# Финален изпит

Тествайте решението в Judge: <https://judge.softuni.org/Contests/4341/17-Final-Exam>.

## Vacc Ops

Всички говорят за ваксини и зелени сертификати. Също така, както можете да предположите, системите, управляващи личните лекари и техните пациенти, са морално остарели. Сега вашата задача е да внедрите нова и свежа система.

Даден ви е скелет с клас **"VaccOps"**, който имплементира интерфейса **"IVaccOps".** Класът **"VaccOps"** работи с обектите **"Doctor"** и **"Patient"**, като всеки обект има **уникално id**. Вашата задача е да имплементирате всички операциите от интерфейса:

* **void AddDoctor(Doctor d)** - **Добавя** лекар. Ако съществува **лекар със същото име**, **хвърлете** **ArgumentException().**
* **void AddPatient(Doctor d, Patient p)** - **Добавя** пациент на лекар. Ако **лекарят не съществува**, **хвърлете** **ArgumentException()**.
* **IEnumerable<Doctor> GetDoctors()** - връща всички добавени лекари. Ако няма добавени лекари, върнете **празна колекция**.
* **IEnumerable<Patient> GetPatients()** - връща всички добавени пациенти. Ако няма добавени пациенти, върнете **празна колекция**.
* **bool Exist(Doctor d)** - връща отговор дали **лекарят** е **добавен**.
* **bool Exist(Patient p)** - връща отговор дали **пациентът** е **добавен**.
* **Doctor RemoveDoctor(string name)** - **премахва лекар** по **дадено име** и всичките му **пациенти**. Ако лекарят не съществува, **хвърлете ArgumentException()**.
* **void ChangeDoctor(Doctor from, Doctor to, Patient p)** - премества пациент от един лекар на друг. Ако някои от **предоставените обекти** не съществува, **хвърлете** **ArgumentException()**.
* **IEnumerable<Doctor> GetDoctorsByPopularity(int popularity)** – връща **всички лекари**, които имат такава **популярност**. Ако няма такива лекари, върнете **празна колекция**.
* **IEnumerable<Patient> GetPatientsByTown(string town)** - връща всички пациенти, които живеят в този град. Ако няма такива пациенти, върнете **празна колекция**.
* **IEnumerable<Patient> GetPatientsInAgeRange(int lo, int hi)** - връща всички пациенти с възраст, която е **по-голяма или равна** на **lo** и е **по-малка или равна** на **hi**. Ако няма такива пациенти, върнете **празна колекция**.
* **IEnumerable<Doctor> GetDoctorsSortedByPatientsCountDescAndNameAsc()** - връща всички лекари, сортирани по брой на пациенти в низходящ ред и след това по възходящ ред по име. Ако няма такива лекари, върнете **празна колекция**.
* **IEnumerable<Patient>GetPatientsSortedByDoctorsPopularityAscThenByHeightDescThenByAge()** - връща всички пациенти, сортирани по възходящ ред по популярност на техния лекар. След това по низходящ ред по височина на пациента и накрая по възходящ ред по възраст на пациента. Ако няма такива пациенти, върнете **празна колекция**.

## Vacc Ops – Производителност

За тази задача ще трябва да предадете само кода от предишната задача. Ако имате проблем с тази задача, трябва да извършите **подробен алгоритмичен анализ на сложността** и да опитате **да откриете слабите места** във вашата имплементация.

За тази задача е важно другите операции да са **имплементирани правилно** според специфичните проблеми: **добавяне**, **размер**, **премахване**, **взимане** и други...

Можете да предадете кода за тази задача **без пълно покритие от предишната задача**, няма да се разглеждат **всички тестови случай**, само **общото поведение** ще е важно, като крайните случай, предизвикващи изключения и други ще бъдат пренебрегнати...

## Управление

Създайте програма, която да управлява низове във файл **"file.txt"** и да чете командите от файла **"input.txt"**. Програмата трябва да поддържа следните операции и да отпечатва **exception-и**:

* **"1:{низ}:"**:
  + Програмата трябва да добави низ във файла. Ако вече съществува такъв низ, трябва да хвърлите **InvalidOperationException(„Името вече същестува“)**
* **"2:{низ}"**:
  + Програмата трябва да премахни низ от файла. Ако не съществува, трябва да хвърлите **InvalidOperationException(„Името не същестува“)**
* **"****3:{стар низ}|{нов низ}"**:
  + Програмата трябва да замени **стария низ** с **новия**. Ако няма такъв низ, трябва да хвърлете **InvalidOperationException(„Името не същестува“)**
* **"****4"**:
  + Програмата трябва да отпечата съдържанието на файла във формата:
  + **Файлът съдържа**
  + **-- {низ1}**
  + **-- {низ2}**
  + **-- {низ3}**
* **"5"**:
  + Програмата трябва да спре.

## Иван и забравеният алгоритъм

Иван е страхотен програмист, но този път се е загубил в своите мисли и не може да се сети как работи алгоритъмът за двоично търсене в сортиран масив. Той има задача да намери определен елемент в такъв масив, но не може да разбере как да го направи. Вашата задача е да му помогнете и да напишете програма, която да извърши двоично търсене в сортиран масив.

### Вход

* На първия ред ще получите сортирания масив. След това ще получите елемента, който трябва да намерите.

### Изход

* Отпечатайте неговия индекс.

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| 1 2 3 4 5  1 | 0 | Индекс на 1 е 0 |
| -1 0 1 2 4  1 | 2 | Индекс на 1 е 2 |
| 1 2 3 4 5  6 | -1 | В масива няма 6 |