# Exam Preparation I

Submit your solutions here: <https://alpha.judge.softuni.org/contests/exam-preparation-i/4512>

## **1. Unit Test: Dictionary Intersection**

Test a given method which takes in **2 <string, int> dictionaries** and **finds the intersection of the two**.

The method is found in the DictionaryIntersection.cs file:

A computer screen shot of text

Description automatically generated

You are given a **test** **file** DictionaryIntersectionTests.cs containing **5 empty tests**. Implement all tests:

A screenshot of a computer

Description automatically generated

When you are ready make sure your **tests run:**

A screenshot of a computer

Description automatically generated

**IMPORTANT:** **DO NOT REMOVE OR CHANGE ANY NAMESPACES AND USINGS.**

## **2. Unit Test: Product**

You are given a **folder of 2 classes -** ProductandProductInventory. The Product **class** is just a helper class:

A screenshot of a computer code

Description automatically generated

The ProductInventory class holds a **list of products** and **methods** for **using the list** that you will **test**:

A screen shot of a computer code

Description automatically generated

A screen shot of a computer code

Description automatically generated

A white background with text and symbols

Description automatically generated with medium confidence

You will need to use the test file ProductInventoryTests.cs, inside they are **5 empty tests with a setup method:**

A screenshot of a computer program

Description automatically generated

When you are ready make sure your **tests run:**

A screenshot of a computer

Description automatically generated

**IMPORTANT:** **DO NOT REMOVE OR CHANGE ANY NAMESPACES AND USINGS.**

**3. JS Unit Testing**

After opening the project, run **npm install** in the console.

**Your Task**

Using **Mocha** and **Chai** write **JS Unit Tests** to test a variable named **findNewApartment**, which represents an object. You may use the following code as a template:

|  |
| --- |
| describe(**"*Tests* …"**, **function**() {  describe(**"*TODO* …"**, **function**() {    ***it***(**"*TODO …*"**, **function**() {  *//* ***TODO:*** …  });  });    *//* ***TODO:*** …  }); |

The object that should have the following functionality:

* **isGoodLocation (city, nearPublicTransportation) -** A function that accepts two parameters: **string** and **boolean**.
* If the value of the string **city** is different than a "**Sofia**", "**Plovdiv**" or "**Varna**"

return :

**"This location is not suitable for you."**

* If the value of the boolean **nearPublicTransportation** is **false**

return :

**"There is no public transport in area."**

* Otherwise, if the above conditions are not met, **return** the following message:
* **"You can go on home tour!"**
* You need to validate the input, if the **city** and **nearPublicTransportation** are not a **string** and **boolean**, **throw** an error: "**Invalid input!**".
* **isLargeEnough (apartments, minimalSquareMeters) -** A function that accepts an **array** and **number**.
  + The **apartments** array will store the area of the apartment in square meters ([40, 50, 60…])
  + You must **add** the area of apartment in **resultArr** if is **equal** or **bigger** than **minimalSquareMeters**.
* Finally, **return** the changed array of apartments.
  + There is a need for validation for the input, an **array** and **number** may not always be valid. In case of submitted **invalid** parameters, **throw** an error "**Invalid input!**":
    - If passed **apartments** parameteris not an array.
    - If **apartments** is empty **array.**
    - If the **minimalSquareMeters** is not a number.
* **isItAffordable (price, budget) -** A function that accepts two parameters: **number** and **number**.
* You need to **calculate** if you can afford buying the apartment by **subtracting** the **price** of the apartment from your **budget**.
* If the **result** is lower than **0,** return**:**

**"You don't have enough money for this house!"**

* Otherwise, if the above conditions are not met, **return** the following message:

**"You can afford this home!"**

* You need to validate the input, if the **price** and **budget** are not a **number** and **price** and **budget** are **less** or **equal** to0**, throw** an error: "**Invalid input!**".

**JS Code**

To ease you in the process, you are provided with an implementation that meets all of the specification requirements for the **findNewApartment** object.

**Submission**

Submit your tests inside a **describe()** statement, as shown above.