**Data Structures Fundamentals – Exam**

# **GitHub**

**Correctness - 100 pts**

The GitHub repository management system handles repositories, commits, collaboration among developers, and user profiles. Implement the following functionalities to make the GitHub-like system fully operative::

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

* **Id** – string
* **Username** – string
* **Email** – string

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

* **Id** – string
* **Name** – string
* **OwnerId** – string
* **Starts** – integer

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

* **Id** – string
* **RepositoryId** – string
* **UserId** – string
* **Message** – string
* **Timestamp** – long

Implement the following functionalities to make the system fully operative:

* **void Create(User user)** – Creates a new user profile.
* **void Create(Repository repository)** – Creates a new code repository.
* **bool Contains(User user)** – Returns true if the given user is present in the system, otherwise returns false.
* **bool Contains(Repository repository)** – Returns true if the given repository is present in the system, otherwise returns false.
* **void CommitChanges(Commit commit)**
  + Creates a new commit in the specified repository with the provided author's unique identifier (authorId).
  + If the repository does not exist, throw an ArgumentException.
  + If the user does not exist, throw an ArgumentException.
* **Repository ForkRepository(string repositoryId, string userId)** - Forks the repository with the specified unique identifier (repositoryId) to create a new repository (reusing only the name and the commits) owned by the user with the provided username.
  + If the repository does not exist, throw an ArgumentException.
  + If the user does not exist, throw an ArgumentException.
* **IEnumerable<Commit> GetCommitsForRepository(string repositoryId)** - Returns a collection of all commits made to the repository with the specified unique identifier (repositoryId).
* **IEnumerable<Repository> GetRepositoriesByOwner(string userId)** - Returns a collection of repositories owned by the user with the given user id.
* **IEnumerable<Repository> GetMostForkedRepositories()** - Returns a collection of repositories ordered by the number of forks in descending order.
* **IEnumerable<Repository> GetRepositoredOrderedByCommitsInDescending()** - Returns a collection of repositories ordered by the number of commits in descending order.

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

Please implement the above functionalities in a " GitHubManager " class that implements the "**IGitHubManager**" interface. The "**IGitHubManager**" interface should contain the method signatures for the above functionalities.

**Performance – 50 pts**

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 a 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** **behavior** will be important, and **edge** **cases** will mostly be ignored such as throwing exceptions, etc…