**Data Structures Advanced with C# - Regular Exam**

# Task Manager – 100 pts

You have been tasked with designing a task management system for a software development team. The system should be able to **handle tasks** with various properties and support different operations. You are given a skeleton with a class called "**Manager**" that implements the "**IManager**" interface.

The "**TaskManager**" class should work with "**Task**" entities, and all tasks are identified by a **unique ID**. Each task contains the following properties:

* **Id** - string
* **Title** - string
* **Assignee** - string (the person responsible for the task)
* **Priority** - an integer value (1 for low, 2 for medium, 3 for high)

Implement the following functionalities to make **TaskManager** fully operative:

* **void AddTask(Task task)** - adds a task to the task manager. If a task with the same ID already exists, throw an ArgumentException.
* **void RemoveTask(string taskId)** - removes the task with the given ID from the task manager. If no task with that ID exists, throw an ArgumentException.
* **bool Contains(string taskId)** - returns whether the task manager contains the specified task.
* **Task Get(string taskId)** - returns the task in the system. If no task with that ID exists, throw an ArgumentException.
* **int Size()** - returns the total count of all tasks in the task manager.
* **void AddDependency(string taskId, string dependentTaskId)** - Adds a dependency between two tasks. The **task with taskId depends on the task with dependentTaskId**. If **either task doesn't exist** or if a **circular dependency is created**, **throw an ArgumentException**.
  + Circular dependency meaning that if **A depends on B**, then **B cannot depends on A**.
  + Keep in mind, that new dependency carries other dependencies too. If **A depends on B**, and **B depends on C**, then **A depends on C indirectly**.
* **void RemoveDependency(string taskId, string dependentTaskId)** - Removes a **dependency between two tasks**. If **either task doesn't exist** or **if there is no dependency between them**, **throw an ArgumentException**.
* **IEnumerable<Task> GetDependencies(string taskId)** - Returns all tasks that the **task with the given taskId depends on**. If the **taskId** does not exist, return empty collection.
  + If **A depends on B**, and **A depends on C:**
    - All dependencies for A are B and C.
    - B and C do not have any dependencies.
* **IEnumerable<Task> GetDependants(string taskId)** - Returns all tasks that **depend on the specified taskId**. If the **taskId** does not exist, return empty collection.
  + If **A depends on B**, and **A depends on C**:
    - A does not have any dependants.
    - B has A as a dependant.
    - C has A as a dependant.
  1. **Task Manager – 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 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…