Hiring - Lobby Wars

Description

We are in the era of "lawsuits", everyone wants to go to court with their lawyer Saul and try to get a lot of dollars as if they were raining over Manhattan.

The laws have changed much lately and governments have been digitized. That's when **Signaturit** comes into play.

The city council through the use of Signaturit maintains a registry of legal signatures of each party involved in the contracts that are made.

During a trial, justice only verifies the signatures of the parties involved in the contract to decide who wins. For that, they assign points to the different signatures depending on the role of who signed.

For example, if the plaintiff has a contract that is signed by a **notary** he gets 2 points, if the defendant has in the contract the signature of only a **validator** he gets only 1 point, so the plaintiff party wins the trial.

Roles

- King (K): 5 points.
- Notary (N): 2 points.
- Validator (V): 1 point.

> Keep in mind that when a King signs, the signatures of the validators on his part have no value.

First phase

We want you to automate this process, given a contract with your 2 parties involved and their signatures, and indicate which one wins the test.

Make a program that receives both contracts as input (for example `KN vs NNV`) and gives the winner as output. We should be able to interact from console or HTTP.

Second stage

Sometimes the contract does not have all the signs, so we represent it using the `#`character. Taking into account that only one signature per part can be empty to be valid, determine which is the minimum signature necessary to win the trial given a contract with the signatures of the known opposition party.

For example, given `N#V vs NVV` should return `N`.

Make a program that receives both contracts as input (for example `N#V vs NVV`) and gives the signature required to win as output. We should be able to interact from console or HTTP.

> What we would like to find (although not mandatory):
>
> - Best practices as Clean Code, SOLID, Unit testing.
> - Usage of DDD and Hexagonal Architecture.
> - Apply Command and Query
> - zip with a git repository so we see how you structure your commits.