# Problem 1. Vapor Winter Sale

*Our favorite gaming platform, Vapor, has started its yearly winter sale. Stamat wants to buy some games and he wants to make a list.*

Your task is to take **every game** and add it **with its price**. The games **may have DLCs** (downloadable content), which you should also add. The games will be given in the following format **"{game}-{price}"**. The game names will contain only letters, digits and spaces. The DLCs will be given in the following format **"{game}:{DLC}"**. **All the of the inputs** will be divided by **", "**. You should only add the DLC **if the game already exists**. The DLC **increases** the game price by **20%**.

After you have processed all the games you should **lower the prices** of all the games by **20%** **if they don't have a DLC** and by **50%** **if they do**. After this you should print the games **with DLCs**, **ordered ascending** **by price**, and then all the games **without DLCs**, **ordered descending by price**, in the following format:

* If the game **has** a DLC:
  + **"{game} - {DLC} - {price}"**
* If the game **doesn't have** a DLC:
  + **"{game} - {price}"**

The prices should be formatted to the second decimal place.

## Input

One **string** with inputs, divided by **", "**. Each input of the string will be in one of the following formats – **"{game}-{price}"** or **"{game}:{DLC}"**.

You don't have to check the input, it will always be valid.

## Output

**"{game1} - {DLC} - {price}"**

**"{game2} - {DLC} - {price}"**

**...**

**"{game n} - {price}"**

**"{game n + 1} - {price}"**

**...**

## Examples

| **Input** | **Output** |
| --- | --- |
| WitHer 3-50, FullLife 3-60, WitHer 3:Blood and Beer, Cyberfunk 2077-120, League of Leg Ends-10, League of Leg Ends:DoaT | League of Leg Ends - DoaT - 6.00  WitHer 3 - Blood and Beer - 30.00  Cyberfunk 2077 - 96.00  FullLife 3 - 48.00 |
| Center Strike-14.99, FortLite-25, BattleShield 5-64.74, BattleShield 5:CoD edition, Dog of War-45, Dead Red Redemption-100, Dead Red Redemption:no DLC | BattleShield 5 - CoD edition - 38.84  Dead Red Redemption - no DLC - 60.00  Dog of War - 36.00  FortLite - 20.00  Center Strike - 11.99 |

**Problem 02. Activation Keys**

*Stamat got his games from the winter sale, but the keys he got to activate them do not work. He needs to make them in the correct format in order to get his games.*

You will receive **one line** with the keys of the games, divided by **"&"**. You will have to see if the keys are valid. A valid key contains only **numbers**, **letters** and is **16 or 25 symbols long**.

In order to fix the keys, you first have to **put dashes** (**'-'**) in them. If the key is **16 symbols long**, you should divide it into **four groups of four symbols**. If the key is **25 symbols long**, you have to divide it in **five groups of five symbols**. Then you have to make **all the letters** in the key **uppercase**. The last thing you have to do is to take **every digit** from the key, **subtract it from nine (9)**, and replace it with the new digit you have.

In the end you have to print all the keys, joined by **", "**.

**Input**

One line with all the keys, divided by **"&"**.

**Output**

One line with all the valid keys, joined by **", "**.

**Examples**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comment** |
| t1kjZU764zIME6Dl9ryD0g1U8&-P4\*(`Q>:x8\yE1{({X/Hoq!gR.&rg93bXgkmILW188m&KroGf1prUdxdA4ln&U3WH9kXPY0SncCfs | T8KJZ-U235Z-IME3D-L0RYD-9G8U1, RG06-BXGK-MILW-811M, KROG-F8PR-UDXD-A5LN, U6WH-0KXP-Y9SN-CCFS | First you find the dividers '&' and then you take all the valid inputs (colored in yellow) |
| xPt8VYhUDalilWLvb6uMSGEEf&KWQ{R.@/HZCbbV++1o]V+oG@@fF^93&y6fT5EGFgZHqlFiS | XPT1V-YHUDA-LILWL-VB3UM-SGEEF, Y3FT-4EGF-GZHQ-LFIS |  |

**03. Basic CRUD – Game Store**

You have been tasked to create a simple application for a Game Store. The application should hold **games**, which are the main app **entities**. The app is called **GameStore**.

The functionality of the application should support **creating**, **listing, editing**, **deleting** games.

The application should **persist** the data into a **database**.

**Overview**

Your application should be built on **one** of the **following technologies**:

**JavaScript**

* **NodeJS** + **ExpressJS** frameworks
* **Handlebars.js** view engine
* **Mongoose** ORM
* **MongoDB**

**PHP**

* **Symfony** framework
* **Twig** view engine
* **Doctrine** ORM
* **MySQL** database

**Java**

* **Spring** framework (**Spring MVC** + **Spring Boot** + **Spring Data**)
* **Thymeleaf** view engine
* **JPA** / **Hibernate ORM** + **Spring Data** data access
* **MySQL** database

**C#**

* **ASP.NET Core** framework (**ASP.NET MVC** + **Entity Framework Core**)
* **Razor** view engine
* **Entity Framework Core** ORM
* **SQL Server** database

**Data Model**

The **Game** entity holds **5 properties**:

* **id – technology-dependent identifier (ObjectID for JavaScript, int for all other technologies)**
* **name** **– non-empty text**
* **dlc** **– non-empty text**
* **price** **– non-null floating-point number**
* **platform** – non empty text

**Project Skeletons**

You will be given the applications’ **skeletons**, which holds about **90%** of the logic. You’ll be given some **files** (**controllers**, **models** etc.). The files will have **partially implemented logic**, so you’ll need to write some code for the application to **function properly**.

The application’s views will be given to you fully implemented. You only need to include them in your business logic.

**Each technology** will have its **own skeleton**, and the **different** **skeletons** may **differ** in **terms** of **what is given to you** and **what is to be implemented**.

Everything that has been given to you inside the skeleton is **correctly implemented** and if you write your code **correctly**, the application should work just fine. You are free to change anything in the Skeleton on your account.

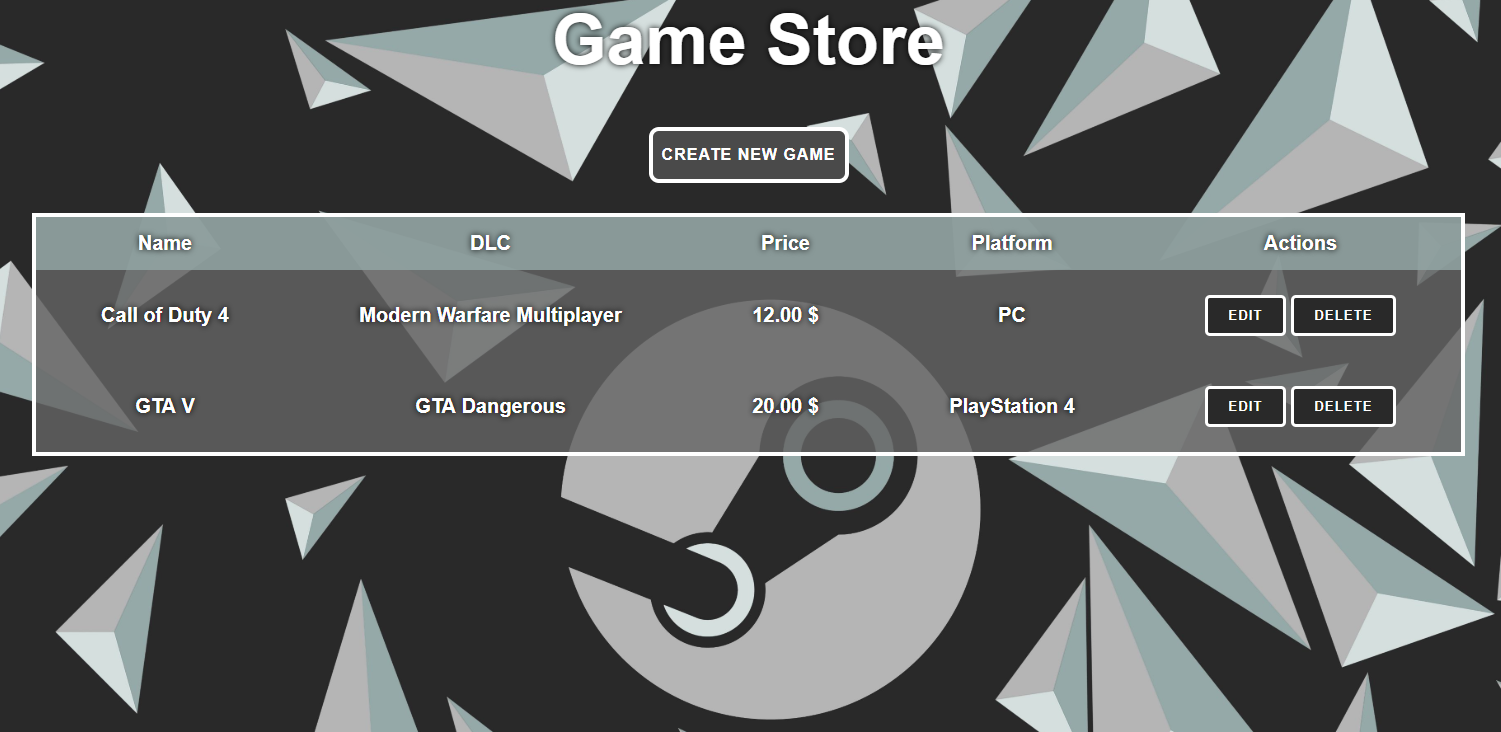
**User Interface**

This is the user interface or how the application’s pages should look in their final form (fully implemented). You have several pages, described below:

**Index Page**

**Route: "/"**

Displays **all** the **games** from the database with **options** to **edit** or **delete** them.



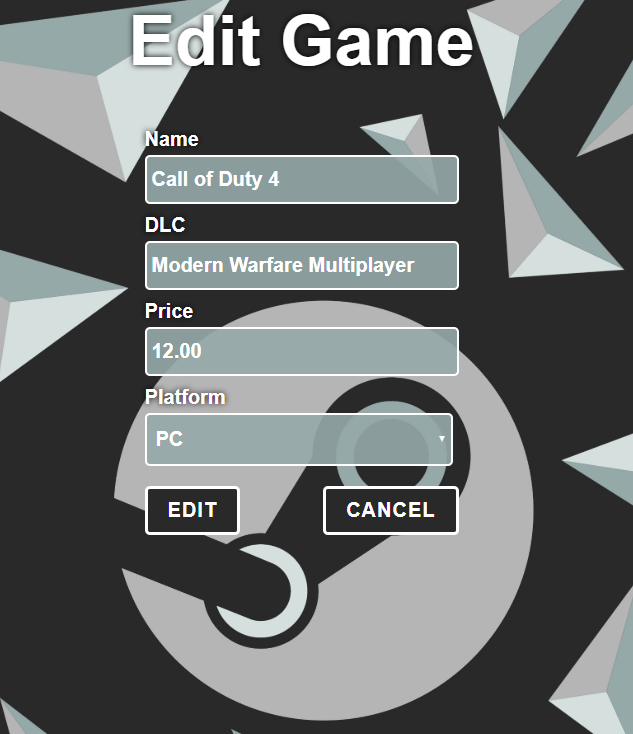
**Create Page**

**Route: "/create"**



**Edit Page**

**Route: "/edit/{id}"**



**Delete Page**

**Route: "/delete/{id}"**



**Problem**

As you can see the different pages are on different routes. Most of the routing logic will be given to you in the **Skeleton**, but you should make sure that the application **works properly**.

Implement the "**Game Store" app** using only **your technology.**

**Setup**

Before you start working, make sure you **download all the dependencies** (packages) required for your technology and **set up** the **databases**! Below are instructions on how to do this:

**PHP and Symfony**

1. Make sure you've started your **MySQL server** (either from **XAMPP** or standalone)
2. Open a **Terminal in PHPStorm** or **shell** / **command prompt** / **PowerShell** window in the **root directory**: [Shift] + [Right click] 🡺 [Open command window here]
3. Enter the "**composer install**" command to restore its **Composer dependencies**   
   (described in **composer.json**)
4. Enter the "**php bin/console doctrine:database:create --if-not-exists**" command
5. Done!

**JavaScript and Node.js**

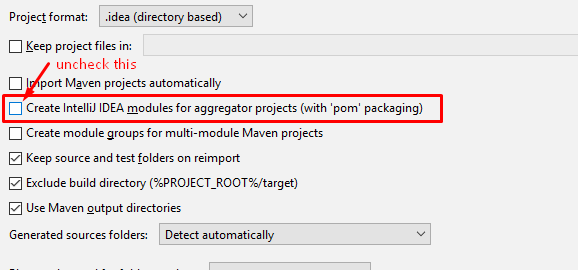
1. Go into the **root directory** of the project (where the **index.js file is**)
2. Open a **shell** / **command prompt** / **PowerShell** window in the **root directory**: [Shift] + [Right click] 🡺 [Open command window here]
3. Enter the “**npm install**” command to restore its **Node.js dependencies** (described in **package.json**)
4. Type **node index.js** to start the server
5. Done!

**C# and ASP.NET**

The C# project will automatically resolve its **NuGet dependencies** (described in **packages.config**) using the NuGet package restore when the project is built.

**Java and Spring MVC**

When you import your project, you should **uncheck "**Create IntelliJ IDEA modules for aggregator projects (with 'pom' packaging)**"**:



This project is **set up to use Java jdk 1.8.** If your version is different, you can change it in **Maven dependencies** like this:



The Java project will automatically resolve its **Maven dependencies** (described in **pom.xml**) when the project is built.