Data Structures Fundamentals Exam

1. Delivery System – 100 points

You've been tasked with implementing a program for managing a delivery system. The software should work with deliverers, which deliver packages.

You are given a skeleton with a class **DeliveriesManagerImpl** that implements the **DeliveriesManager interface.**

This **Delivery System** works with **Deliverers** and **Packages** as entities. All entities are identified by a **unique Id**.

The **Deliverer** entity contains the following properties:

- **Id** – string
- **Name** – string

The **Package** entity contains the following properties:

- **Id** – string
- **Receiver** – string
- **Address** – string
- **Phone** – string
- **Weight** – double

Implement the following functionalities to make the **Delivery System** software fully operative:

- **void addDeliverer(Deliverer deliverer)** – **adds** an **deliverer** to the **Delivery System** software.
- **void addPackage(Package _package)** – **adds** a **package** to the **Delivery System** software.
- **bool contains(Deliverer deliverer)** – returns whether the **deliverer** is **contained** inside the **Delivery System** software.
- **bool contains(Package _package)** – returns whether the **package** is **contained** inside the **Delivery System** software.
- **Iterable<Deliverer> getDeliverers() –** returns a collection of all **deliverers**.
- **Iterable<Package> getPackages() –** returns a collection of all **packages**.
- **void assignPackage(Deliverer deliverer, Package _package) –** assigns the given **package** to the given **deliverer**. If the **deliverer** or the **package** do **not exist** in the Delivery System - **`throw IllegalArgumentException()`**
- **Iterable<Package> getUnassignedPackages() –** returns a collection of all **packages**, which have not been assigned to any **deliverer**.
- **Iterable<Package> getPackagesOrderedByWeightThenByReceiver()** – returns all of the **packages** ordered by **weight** in **descending order**, then by **receiver** in **alphabetical (ascending) order**.
  If there aren't any packages – return an **empty collection**.
- **Iterable<Deliverer> getDeliverersOrderedByCountOfPackagesThenByName()** – returns all of the **deliverers** ordered by **count of packages** in **descending order**, then by **name** in **alphabetical (ascending) order**.
  If there aren't any deliverers – return an **empty collection**.

NOTE: If all sorting criteria fails, you should order by order of input. This is for all methods with ordered output.

1.5 Delivery System – Performance – 50 points

For this task you will only be required to submit the **code from the previous problem**. If you are having a problem with this task you should **perform detailed algorithmic complexity analysis** and try to **figure out weak** spots inside your implementation.

For this problem it is important that other operations are **implemented correctly** according to the specific problems: **add**, **size**, **remove**, **get** etc… Also, make sure you are using the correct data structures. ☺

You can submit code to this problem **without full coverage** from the previous problem, **not all test cases** will be considered, only the **general behaviour** will be important, **edge cases** will mostly be ignored such as throwing exceptions etc…

2. Airlines – 100 points

You've been tasked with implementing a program for managing an airline tracking system. The software should work with airlines and flights.

You are given a skeleton with a class **AirlinesManagerImpl** that implements the **AirlinesManager interface.**

This **Airlines System** works with **Airlines** and **Flights** as entities. All entities are identified by a **unique Id**.

The **Airline** entity contains the following properties:

- **Id** – string
- **Name** – string
- **Rating** - double

The **Flight** entity contains the following properties:

- **Id** – string
- **Number** – string
- **Origin** – string
- **Destination** – string
- **IsCompleted** – boolean

Implement the following functionalities to make the **Airlines System** software fully operative:

- **void addAirline(Airline airline)** – **adds** an **airline** to the **Airlines System** software.
- **void addFlight(Airline airline, Flight flight)** – **adds** a **flight** to the given **airline** in the **Airlines System** software. If the airline does not exist - **`throw IllegalArgumentException()`**
- **bool contains(Airline airline)** – returns whether the **airline** is **contained** inside the **Airlines System** software.
- **bool contains(Flight flight)** – returns whether the **flight** is **contained** inside the **Airlines System** software.
- **void deleteAirline(Airline airline)** – **removes** the given **airline** from the **Airlines System** software and every Flight associated with it. If the airline does not exist - **`throw IllegalArgumentException()`**
- **Iterable<Flight> getAllFlights() –** returns a collection of all **flights**.

- **Flight performFlight(Airline airline, Flight flight)** – performs the **given flight – setting** its **IsCompleted** property to **true**, and **returning** it as a **result**. If the **airline** or the **flight** do **not exist** in the Airline System - `throw IllegalArgumentException()`
- **Iterable<Flight> getCompletedFlights()** – returns a collection of all **completed flights**.
- **Iterable<Flight> getFlightsOrderedByCompletionThenByNumber()** – returns all of the **flights** ordered by **number** in **asceding (alphabetical) order**.
    - **NOTE**: Flights that are not completed (**IsCompleted** is **false**) should be **returned FIRST** (**completeness** is the **first ordering criteria**).

  If there aren't any flights – return an **empty collection**.
- **Iterable<Airline> getAirlinesOrderedByRatingThenByCountOfFlightsThenByName()** – returns all of the **airlines** ordered by **rating** in **descending order**, then by **count** of **flights** in **descending order**, then by **name** in **ascending** (**alphabetical**) order.
  If there aren't any airlines – return an **empty collection**.
- **Iterable<Airline> getAirlinesWithFlightsFromOriginToDestination(String origin, String destination)** – returns all of the **airlines** which contain **atleast 1 flight**, which is not completed (**IsCompleted** is **false**) and has **origin equal** to the **given one** and **destination equal** to the **given one**.
  If there aren't any eligible results – return an **empty collection**.

2.5 Airlines – Performance – 50 points

For this task you will only be required to submit the **code from the previous problem**. If you are having a problem with this task you should **perform detailed algorithmic complexity analysis** and try to **figure out weak** spots inside your implementation.

For this problem it is important that other operations are **implemented correctly** according to the specific problems: **add**, **size**, **remove**, **get,** etc. Also, make sure you are using the correct data structures. ☺

You can submit code to this problem **without full coverage** from the previous problem, **not all test cases** will be considered, only the **general behaviour** will be important, **edge cases** will mostly be ignored such as throwing exceptions, etc.