions: 1  -------------  Copied and edited:  -------------  Student: Stefan  Student's index: 112233  Reason: Prepis  Sessions: 1  -- Testing if array was inputted correctly --  Student: Stefan  Student's index: 151020  Reason: Prepis  Sessions: 1  Testing operator ++ and <<  Student: Stefan  Student's index: 151020  Reason: Prepis  Sessions: 2  Testing operator >= |  |
|  | 3  Nikola 216051 Prepis 1  Darijan 216055 Osteta\_na\_oprema 2  Ime 123456 Nedolicno\_odnesuvanje 3 | -- Testing operator = & operator << --  Student: Nikola  Student's index: 216051  Reason: Prepis  Sessions: 1  -- Testing copy constructor & set index --  -------------  Source:  -------------  Student: Nikola  Student's index: 216051  Reason: Prepis  Sessions: 1  -------------  Copied and edited:  -------------  Student: Nikola  Student's index: 112233  Reason: Prepis  Sessions: 1  -- Testing if array was inputted correctly --  Student: Nikola  Student's index: 216051  Reason: Prepis  Sessions: 1  Student: Darijan  Student's index: 216055  Reason: Osteta\_na\_oprema  Sessions: 2  Student: Ime  Student's index: 123456  Reason: Nedolicno\_odnesuvanje  Sessions: 3  Testing operator ++ and <<  Student: Nikola  Student's index: 216051  Reason: Prepis  Sessions: 2  Student: Darijan  Student's index: 216055  Reason: Osteta\_na\_oprema  Sessions: 3  Student: Ime  Student's index: 123456  Reason: Nedolicno\_odnesuvanje  Sessions: 4  Testing operator >=  0 < 1  0 < 2  1 >= 0  1 < 2  2 >= 0  2 >= 1 | -- Testing operator = & operator << --  Student: Nikola  Student's index: 216051  Reason: Prepis  Sessions: 1  -- Testing copy constructor & set index --  -------------  Source:  -------------  Student: Nikola  Student's index: 216051  Reason: Prepis  Sessions: 1  -------------  Copied and edited:  -------------  Student: Nikola  Student's index: 112233  Reason: Prepis  Sessions: 1  -- Testing if array was inputted correctly --  Student: Nikola  Student's index: 216051  Reason: Prepis  Sessions: 1  Student: Darijan  Student's index: 216055  Reason: Osteta\_na\_oprema  Sessions: 2  Student: Ime  Student's index: 123456  Reason: Nedolicno\_odnesuvanje  Sessions: 3  Testing operator ++ and <<  Student: Nikola  Student's index: 216051  Reason: Prepis  Sessions: 2  Student: Darijan  Student's index: 216055  Reason: Osteta\_na\_oprema  Sessions: 3  Student: Ime  Student's index: 123456  Reason: Nedolicno\_odnesuvanje  Sessions: 4  Testing operator >=  0 < 1  0 < 2  1 >= 0  1 < 2  2 >= 0  2 >= 1 |  |
|  | 5  Damjan 123042 Prepis 1  Iskra 227700 Prepis 2  Jakov 217070 Nedolicno\_odnesuvanje 4  Vladislav 232323 Prepis 2  Jovan 210000 Nedolicno\_odnesuvanje 3 | -- Testing operator = & operator << --  Student: Damjan  Student's index: 123042  Reason: Prepis  Sessions: 1  -- Testing copy constructor & set index --  -------------  Source:  -------------  Student: Damjan  Student's index: 123042  Reason: Prepis  Sessions: 1  -------------  Copied and edited:  -------------  Student: Damjan  Student's index: 112233  Reason: Prepis  Sessions: 1  -- Testing if array was inputted correctly --  Student: Damjan  Student's index: 123042  Reason: Prepis  Sessions: 1  Student: Iskra  Student's index: 227700  Reason: Prepis  Sessions: 2  Student: Jakov  Student's index: 217070  Reason: Nedolicno\_odnesuvanje  Sessions: 4  Student: Vladislav  Student's index: 232323  Reason: Prepis  Sessions: 2  Student: Jovan  Student's index: 210000  Reason: Nedolicno\_odnesuvanje  Sessions: 3  Testing operator ++ and <<  Student: Damjan  Student's index: 123042  Reason: Prepis  Sessions: 2  Student: Iskra  Student's index: 227700  Reason: Prepis  Sessions: 3  Student: Jakov  Student's index: 217070  Reason: Nedolicno\_odnesuvanje  Sessions: 5  Student: Vladislav  Student's index: 232323  Reason: Prepis  Sessions: 3  Student: Jovan  Student's index: 210000  Reason: Nedolicno\_odnesuvanje  Sessions: 4  Testing operator >=  0 < 1  0 < 2  0 < 3  0 < 4  1 >= 0  1 < 2  1 >= 3  1 < 4  2 >= 0  2 >= 1  2 >= 3  2 >= 4  3 >= 0  3 >= 1  3 < 2  3 < 4  4 >= 0  4 >= 1  4 < 2  4 >= 3 | -- Testing operator = & operator << --  Student: Damjan  Student's index: 123042  Reason: Prepis  Sessions: 1  -- Testing copy constructor & set index --  -------------  Source:  -------------  Student: Damjan  Student's index: 123042  Reason: Prepis  Sessions: 1  -------------  Copied and edited:  -------------  Student: Damjan  Student's index: 112233  Reason: Prepis  Sessions: 1  -- Testing if array was inputted correctly --  Student: Damjan  Student's index: 123042  Reason: Prepis  Sessions: 1  Student: Iskra  Student's index: 227700  Reason: Prepis  Sessions: 2  Student: Jakov  Student's index: 217070  Reason: Nedolicno\_odnesuvanje  Sessions: 4  Student: Vladislav  Student's index: 232323  Reason: Prepis  Sessions: 2  Student: Jovan  Student's index: 210000  Reason: Nedolicno\_odnesuvanje  Sessions: 3  Testing operator ++ and <<  Student: Damjan  Student's index: 123042  Reason: Prepis  Sessions: 2  Student: Iskra  Student's index: 227700  Reason: Prepis  Sessions: 3  Student: Jakov  Student's index: 217070  Reason: Nedolicno\_odnesuvanje  Sessions: 5  Student: Vladislav  Student's index: 232323  Reason: Prepis  Sessions: 3  Student: Jovan  Student's index: 210000  Reason: Nedolicno\_odnesuvanje  Sessions: 4  Testing operator >=  0 < 1  0 < 2  0 < 3  0 < 4  1 >= 0  1 < 2  1 >= 3  1 < 4  2 >= 0  2 >= 1  2 >= 3  2 >= 4  3 >= 0  3 >= 1  3 < 2  3 < 4  4 >= 0  4 >= 1  4 < 2  4 >= 3 |  |

Passed

# 2.

Write a class **Book** that stores information about the name of the book (dynamically allocated array of characters), ISBN of the book (dynamically allocated array of 5 integers), and the number of pages (integer). Provide set and get methods for the class that are used, and overload the following operators:

* operator**==** for comparing two books based on their ISBN
* operator **<<** for printing information about the book in the format **Title: name ISBN: [x y z k l]**

Write a class **BorrowABook** for a library for renting books that stores information about the name of the library (array of 100 characters), an array of books (dynamically allocated array of objects of the Book class), and the number of books it has (integer). In the BorrowABook class, write **a constructor with one argument** for initializing the name of the library. Each time a new book is added to the library, the dynamically allocated array should increase its capacity by 1 element. In this class, overload the following operators:

* operator **+=**for adding a new book to the library
* operator **-=**for removing a book from the library (the one with the existing ISBN that matches the ISBN of another book).

As an output, print the name of the library and the list of books it rents, whose number of pages is greater than 150. Finally, create a function **printNumberOfPages(*int max*)** that should be defined in the **BorrowABook** class.

**Main function explanation:** The information about the books is read from the keyboard using the operator +=. In the last line of the input, information is given for a book that is entered incorrectly because it has the same ISBN as another book. It is necessary to remove it.

**Do NOT make changes in the main() function !!!**

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 3  Hamlet 4 5 6 7 4 200  Macbeth 2 3 4 1 2 120  Dracula 1 2 1 1 1 130  Rebecca 1 2 1 1 1 100 | FINKI-Library  Title: Hamlet ISBN: [ 4 5 6 7 4 ] | FINKI-Library  Title: Hamlet ISBN: [ 4 5 6 7 4 ] |  |
|  | 2  Persuasion 5 3 2 1 5 100  Night 1 2 4 5 6 200  It 5 3 2 1 5 100 | FINKI-Library  Title: Night ISBN: [ 1 2 4 5 6 ] | FINKI-Library  Title: Night ISBN: [ 1 2 4 5 6 ] |  |
|  | 4  Beowulf 4 5 6 7 4 200  Candide 2 3 4 1 2 70  Cranford 1 2 1 1 1 150  Beloved 1 2 1 1 1 100  Coraline 2 3 4 1 2 120 | FINKI-Library  Title: Beowulf ISBN: [ 4 5 6 7 4 ] | FINKI-Library  Title: Beowulf ISBN: [ 4 5 6 7 4 ] |  |

Passed

LAB 7

# 1.

Define a class **Square**that stores information about:

* length of the side **a**of the square (*double*)

For the class define the following:

* default constructor
* constructor with arguments
* copy constructor
* *double perimeter()* that calculates the perimeter of the square
* *double area()* that calculates the area of the square
* *void print()* that prints information about the square as in the test cases

Implement another class **Rectangle**that inherits from the class **Square**. This class stores information about:

* the expansion **x**on the two parallel sides of the square (*double*)
* the expansion **y**on the other two parallel sides (*double*)

The **Rectangle**now will have two sides with length **a+x** and two sides with length **a+y**.

About this class define the following:

* default constructor
* *constructor Rectangle(const Square &s, double x, double y)*
* copy constructor

Override the three methods defined above in the class **Square**.

**Warning**:

* If **x**and **y**are the same, then the form is not rectangle it is again square. In that case *print()* is the same as it is for square.
* When overriding the methods in the class **Rectangle**you must call the methods with the same name from the class **Square**.

| * **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 5  4.5  4  10  17.5  8.9  1 2  1.5 2.5  3 4.5  7 7  8 9  2 | ===Testing class Rectangle===  Rectangle with sides: 5.5 and 6.5 P=35.75 and perimeter L=24  Rectangle with sides: 5.5 and 6.5 P=35.75 and perimeter L=24  Rectangle with sides: 13 and 14.5 P=188.5 and perimeter L=55  Square with side a=24.5 has area P=600.25 and perimeter L=98  Rectangle with sides: 16.9 and 17.9 P=302.51 and perimeter L=69.6 | ===Testing class Rectangle===  Rectangle with sides: 5.5 and 6.5 P=35.75 and perimeter L=24  Rectangle with sides: 5.5 and 6.5 P=35.75 and perimeter L=24  Rectangle with sides: 13 and 14.5 P=188.5 and perimeter L=55  Square with side a=24.5 has area P=600.25 and perimeter L=98  Rectangle with sides: 16.9 and 17.9 P=302.51 and perimeter L=69.6 |  |
|  | 5  4.5  4  10  17.5  8.9  1 2  1.5 2.5  3 4.5  7 7  8 9  1 | ===Testing class Square===  Square with side a=4.5 has area P=20.25 and perimeter L=18  Square with side a=4 has area P=16 and perimeter L=16  Square with side a=10 has area P=100 and perimeter L=40  Square with side a=17.5 has area P=306.25 and perimeter L=70  Square with side a=8.9 has area P=79.21 and perimeter L=35.6 | ===Testing class Square===  Square with side a=4.5 has area P=20.25 and perimeter L=18  Square with side a=4 has area P=16 and perimeter L=16  Square with side a=10 has area P=100 and perimeter L=40  Square with side a=17.5 has area P=306.25 and perimeter L=70  Square with side a=8.9 has area P=79.21 and perimeter L=35.6 |  |

Passed

# 2.

For the needs of a vehicle rental company, you need to create a program that will manage the work in the company. The company offers 2 types of vehicles for rent:

* passenger vehicles (**PassengerVehicle**)
* freight vehicles (**FreightVehicle**)

For this purpose, you need to create a **Vehicle** class that will have the following information:

* brand - a string of maximum 100 characters
* model - a string of maximum 100 characters
* year of production - an integer
* rental price per hour - an integer

For this class, you need to create:

* Default constructor
* Constructor with parameters
* Copy constructor
* Operator =
* A function that returns the rental price of the vehicle per hour (**getRentPricePerHour**)
* A **print** function that prints the information about the vehicle in the format:  
  **[Brand] - [Model]**

Furthermore, the **PassengerVehicle** class should inherit from the **Vehicle** class and additionally have the following information:

* number of seats - an integer
* number of doors - an integer

For this class, you need to create:

* Default constructor
* Constructor with parameters
* A function that returns the rental price of the vehicle per hour (**getRentPricePerHour**) which is calculated by adding 10% to the price obtained from the Vehicle class. For example, if the hourly price you get from the Vehicle class is 20 euros, this function should return 22 euros because 20 + 20 \* 0.1 = 22.
* A **print** function that prints the information about the vehicle in the format:  
  **[Brand] - [Model] - [Number of seats] - [Price per hour] euros**

Furthermore, the **FreightVehicle** class should inherit from the **Vehicle** class and additionally have the following information:

* cargo capacity - an integer
* horsepower - an integer

For this class, you need to create:

* Default constructor
* Constructor with parameters
* A function that returns the rental price of the vehicle per hour (**getRentPricePerHour**) which is calculated by adding 50% to the price obtained from the Vehicle class. For example, if the hourly price you get from the Vehicle class is 20 euros, this function should return 30 euros because 20 + 20 \* 0.5 = 30.
* A print function that prints the information about the vehicle in the format:  
  **[Brand] - [Model] - [Cargo capacity] - [Price per hour] euros**

| * **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 3  Peugeot  508  2021  50  5  5  Mercedes-Benz  Actros  2020  100  5000  530  Audi  A4  2015  20  5  5 | Peugeot - 508 - 5 - 55 euros  Mercedes-Benz - Actros - 5000 - 150 euros  Audi - A4 - 5 - 22 euros | Peugeot - 508 - 5 - 55 euros  Mercedes-Benz - Actros - 5000 - 150 euros  Audi - A4 - 5 - 22 euros |  |
|  | 2  BMW  X6  2012  10  5  4  MAZ  5440  2020  200  3000  500 | BMW - X6 - 5 - 11 euros  MAZ - 5440 - 3000 - 300 euros | BMW - X6 - 5 - 11 euros  MAZ - 5440 - 3000 - 300 euros |  |

Passed

LAB 8

# 1.

You are designing a program for a hotel reservation system. For this you will need to create a bass class called **Room** that represents a generic hotel room and has these two pure virtual functions:

1. **displayRoomInfo()**
2. **displayRoomPrice()**

After that you need to create 2 subclasses of **Room** called:

* **StandardRoom**
* **DeluxeRoom**

The **StandardRoom** class should have this information:

* **hasBathroom** - bool variable

and implement both function above with these implementations:

* For **displayRoomInfo()**should print **This is a standard room with a queen-sized bed.**If the room has bathroom then add `**and a bathroom`**to the end of the message.
* For **displayRoomPrice()**should print **The price for a standard room is [price] per night.** where price is calculated based on if the room has bathroom or not. If it has bathroom it's 100$ and if not it's 80$.

The **DeluxeRoom**class should have this information:

* **hasBalcony** - bool variable

and implement both function above with these implementations:

* For **displayRoomInfo()**should print **This is a deluxe room with a king-sized bed, a bathroom, a mini-fridge.**If the room has balcony then add `**and a balcony`**to the end of the message.
* For **displayRoomPrice()**should print **The price for a deluxe room is [price] per night.**where price is calculated based on if the room has balcony or not. If it has balcony it's 200$ and if not it's 160$.

| * **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 1 | TEST CASE 1: TESTING STANDARD ROOM CLASS  This is a standard room with a queen-sized bed.  The price for a standard room is $80 per night.  This is a standard room with a queen-sized bed.  The price for a standard room is $80 per night.  This is a standard room with a queen-sized bed.  The price for a standard room is $80 per night.  This is a standard room with a queen-sized bed.  The price for a standard room is $80 per night.  This is a standard room with a queen-sized bed.  The price for a standard room is $80 per night. | TEST CASE 1: TESTING STANDARD ROOM CLASS  This is a standard room with a queen-sized bed.  The price for a standard room is $80 per night.  This is a standard room with a queen-sized bed.  The price for a standard room is $80 per night.  This is a standard room with a queen-sized bed.  The price for a standard room is $80 per night.  This is a standard room with a queen-sized bed.  The price for a standard room is $80 per night.  This is a standard room with a queen-sized bed.  The price for a standard room is $80 per night. |  |
|  | 2 | TEST CASE 2: TESTING DELUXE ROOM CLASS  This is a deluxe room with a king-sized bed, a bathroom, a mini-fridge.  The price for a deluxe room is $160 per night.  This is a deluxe room with a king-sized bed, a bathroom, a mini-fridge.  The price for a deluxe room is $160 per night.  This is a deluxe room with a king-sized bed, a bathroom, a mini-fridge.  The price for a deluxe room is $160 per night.  This is a deluxe room with a king-sized bed, a bathroom, a mini-fridge.  The price for a deluxe room is $160 per night.  This is a deluxe room with a king-sized bed, a bathroom, a mini-fridge.  The price for a deluxe room is $160 per night. | TEST CASE 2: TESTING DELUXE ROOM CLASS  This is a deluxe room with a king-sized bed, a bathroom, a mini-fridge.  The price for a deluxe room is $160 per night.  This is a deluxe room with a king-sized bed, a bathroom, a mini-fridge.  The price for a deluxe room is $160 per night.  This is a deluxe room with a king-sized bed, a bathroom, a mini-fridge.  The price for a deluxe room is $160 per night.  This is a deluxe room with a king-sized bed, a bathroom, a mini-fridge.  The price for a deluxe room is $160 per night.  This is a deluxe room with a king-sized bed, a bathroom, a mini-fridge.  The price for a deluxe room is $160 per night. |  |
|  | 3  1  0  0  1  1 | TEST CASE 3: TESTING BOTH CLASSES  This is a standard room with a queen-sized bed and a bathroom.  The price for a standard room is $100 per night.  This is a deluxe room with a king-sized bed, a bathroom, a mini-fridge.  The price for a deluxe room is $160 per night.  This is a standard room with a queen-sized bed.  The price for a standard room is $80 per night.  This is a deluxe room with a king-sized bed, a bathroom, a mini-fridge and a balcony.  The price for a deluxe room is $200 per night.  This is a standard room with a queen-sized bed and a bathroom.  The price for a standard room is $100 per night. | TEST CASE 3: TESTING BOTH CLASSES  This is a standard room with a queen-sized bed and a bathroom.  The price for a standard room is $100 per night.  This is a deluxe room with a king-sized bed, a bathroom, a mini-fridge.  The price for a deluxe room is $160 per night.  This is a standard room with a queen-sized bed.  The price for a standard room is $80 per night.  This is a deluxe room with a king-sized bed, a bathroom, a mini-fridge and a balcony.  The price for a deluxe room is $200 per night.  This is a standard room with a queen-sized bed and a bathroom.  The price for a standard room is $100 per night. |  |

Passed

# 2.

You need to write a class called **FoodItem**that has these two pure virtual functions:

* **getPrice()**- the price of the food
* **getTime()**- the time it takes to prepare the food

The FoodItem class should also have the following information:

* **type**(dynamically allocated character array) - the type of food
* **num**(int) - the quantity of the food

Furthermore, you need to create two subclasses of **FoodItem**called:

**Pizza**

**Steak**

The **Pizza**class should have the following information:

* **dough** (dynamically allocated character array) - the type of dough for the pizza

Price of one dough:

"wholeWheat": 250

"glutenFree": 350

The time it takes to make pizza is 25 minutes, regardless of how many.

If you are making **n**pizzas you need **n**much of the dough.

The **Steak**class should have the following information:

* **bool cooked**(whether it is well done or medium rare)

The time it takes to cook a steak, regardless of how many:

-well done: 20 min

-medium rare: 15 min

The price of one steak is 1300 den.

You should also write a global function:

**getMaxFoodItem(FoodItem \*pItem[], int n)**, which returns a pointer to the most expensive FoodItem

**Do not modify the main function!**

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 3  pizza 2 wholeWheat  steak 1 0  steak 2 1 | Type: steak  The max price is: 2600  Time to cook: 20 | Type: steak  The max price is: 2600  Time to cook: 20 |  |
|  | 5  pizza 5 wholeWheat  steak 1 0  steak 1 1  pizza 5 glutenFree  steak 1 1 | Type: pizza  The max price is: 1750  Time to cook: 25 | Type: pizza  The max price is: 1750  Time to cook: 25 |  |
|  | 4  pizza 7 wholeWheat  steak 1 0  pizza 5 glutenFree  steak 2 0 | Type: steak  The max price is: 2600  Time to cook: 15 | Type: steak  The max price is: 2600  Time to cook: 15 |  |

Passed

LAB 9

# 1.

Implement a class called **Employee** which stores the following information:

* name (string)
* age (integer)

The class should have a virtual function called displayInfo() which prints information about the employee.

Next, implement a class called **Payable** which stores:

* salary (floating-point number)

The class should have a pure virtual function called calculateSalary().

From these classes, derive two classes: "Developer" and "Manager".

For the **Developer** class, add an additional member variable:

* programmingLanguage (string)

Override the necessary functions in the derived classes. The salary for a developer is calculated by deducting a 10% tax from the base salary.

For the **Manager** class, add an additional member variable:

* numberOfDepartments (integer)

Override the necessary functions in the derived classes. The salary for a manager is calculated by adding a 5% bonus for each department they are responsible for.

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | John  30  5000  C++  Mike  40  8000  3  Jane  45  6000  Java  Smith  28  5700  2  Ana  33  6200  Python | -- Developer Information --  Name: John  Age: 30  Salary: $4500  Programming Language: C++  -- Manager Information --  Name: Mike  Age: 40  Salary: $9200  Number of Departments: 3  -- Developer Information --  Name: Jane  Age: 45  Salary: $5400  Programming Language: Java  -- Manager Information --  Name: Smith  Age: 28  Salary: $6270  Number of Departments: 2  -- Developer Information --  Name: Ana  Age: 33  Salary: $5580  Programming Language: Python  Biggest Salary: 9200 | -- Developer Information --  Name: John  Age: 30  Salary: $4500  Programming Language: C++  -- Manager Information --  Name: Mike  Age: 40  Salary: $9200  Number of Departments: 3  -- Developer Information --  Name: Jane  Age: 45  Salary: $5400  Programming Language: Java  -- Manager Information --  Name: Smith  Age: 28  Salary: $6270  Number of Departments: 2  -- Developer Information --  Name: Ana  Age: 33  Salary: $5580  Programming Language: Python  Biggest Salary: 9200 |  |
|  | Sarah  27  4700  C  Bob  52  7800  9  Maria  39  3150  C#  John  42  1520  5  David  23  2520  Java | -- Developer Information --  Name: Sarah  Age: 27  Salary: $4230  Programming Language: C  -- Manager Information --  Name: Bob  Age: 52  Salary: $11310  Number of Departments: 9  -- Developer Information --  Name: Maria  Age: 39  Salary: $2835  Programming Language: C#  -- Manager Information --  Name: John  Age: 42  Salary: $1900  Number of Departments: 5  -- Developer Information --  Name: David  Age: 23  Salary: $2268  Programming Language: Java  Biggest Salary: 11310 | -- Developer Information --  Name: Sarah  Age: 27  Salary: $4230  Programming Language: C  -- Manager Information --  Name: Bob  Age: 52  Salary: $11310  Number of Departments: 9  -- Developer Information --  Name: Maria  Age: 39  Salary: $2835  Programming Language: C#  -- Manager Information --  Name: John  Age: 42  Salary: $1900  Number of Departments: 5  -- Developer Information --  Name: David  Age: 23  Salary: $2268  Programming Language: Java  Biggest Salary: 11310 |  |

Passed

# 2.

Implement a class called **Game** which stores the following information:

* title (character array)

The class should have virtual function **void** **displayInfo()** which prints information for the game and virtual function**int complexity()**which returns the complexity of the game.

Implement the classes **BoardGame**and **CardGame**which will inherit from the class **Game**.

The class **BoardGame**stores the following information:

* **maxPlayers** - maximum number of players (integer)

If the maximum number of players is greater than 6, then the complexity is 10, if the maximum number of players is greater than 3 and less than 6, then the complexity is 5, and if it is less than 3, then the complexity is 3.

The function **displayInfo()**prints:

Game: [game title]

Max players: [maximum number of players]

Complexity: [complexity]

The class **CardGame**stores the following information:

* **numCards** - number of cards (integer)

If the number of cards is greater than 20, then the complexity is 10, if the number of cards is greater than 10 and less than 20, then the complexity is 5, and if it is less than 10, then the complexity is 3.

The function **displayInfo()**prints:

Game: [game title]

Number of cards: [number of cards]

Complexity: [complexity]

Implement the class **BoardCardGame**which will inherit from the classes **BoardGame** and **CardGame**.  
  
The complexity is calculated as follows: **(CardGame complexity + BoardGame complexity)/2**

The function **displayInfo()**prints:

Game: [game title]

Complexity: [complexity]

Create a global function**Game \*mostComplexCardGame(Game \*\*games, int n)** which returns a pointer to the **most complex card game**.

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 1  Jenny | Game: Jenny  Complexity: 7 | Game: Jenny  Complexity: 7 |  |
|  | 0  6  1  Game1 7  0  Game2 8  1  Game3 2  0  Game4 15  1  Game5 7  0  Game6 9  1 | Game: Game4  Number of cards: 15  Complexity: 5 | Game: Game4  Number of cards: 15  Complexity: 5 |  |
|  | 0  5  1  Game1 8  0  Game2 8  1  Game3 2  0  Game4 52  1  Game5 4  0 | Game: Game4  Number of cards: 52  Complexity: 10 | Game: Game4  Number of cards: 52  Complexity: 10 |  |

Passed

LAB 10

# 1.

You are given the **User** and **SocialNetwork** classes.

The **User** class stores information about username, age and number of friends within a social network.

The **SocialNetwork** class stores a list of members (objects of the User class) and the size of the list.

Add in the initial code the following things:

1. Prevent adding a user with an already existing username by throwing an exception of type **UserAlreadyExistsException**. Handle the exception where necessary.

2. Add a variable for the limit of friends allowed per user (integer). The variable should be valid for all users (objects of the **User** class) and it can be changed through the **setLimit** method. Initially, the variable should have a value of 3.

3. Throw an exception of type **UserNotFoundException** in the friendRequest method if friendRequest or secondUsername are not usernames of users in the social network. Handle the exception where necessary.

4. To prevent an increase in the number of friends of a user (beyond the allowed limit) with an exception of type **FriendsLimitExceededException**. Handle the exception where necessary.

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 5  stefan.andonov 19  vlatko.spasev 20  marko.petrov 21  dimitar.kitanovski 22  ema.pandilova 18  0  0  0 | Registration of users  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: dimitar.kitanovski Age: 22 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  Friend requests  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: dimitar.kitanovski Age: 22 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  CHANGE STATIC VALUE  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: dimitar.kitanovski Age: 22 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0 | Registration of users  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: dimitar.kitanovski Age: 22 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  Friend requests  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: dimitar.kitanovski Age: 22 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  CHANGE STATIC VALUE  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: dimitar.kitanovski Age: 22 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0 |  |
|  | 5  stefan.andonov 19  vlatko.spasev 20  marko.petrov 21  stefan.andonov 22  ema.pandilova 18  0  0  0 | User with username stefan.andonov already exists!  Registration of users  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  Friend requests  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  CHANGE STATIC VALUE  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0 | User with username stefan.andonov already exists!  Registration of users  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  Friend requests  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  CHANGE STATIC VALUE  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0 |  |
|  | 5  stefan.andonov 19  vlatko.spasev 20  marko.petrov 21  dimitar.kitanovski 22  ema.pandilova 18  2  stefan.andonov vlatko.spasev  marko.petrov dimitar.kitanovski  0  0 | Registration of users  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: dimitar.kitanovski Age: 22 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  Friend requests  Users:  Username: stefan.andonov Age: 19 # of friends: 1  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 1  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 0  CHANGE STATIC VALUE  Users:  Username: stefan.andonov Age: 19 # of friends: 1  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 1  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 0 | Registration of users  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: dimitar.kitanovski Age: 22 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  Friend requests  Users:  Username: stefan.andonov Age: 19 # of friends: 1  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 1  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 0  CHANGE STATIC VALUE  Users:  Username: stefan.andonov Age: 19 # of friends: 1  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 1  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 0 |  |
|  | 5  stefan.andonov 19  vlatko.spasev 20  marko.petrov 21  dimitar.kitanovski 22  ema.pandilova 18  5  stefan.andonov vlatko.spasev  marko.petrov dimitar.kitanovski  stefan.andonov marko.petrov  stefan.andonov ema.pandilova  marko.petrov vlatko.spasev  0  0 | Registration of users  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: dimitar.kitanovski Age: 22 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  Friend requests  Users:  Username: stefan.andonov Age: 19 # of friends: 3  Username: vlatko.spasev Age: 20 # of friends: 2  Username: marko.petrov Age: 21 # of friends: 3  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 1  CHANGE STATIC VALUE  Users:  Username: stefan.andonov Age: 19 # of friends: 3  Username: vlatko.spasev Age: 20 # of friends: 2  Username: marko.petrov Age: 21 # of friends: 3  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 1 | Registration of users  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: dimitar.kitanovski Age: 22 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  Friend requests  Users:  Username: stefan.andonov Age: 19 # of friends: 3  Username: vlatko.spasev Age: 20 # of friends: 2  Username: marko.petrov Age: 21 # of friends: 3  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 1  CHANGE STATIC VALUE  Users:  Username: stefan.andonov Age: 19 # of friends: 3  Username: vlatko.spasev Age: 20 # of friends: 2  Username: marko.petrov Age: 21 # of friends: 3  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 1 |  |
|  | 5  stefan.andonov 19  vlatko.spasev 20  marko.petrov 21  dimitar.kitanovski 22  ema.pandilova 18  5  stefan.andonov vlatko.spasev  marko.petrov dimitar.kitanovski  stefan.andonov marko.petrov  stefan.andonov ema.pandilova  stefan.andonov dimitar.kitanovski  0  0 | Registration of users  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: dimitar.kitanovski Age: 22 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  Friend requests  Can't have more than 3 friends.  Users:  Username: stefan.andonov Age: 19 # of friends: 3  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 2  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 1  CHANGE STATIC VALUE  Users:  Username: stefan.andonov Age: 19 # of friends: 3  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 2  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 1 | Registration of users  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: dimitar.kitanovski Age: 22 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  Friend requests  Can't have more than 3 friends.  Users:  Username: stefan.andonov Age: 19 # of friends: 3  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 2  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 1  CHANGE STATIC VALUE  Users:  Username: stefan.andonov Age: 19 # of friends: 3  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 2  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 1 |  |
|  | 5  stefan.andonov 19  vlatko.spasev 20  marko.petrov 21  dimitar.kitanovski 22  ema.pandilova 18  5  stefan.andonov vlatko.spasev  marko.petrov dimitar.kitanovski  stefan.andonov marko.petrov  stefan.andonov ema.pandilova  stefan.andonov dimitar.kitanovski  5  1  stefan.andonov dimitar.kitanovski | Registration of users  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: dimitar.kitanovski Age: 22 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  Friend requests  Can't have more than 3 friends.  Users:  Username: stefan.andonov Age: 19 # of friends: 3  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 2  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 1  CHANGE STATIC VALUE  Users:  Username: stefan.andonov Age: 19 # of friends: 4  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 2  Username: dimitar.kitanovski Age: 22 # of friends: 2  Username: ema.pandilova Age: 18 # of friends: 1 | Registration of users  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: dimitar.kitanovski Age: 22 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  Friend requests  Can't have more than 3 friends.  Users:  Username: stefan.andonov Age: 19 # of friends: 3  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 2  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 1  CHANGE STATIC VALUE  Users:  Username: stefan.andonov Age: 19 # of friends: 4  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 2  Username: dimitar.kitanovski Age: 22 # of friends: 2  Username: ema.pandilova Age: 18 # of friends: 1 |  |
|  | 5  stefan.andonov 19  vlatko.spasev 20  marko.petrov 21  dimitar.kitanovski 22  ema.pandilova 18  5  stefan.andonov vlatko.spasev  marko.petrov dimitar.kitanovski  stefan.andonov marko.petrov  stefan.andonov ema.pandilova  marko.petrov petre.petrev  0  0 | Registration of users  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: dimitar.kitanovski Age: 22 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  Friend requests  User with username petre.petrev was not found!  Users:  Username: stefan.andonov Age: 19 # of friends: 3  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 2  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 1  CHANGE STATIC VALUE  Users:  Username: stefan.andonov Age: 19 # of friends: 3  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 2  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 1 | Registration of users  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: dimitar.kitanovski Age: 22 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  Friend requests  User with username petre.petrev was not found!  Users:  Username: stefan.andonov Age: 19 # of friends: 3  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 2  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 1  CHANGE STATIC VALUE  Users:  Username: stefan.andonov Age: 19 # of friends: 3  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 2  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 1 |  |
|  | 5  stefan.andonov 19  vlatko.spasev 20  marko.petrov 21  dimitar.kitanovski 22  ema.pandilova 18  5  stefan.andonov vlatko.spasev  marko.petrov dimitar.kitanovski  riste.ridtovski marko.petrovski  stefan.andonov ema.pandilova  marko.petrov petre.petrev  0  0 | Registration of users  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: dimitar.kitanovski Age: 22 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  Friend requests  User with username riste.ridtovski was not found!  User with username petre.petrev was not found!  Users:  Username: stefan.andonov Age: 19 # of friends: 2  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 1  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 1  CHANGE STATIC VALUE  Users:  Username: stefan.andonov Age: 19 # of friends: 2  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 1  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 1 | Registration of users  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: dimitar.kitanovski Age: 22 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  Friend requests  User with username riste.ridtovski was not found!  User with username petre.petrev was not found!  Users:  Username: stefan.andonov Age: 19 # of friends: 2  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 1  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 1  CHANGE STATIC VALUE  Users:  Username: stefan.andonov Age: 19 # of friends: 2  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 1  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 1 |  |
|  | 5  stefan.andonov 19  vlatko.spasev 20  marko.petrov 21  dimitar.kitanovski 22  ema.pandilova 18  8  stefan.andonov vlatko.spasev  marko.petrov dimitar.kitanovski  stefan.andonov marko.petrov  stefan.andonov ema.pandilova  stefan.andonov dimitar.kitanovski  marko.petrov ema.pandilova  dimitar.kitanovski marko.petrov  vlatko.spasev marko.petrov  5  3  marko.petrov ema.pandilova  dimitar.kitanovski marko.petrov  vlatko.spasev marko.petrov | Registration of users  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: dimitar.kitanovski Age: 22 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  Friend requests  Can't have more than 3 friends.  Can't have more than 3 friends.  Can't have more than 3 friends.  Users:  Username: stefan.andonov Age: 19 # of friends: 3  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 3  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 2  CHANGE STATIC VALUE  Can't have more than 5 friends.  Users:  Username: stefan.andonov Age: 19 # of friends: 3  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 5  Username: dimitar.kitanovski Age: 22 # of friends: 2  Username: ema.pandilova Age: 18 # of friends: 3 | Registration of users  Users:  Username: stefan.andonov Age: 19 # of friends: 0  Username: vlatko.spasev Age: 20 # of friends: 0  Username: marko.petrov Age: 21 # of friends: 0  Username: dimitar.kitanovski Age: 22 # of friends: 0  Username: ema.pandilova Age: 18 # of friends: 0  Friend requests  Can't have more than 3 friends.  Can't have more than 3 friends.  Can't have more than 3 friends.  Users:  Username: stefan.andonov Age: 19 # of friends: 3  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 3  Username: dimitar.kitanovski Age: 22 # of friends: 1  Username: ema.pandilova Age: 18 # of friends: 2  CHANGE STATIC VALUE  Can't have more than 5 friends.  Users:  Username: stefan.andonov Age: 19 # of friends: 3  Username: vlatko.spasev Age: 20 # of friends: 1  Username: marko.petrov Age: 21 # of friends: 5  Username: dimitar.kitanovski Age: 22 # of friends: 2  Username: ema.pandilova Age: 18 # of friends: 3 |  |

# 2.

Impplement a class Race that contains:

* city name(array of max 40 characters)
* year (int)
* month (int)
* best time in seconds (double)
* length in kilometers (double)

You should also implement the following methods:

* **heaviness**() that returns the heaviness of the race calculated by the following formula:

**(best time in seconds / length in kilometers)**

* **operator<<** for printing the race in the following format:

[City Month.Year Heaviness]

Implement another class **CarRace**which will contain:

* array of best times of practices(dynamically reserved array of decimal numbers)
* number of best times (int)
* number of laps (int)

In this class override the function **heaviness**() so it should return the basic value of the Race and add the average from the best times multiplied by the **CAR\_COEF**, which is initially 0.3 but it can be changed. Additionally if the number of lap sis bigger than 15, the heaviness is greater 10%. For this class you should implement:

* operator += for adding new time from practices in dynamically allocated array of best times. If someone adds time that is less than 10, exception should be thrown from the class InvalidTimeException, that should print "The time is not valid".

| * **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 1  Paris  2011  5  1986.3  49.6  Jerez  2017  6  41.53  4.429  24  5  49.2  39.11  42.4  54.3  44.4 | ===== Testing the classes ======  Paris 5.2011 40.0464  Jerez 6.2017 25.4556 | ===== Testing the classes ======  Paris 5.2011 40.0464  Jerez 6.2017 25.4556 |  |
|  | 2  5  1  Madrid  2015  1  206.3  34.5  2  Mugello  2019  6  99.7  5.245  20  4  105.2  103.4  99.99  34.2  1  Skopje  2013  8  70.0  1844.5  2  Brno  2017  7  85.824  4.533  24  4  105.2  103.4  99.99  34.2  1  Sofia  2015  9  85.824  4.533  55.55 | ===== Testing the operator += ======  List of all races:  Madrid 1.2015 5.97971  Mugello 6.2019 49.1896  Skopje 8.2013 0.0379507  Brno 7.2017 49.1067  Sofia 9.2015 18.9332  List of all races:  Madrid 1.2015 5.97971  Mugello 6.2019 47.1999  Skopje 8.2013 0.0379507  Brno 7.2017 49.1067  Sofia 9.2015 18.9332 | ===== Testing the operator += ======  List of all races:  Madrid 1.2015 5.97971  Mugello 6.2019 49.1896  Skopje 8.2013 0.0379507  Brno 7.2017 49.1067  Sofia 9.2015 18.9332  List of all races:  Madrid 1.2015 5.97971  Mugello 6.2019 47.1999  Skopje 8.2013 0.0379507  Brno 7.2017 49.1067  Sofia 9.2015 18.9332 |  |
|  | 3  Qatar  2011  3  54.822  5.535  24  33.33  5.5 | ===== Testing the exceptions ======  The time is not valid  Qatar 3.2011 21.894 | ===== Testing the exceptions ======  The time is not valid  Qatar 3.2011 21.894 |  |
|  | 5  5  1  Madrid  2015  1  206.3  34.5  2  Mugello  2019  6  99.7  5.245  20  4  105.2  103.4  99.99  34.2  1  Skopje  2013  8  70.0  1844.5  2  Brno  2017  7  85.824  4.533  24  4  105.2  103.4  99.99  34.2  1  Sofia  2015  9  85.824  4.533  55.55 | ===== Testing static members ======  List of all races:  Madrid 1.2015 5.97971  Mugello 6.2019 49.1896  Skopje 8.2013 0.0379507  Brno 7.2017 49.1067  Sofia 9.2015 18.9332  List of all races:  Madrid 1.2015 5.97971  Mugello 6.2019 86.8965  Skopje 8.2013 0.0379507  Brno 7.2017 86.8136  Sofia 9.2015 18.9332 | ===== Testing static members ======  List of all races:  Madrid 1.2015 5.97971  Mugello 6.2019 49.1896  Skopje 8.2013 0.0379507  Brno 7.2017 49.1067  Sofia 9.2015 18.9332  List of all races:  Madrid 1.2015 5.97971  Mugello 6.2019 86.8965  Skopje 8.2013 0.0379507  Brno 7.2017 86.8136  Sofia 9.2015 18.9332 |  |

Passed