**Parking**

*Parking games are also among the popular games. Let's create one.*

* **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 items by creating the classes described below.

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

* **Manufacturer: string**
* **Model: string**
* **Year: int**

The class **constructor** should receive **manufacturer, model** and **year** and override the **ToString()** method in the following format:

**"{manufacturer} {model} ({year})"**

**Next**, write a C# class **Parking** that has **data** (a collection, which stores the entity **Car**). All entities inside the repository have the **same properties**. Also, the Parking class should have those properties:

* **Type: string**
* **Capacity: int**

The class **constructor** should receive **type** and **capacity**, also it should initialize the **data** with a new instance of the collection**.** Implement the following features:

* Field **data** – **collection** that holds added cars
* Method **Add(Car car)** – **adds** an **entity** to the data **if** **there** **is** an **empty cell** for the car.
* Method **Remove(string manufacturer, string model)** – removes the car by **given manufacturer and model,** if such **exists**, and **returns bool**.
* Method **GetLatestCar()** – returns the **latest** car (by year) or **null** if have no cars.
* Method **GetCar(string manufacturer, string model)** – returns the car with the **given manufacturer** and **model** or null if there is no such car.
* Getter **Count** – **returns** the **number** of cars.
* **GetStatistics()** – **returns** a **string** in the following **format**:
* **"The cars are parked in {parking type}:  
  {Car1}  
  {Car2}  
  (…)**"
* **Constraints**
* The **combinations** of **manufacturers** and **models** will be **always unique**.
* The **year** of the cars will always be **positive**.
* There won't be cars with the same years.
* **Examples**

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

|  |
| --- |
| **Sample code usage** |
| // Initialize the repository  Parking parking = new Parking("Underground parking garage", 5);  // Initialize entity  Car volvo = new Car("Volvo", "XC70", 2010);  // Print Car  Console.WriteLine(volvo); // Volvo XC70 (2010)  // Add Car  parking.Add(volvo);  // Remove Car  Console.WriteLine(parking.Remove("Volvo", "XC90")); // False  Console.WriteLine(parking.Remove("Volvo", "XC70")); // True  Car peugeot = new Car("Peugeot", "307", 2011);  Car audi = new Car("Audi", "S4", 2005);  parking.Add(peugeot);  parking.Add(audi);  // Get Latest Car  Car latestCar = parking.GetLatestCar();  Console.WriteLine(latestCar); // Peugeot 307 (2011)  // Get Car  Car audiS4 = parking.GetCar("Audi", "S4");  Console.WriteLine(audiS4); // Audi S4 (2005)  // Count  Console.WriteLine(parking.Count); // 2  // Get Statistics  Console.WriteLine(parking.GetStatistics());  // The cars are parked in Underground parking garage:  // Peugeot 307(2011)  // Audi S4(2005) |

* **Submission**

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