# Cocktail Party

*You and your friends can’t wait for the bars to open so you decide to get together for a cocktail party and try your skills as bartenders. You develop a program that ensures all cocktails recipes will meet your high standards.*

## Preparation

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

## Problem description

Your task is to create a cocktail that is made of different ingredients.

### Ingredient

First, write a C# **class**, called **Ingredient** with properties:

* **Name: string**
* **Alcohol: int**
* **Quantity: int**

The **constructor** of **Ingredient** class should receive **name, alcohol and quantity**.

The class should also have the following methods:

* Override **ToString()** method in the format:

**"****Ingredient: {Name}**

**Quantity: {Quantity}**

**Alcohol: {Alcohol}"**

### Cocktail

Next step is to write **Cocktail** class that has a **collection** of type **Ingredient** with corresponding **unique** **name** of an Ingredient. The name of the collection should be **Ingredients**. All the entities of the **Ingredients** collection have the **same** properties. The **Cocktail** has also some additional properties:

* **Name: string**
* **Capacity: int -** the maximum allowed number of ingredients in the cocktail
* **MaxAlcoholLevel: int -** the maximum allowed amount of alcohol in the cocktail

The **constructor** of the Cocktail class should receive **name, capacity** and **maxAlcoholLevel**.

Implement the coming features:

* Method **Add(Ingredient ingredient)** - adds the entity **if** there **isn't** an Ingredient with the same **name** and **if** there is enough space in terms of quantity and alcohol.
* Method **Remove(string name)** - removes an Ingredient from the cocktail with the given **name**, if such **exists** and returns **bool** if the deletion is successful.
* Method **FindIngredient(string name)** - returns an **Ingredient** with the given name. If it doesn't exist, return **null**.
* Method **GetMostAlcoholicIngredient() –** returns the **Ingredient** with most Alcohol.
* Getter **CurrentAlcoholLevel –** returns the amount of alcohol currently in the cocktail.
* Method **Report()** - returns information about the Cocktail and the Ingredients inside it in the following format:

**"****Cocktail: {name} - Current Alcohol Level: {CurrentAlcoholLevel}**

**{Ingredient1}**

**{Ingredient2}**

**… "**

## Constraints

* The name of each Ingredient in the pool will always be unique.
* Each Ingredient will have a different number of Alcohol.
* The Alcohol of an Ingredient and the MaxAlcoholLevel of the Cocktail will always be positive numbers.
* You will always be given an Ingredient added before receiving the method for its manipulation.

## Examples

|  |
| --- |
| Sample code usage |
| //Sample Code Usage:  //Initialize Cocktail  Cocktail cocktail = new Cocktail("Pina Colada", 3, 10);  //Initialize Ingredient  Ingredient rum = new Ingredient("Rum", 2, 3);  //Print rum  Console.WriteLine(rum.ToString());  //Ingredient: Rum  //Quantity: 3  //Alcohol: 2  //Add rum  cocktail.Add(rum);  //Remove rum  Console.WriteLine(cocktail.Remove("Rum")); // true  Ingredient vodka = new Ingredient("Vodka", 2, 5);  Ingredient milk = new Ingredient("Milk", 0, 5);  //Add ingredients  cocktail.Add(vodka);  cocktail.Add(milk);  //GetMostAlcoholicIngredient  Console.WriteLine(cocktail.GetMostAlcoholicIngredient());  //Ingredient: Vodka  //Quantity: 5  //Alcohol: 2  //CurrentAlcoholLevel  Console.WriteLine(cocktail.CurrentAlcoholLevel);  //2  //Print Cocktail report  Console.WriteLine(cocktail.Report());  //Cocktail: Pina Colada - Current Alcohol Level: 2  //Ingredient: Vodka  //Quantity: 5  //Alcohol: 2  //Ingredient: Milk  //Quantity: 5  //Alcohol: 0 |

## Submission

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