Heroes

Preparation

Download the skeleton provided in Judge. **Do not** change the **StartUp** class or its **namespace**.

Problem description

Your task is to create a repository which stores hero by creating the classes described below.

First, write a C# class **Item** with the following properties:

- **Strength: int**
- **Ability: int**
- **Intelligence: int**

The class **constructor** should receive **strength, ability and intelligence** and override the **ToString()** method in the following format:

"Item:"

" * Strength: {Strength Value}"

" * Ability: {Ability Value}"

" * Intelligence: {Intelligence Value}"

Next, write a C# class **Hero** with the following properties:

- **Name: string**
- **Level: int**
- **Item: Item**

The class **constructor** should receive **name, level and item** and override the **ToString()** method in the following format:

"Hero: {Name} – {Level}lvl"

"Item:"

" * Strength: {Strength Value}"

" * Ability: {Ability Value}"

" * Intelligence: {Intelligence Value}"

Write a C# class **HeroRepository** that has **data** (a collection which stores the entity **Hero**). All entities inside the repository have the **same properties**.

```
public class HeroRepository
{
   // TODO: implement this class
}
```

The class **constructor** should initialize the **data** with a new instance of the collection**.** Implement the following features:

- Field **data** – **collection** that holds added heroes
- Method **Add(Hero hero)** – adds an entity to the data.
- Method **Remove(string name)** – removes an entity by given hero name.
- Method **GetHeroWithHighestStrength()** – returns the Hero which poses the item with the highest stength.
- Method **GetHeroWithHighestAbility()** – returns the Hero which poses the item with the highest ability.
- Method **GetHeroWithHighestIntelligence()** – returns the Hero which poses the item with the highest intellgence.
- Getter **Count** – returns the number of stored heroes.
- Override **ToString()** – Print all the heroes.

Constraints

- The names of the heroes will be always unique.
- The items of the heroes will always be with positive values.
- The items of the heroes will always be different.
- You will always have an item with the highest strength, ability and intelligence.

Examples

This is an example how the **HeroRepository** class is **intended to be used**.

Sample code usage

```
//Initialize the repository
HeroRepository repository = new HeroRepository();
//Initialize entity
Item item = new Item(23, 35, 48);
//Print Item
Console.WriteLine(item);

//Item:
//    * Strength: 23
//    * Ability: 35
//    * Intelligence: 48

//Initialize entity
Hero hero = new Hero("Hero Name", 24, item);
//Print Hero
Console.WriteLine(hero);

//Hero: Hero Name - 24lvl
//Item:
//    * Strength: 23
//    * Ability: 35
//    * Intelligence: 48

//Add Hero
repository.Add(hero);
//Remove Hero
repository.Remove("Hero Name");
```

```
Item secondItem = new Item(100, 20, 13);
Hero secondHero = new Hero("Second Hero Name", 125, secondItem);

//Add Heroes
repository.Add(hero);
repository.Add(secondHero);

Hero heroStrength = repository.GetHeroWithHighestStrength(); // Hero with name Second Hero
Hero heroAbility = repository.GetHeroWithHighestAbility(); // Hero with name Hero Name
Hero heroIntelligence = repository.GetHeroWithHighestIntelligence(); // Hero with name Hero

Console.WriteLine(repository.Count); //2

Console.WriteLine(repository);
//Hero: Hero Name - 24lvl
//Item:
//*Strength: 23
//    * Ability: 35
//    * Intelligence: 48
//Hero: Second Hero Name - 125lvl
//Item:
//    * Strength: 100
//    * Ability: 20
//    * Intelligence: 13
```

Submission

Zip all the files in the project folder except **bin** and **obj** folders