More Exercises: Object Composition

1. Sorted List

Implement a collection, which keeps a list of numbers, sorted in **ascending order**. It must support the following functionality:

- **`add(element)`** - adds a new element to the collection
- **`remove(index)`** - removes the element at position **index**
- **`get(index)`** - returns the value of the element at position **index**
- **`size`** - number of elements stored in the collection

The **correct order** of the element must be kept **at all times**, regardless of which operation is called. **Removing** and **retrieving** elements **shouldn't** work if the provided index points **outside the length** of the collection (either throw an error or do nothing). Note the **size** of the collection is **NOT** a function. Write your code such that the first function in your solution **returns an instance** of your Sorted List.

Input / Output

All function that expect **input** as **parameters** will receive valid data. Any result expected from a function should be **returned** as it's result. Your **main function** should **return** an **object instance** with the required functionality as it's result.

2. Bug Tracker

Create a program for managing bug reports. It must perform as a self-contained module with exposed functionality. Whenever a new element is added, deleted or changed with a command, the HTML should be updated automatically. A bug report has the following structure:

```
{ ID: Number,
  author: String,
  description: String,
  reproducible: Boolean,
  severity: Number,
  status: String }
```

The **ID** of each report has to be a **unique** number, starting from **zero** and increasing **sequentially**. The module needs to implement the following **functions**:

- **`report(author, description, reproducible, severity)`** - **create** a new bug report and store it. The **ID** is assigned automatically to the next available number and the **status** defaults to **`'Open'`**
- **`setStatus(id, newStatus)`** - change the status of a bug registered in the system to **`newStatus`** by given **ID**
- **`remove(id)`** - delete a bug report by given **ID**
- **`sort(method)`** - change the order in which bug reports are displayed on the webpage. The **`method`** argument is a string and can be either **`'author'`**, **`'severity'`** or **`'ID'`**. Always sort in ascending order (default behavior for alphabetical sort). The default sorting method is by **`'ID'`**.
- **`output(selector)`** - set the HTML element inside which the result is to be displayed to **`selector`**

Use the following structure for each HTML report:

index.html

```
<div id="report_${ID}" class="report">
  <div class="body">
    <p>${description}</p>
  </div>
  <div class="title">
    <span class="author">Submitted by: ${author}</span>
    <span class="status">${status} | ${severity}</span>
  </div>
</div>
```

Input

Input will be passed to each applicable function as **parameters** in the correct format.

Output

Your solution must **expose** a **module** with all required functions bundled in it (**return** it as a result of your main function). The HTML should be **modified** as specified.