**Bonus: Error Handling**

## Time Span

Create a JS class TimeSpan that holds a seconds property. The constructor takes three numeric parameters for hours, minutes and seconds that initialize the value of the internal property. The class must work with negative values and overflow values (e.g. more than 59 seconds), but throws a RangeError if any of the parameters is invalid with the following error message:

Invalid {hours/minutes/seconds}: {parameter value}

Replace the parts in curly braces with the relevant value. See the examples for details.

The class must have a toString() method that prints a string with the held timespan in format 'hh:mm:ss'. The minutes and seconds must have leading zeros if they are less than 10.

### Input / Output

The class constructor must take three parameters. As output of the toString() method **return** a string.

### Examples

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| console.log('' + new TimeSpan(7, 8, 5)); | 7:08:05 |
| console.log('' + new TimeSpan(12, 76, -5)); | 13:15:55 |
| console.log('' + new TimeSpan('', 2.5, {})); | RangeError: Invalid hours: |
| console.log('' + new TimeSpan(3, 2.5, {})); | RangeError: Invalid minutes: 2.5 |
| console.log('' + new TimeSpan(3, 2, {})); | RangeError: Invalid seconds: [object Object] |

## Parse Data

Create a JS program that defines a class Candy and a function that receives an array of recipes and generates class instances from them. The **Candy** has topping, filling and spice, all set trough the constructor as strings. The recipes will be an array of strings in format:

{topping}:{filling1},{filling2},{…}:{spice}

There can be any number of comma-separated **fillings** in the recipe and zero or one type of **spice**. If one of the parameters is not specified (it is empty string) set it to null. If this format is not followed, consider the recipe invalid. Each instance of **Candy** must have valid initial state, but not all recipes will be correct and the constructor must throw a TypeError if the following requirements aren’t met:

topping – must be either '**milk chocolate**', '**white chocolate**' or '**dark chocolate**'; any other topping, including null is invalid

filling – can be null or a combination of up to three of the following: **'hazelnut'**, **'caramel'**, **'strawberry'**, **'blueberry'**, **'yogurt'**, **'fudge'**; if more than 3 fillings are specified or the filling is not in the list, the recipe is invalid

spice – can be null or any string except **'poison'** or **'asbestos'**

As a result, return an array containing all valid instances – if an instance was created with invalid parameters, do not include it in the final array. Do not display any errors as you encounter them.

### Input / Output

Your function will receive an array of string as input. As output, **return** an array of valid objects. Do not display any errors.

Submit a function that receives one parameter and holds the class definition inside it. The properties of the class must be named exactly as described (either the data property, or the getter).

### Examples

|  |
| --- |
| Sample Input |
| makeCandy([  'milk chocolate:hazelnut,caramel:pumpkin',  'dark chocolate::chips',  'white chocolate::poison', *// invalid*  'white chocolate:fudge:',  'frosting:yogurt:frosting', *// invalid*  'dark chocolate:blueberry:rock crystals'  ]) |
| Output |
| [ Candy {  topping: 'milk chocolate',  filling: 'hazelnut,caramel',  spice: 'pumpkin' },  Candy {  topping: 'dark chocolate',  filling: null,  spice: 'chips' },  Candy {  topping: 'white chocolate',  filling: 'fudge',  spice: null },  Candy {  topping: 'dark chocolate',  filling: 'blueberry',  spice: 'rock crystals' } ] |