# Santa's Bag of Presents

# A picture containing LEGO, vector graphics Description automatically generated

*Can you imagine Santa without his special bag that carries presents for so many children? Neither can we. So let's have a quick sneak peek inside it and help Santa rearrange it.*

## 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 presents by creating the classes described below.

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

* **Name: string**
* **Weight: double**
* **Gender: string**

The class **constructor** should receive **name, weight and gender**. Override the **ToString()** method in the following format:

**"****Present {name} ({weight}) for a {gender}"**

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

* **Color: string**
* **Capacity: int**

The class **constructor** should receive **color** 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 presents
* Method Add(Present present) - **adds** an **entity** to the data **if** **there** **is** **room** for it
* Method Remove(string name) - removes a present by **given name,** if such **exists**, and **returns bool**
* Method GetHeaviestPresent() - **returns the heaviest present**
* Method GetPresent(string name) - **returns** the **present** with the **given name**
* Getter Count - **returns** the **number** of presents
* **Report()** - **returns** a **string** in the following **format** (print the presents in **order of appearance**):
  + **"{color of Bag} bag contains:  
    {Present1}  
    {Present2}  
    (…)**"

## Constraints

* The **names** of the presents will be **always unique**.
* You will always have a present added before receiving methods manipulating the Bag’s presents.

## Examples

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

|  |
| --- |
| Sample code usage |
| //Initialize the repository (Bag)  Bag bag = new Bag("Blue", 20);  //Initialize entity  Present present = new Present("Train", 0.4, "Boy");  //Print Present  Console.WriteLine(present); // Present Train for a Boy  //Add Present  bag.Add(present);  Console.WriteLine(bag.Count); //1  //Remove Present  bag.Remove("Doll"); //false  Present secondPresent = new Present("Doll", 0.7, "Girl");  //Add Present  bag.Add(secondPresent);  //Get heaviest present  Present heaviestPresent = bag.GetHeaviestPresent(); // Present Doll for a Girl  //Get present  Present searchedPresent = bag.GetPresent("Train"); // Present Train for a Boy  Console.WriteLine(bag.Report());  // Blue bag contains:  // Present Train for a Boy  // Present Doll for a Girl |

## Submission

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