**Data Structures Fundamentals – Exam**

# **National Election System**

**Correctness - 100 pts**

The **National Election System** is responsible for managing candidates and voters for a country's national elections. All entities are identified by a **unique Id**. The entities are defined as follows:

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

* **Id** – string
* **Name** – string
* **Party** – string

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

* **Id** – string
* **Name** – string
* **Age** – int

Implement the following functionalities to make the system fully operative:

* **void AddCandidate(Candidate candidate)** – Adds a candidate to the election system.
* **void AddVoter(Voter voter)** - Adds a voter to the election system.
* **bool Contains(Candidate candidate)** - Returns true if the given candidate is present in the election system, otherwise returns false.
* **bool Contains(Voter voter)** - Returns true if the given voter is present in the election system, otherwise returns false.
* **IEnumerable<Candidate> GetCandidates()** - Returns a collection of all candidates in the election system.
* **IEnumerable<Voter> GetVoters()** - Returns a collection of all voters in the election system.
* **void Vote(string voterId, string candidateId)** - Registers a vote from the given voter to the specified candidate.
  + If the voter is not eligible (age below 18), throw an ArgumentException.
  + If the voter or candidate is not present in the election system, throw an ArgumentException.
  + If the voter already voted, throw an ArgumentException.
* **int GetVotesForCandidate(string candidateId)** - Returns the number of votes received by the specified candidate.
* **IEnumerable<Candidate> GetWinner()** - Returns the candidate who received the highest number of votes.
  + If there is a tie, return the candidates who are tied.
  + If no votes have been cast yet (no winner determined), return null.
* **IEnumerable<Candidate>** GetCandidatesByParty(string party) - Returns a collection of candidates belonging to the specified part, ordered by the number of votes 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 class named "**ElectionManager**" that implements the "**IElectionManager**" interface. The "**IElectionManager**" 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…