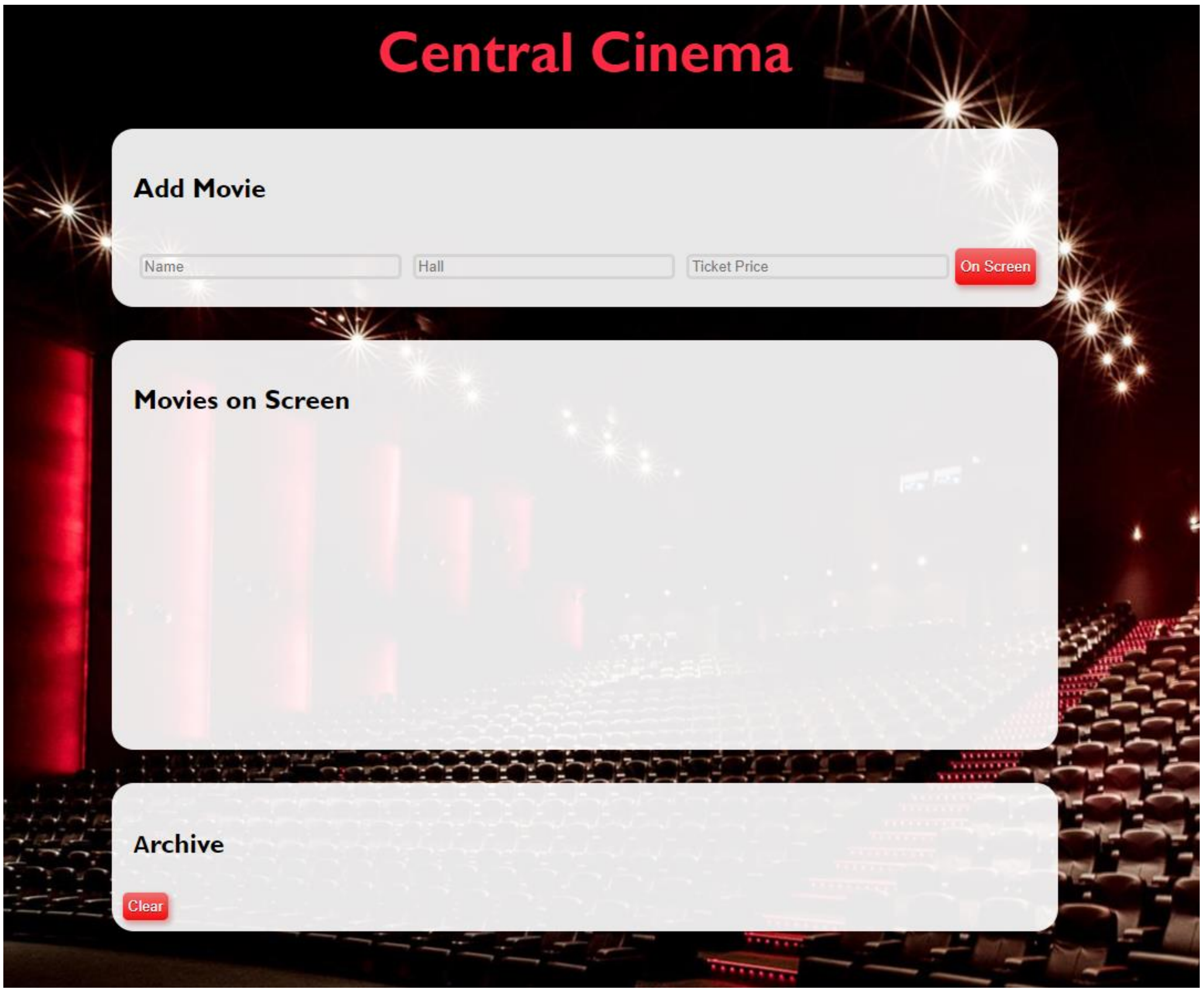
JS Advanced: Retake Exam Aug 2020

Exam problems for the "JavaScript Advanced" course @ SoftUni.

Problem 1. Central Cinema (DOM Manipulation)

Use the **given skeleton** to solve this problem.

Note: You have **NO permission** to change directly the given HTML *(index.html file)*.

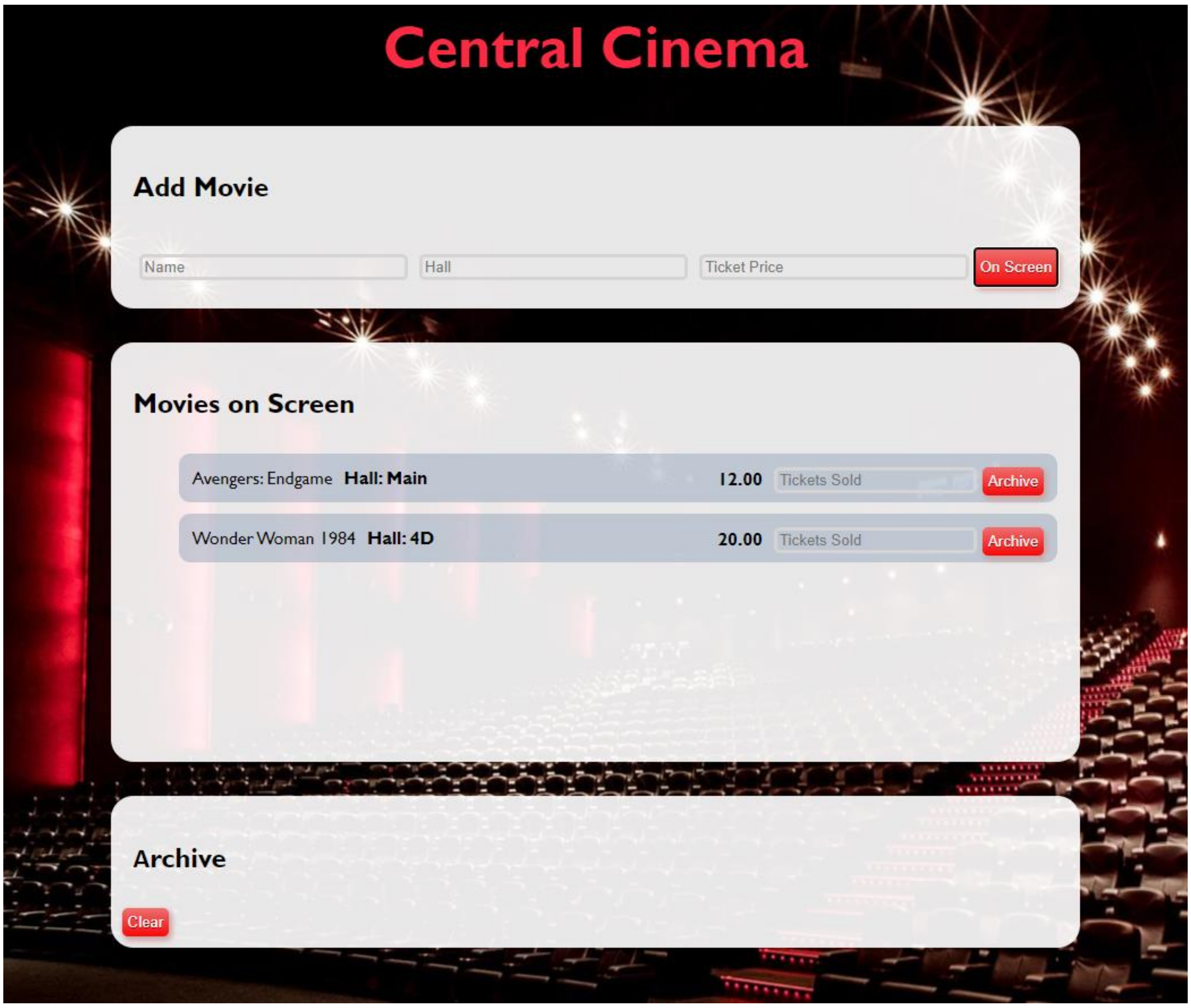


Your Task

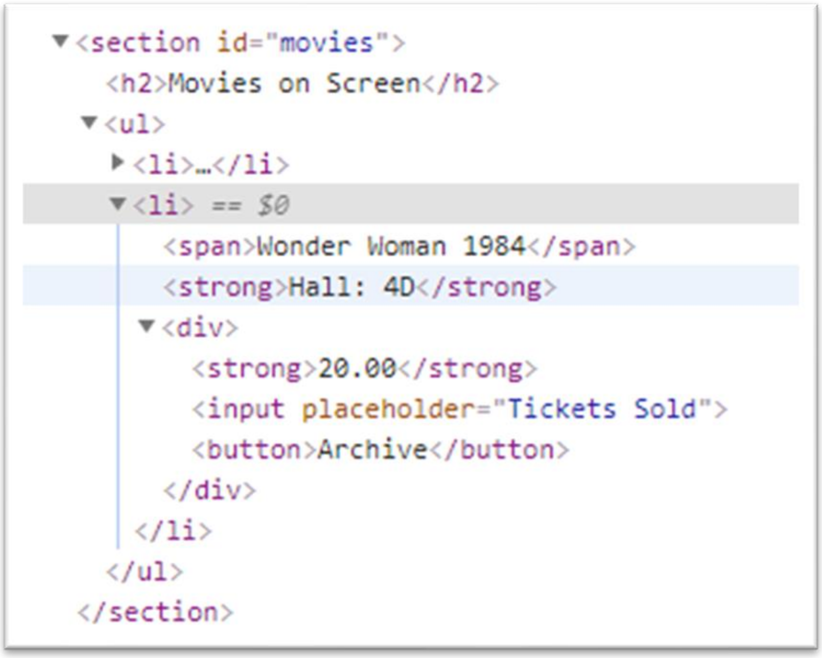
Write the missing JavaScript code to make the Central Cinema application work as expected.

Each movie have **Name**, **`Hall`** and **`Ticket Price`**.

When you click the **`[On Screen]`** button and **only** if **inputs** are all **filled** and the ticket **price value** is a **number**, then a new **list item** should be added to the **Movies on Screen** section. Clear the inputs.


Central Cinema
Add Movie
Name
Hall
Ticket Price
On Screen
Movies on Screen
Avengers: Endgame Hall: Main 12.00 Tickets Sold Archive
Wonder Woman 1984 Hall: 4D 20.00 Tickets Sold Archive
Archive
Clear

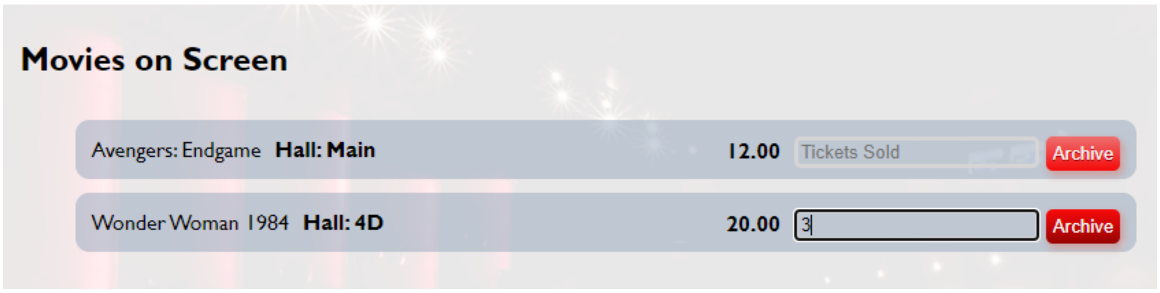
The new item should have the **following structure**:



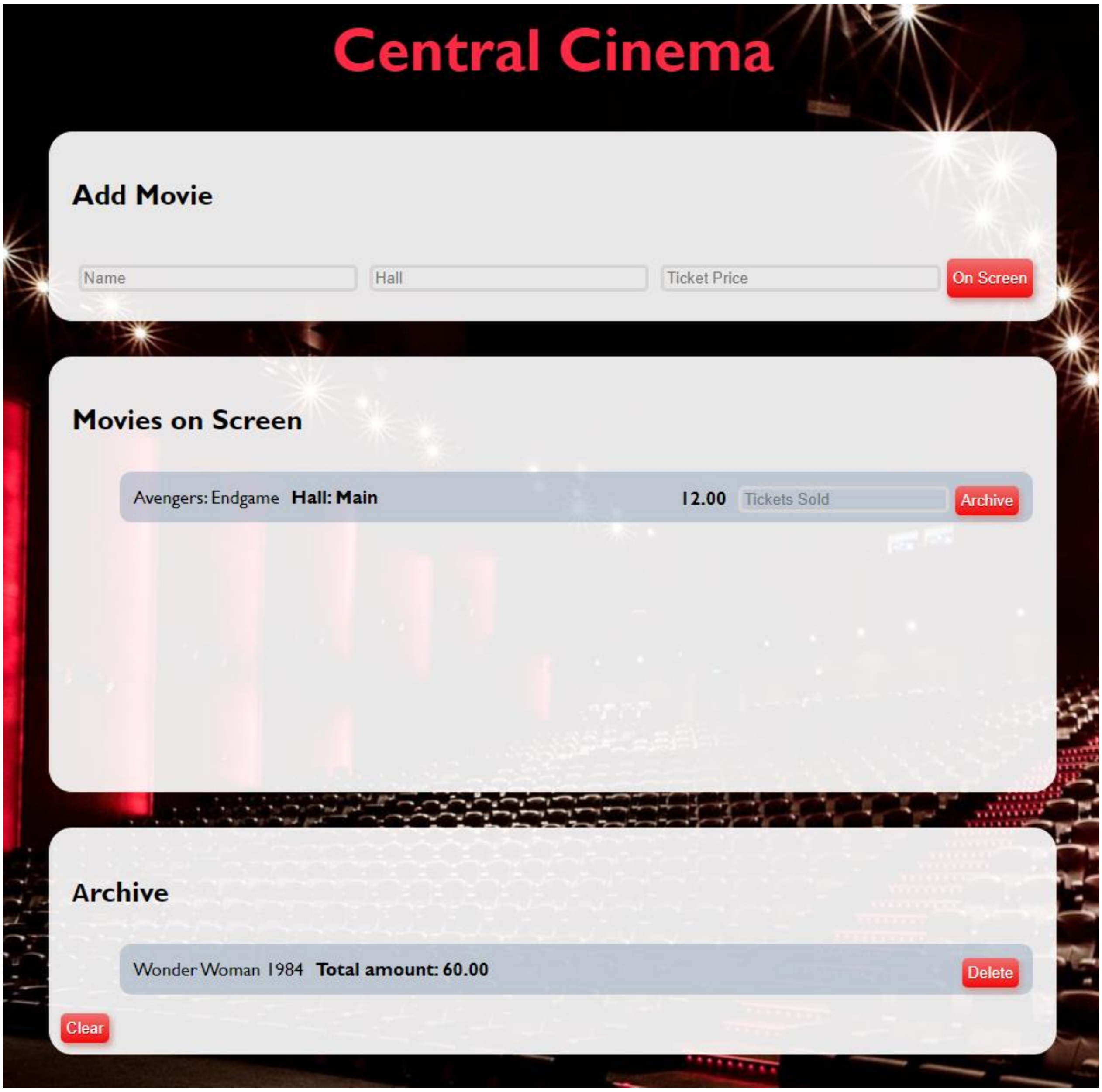
```
▼<section id="movies">
   <h2>Movies on Screen</h2>
  ▼<ul>
    ▶<li>…</li>
    ▼<li> == $0
        <span>Wonder Woman 1984</span>
        <strong>Hall: 4D</strong>
      ▼<div>
          <strong>20.00</strong>
          <input placeholder="Tickets Sold">
          <button>Archive</button>
        </div>
      </li>
   </ul>
 </section>
```

You should create a **`li`** element that contains **`span`** element with the name of the movie, a **`strong`** element with the name of the hall like **“`Hall: { hallName }`“** and a **`div`** element. Inside the **`div`** element, there are a **`strong`** element with the ticket price, **`input`** element with **`placeholder`** containing **“`Tickets Sold`”** and a button **`[Archive]`.**

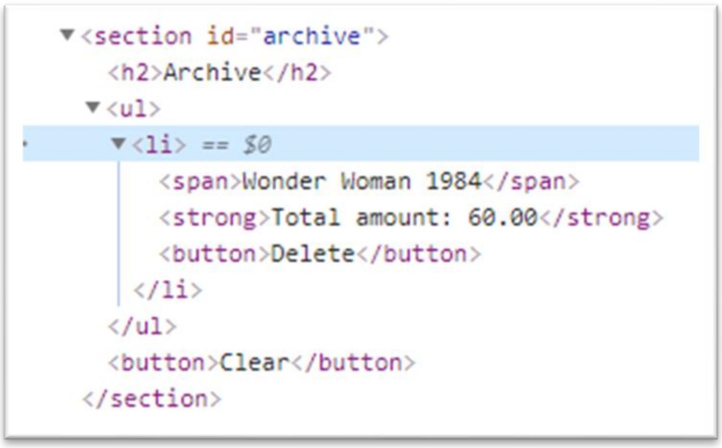
When you click the **`[Archive]`** button and **only** if the input for tickets count is **filled with a number** like:



You should **add** that item to **Archive** section like a list item in the **`ul`,** calculating the total profit of the movie like this:



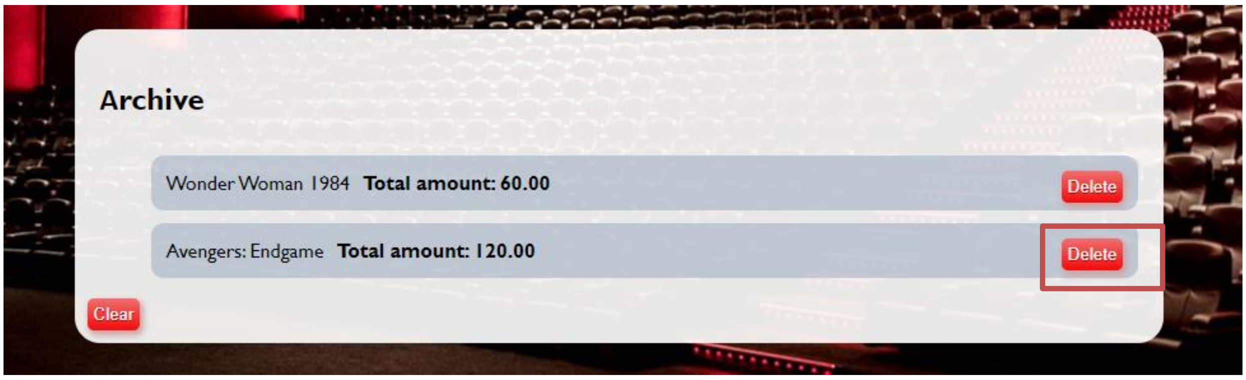
Use the following format:



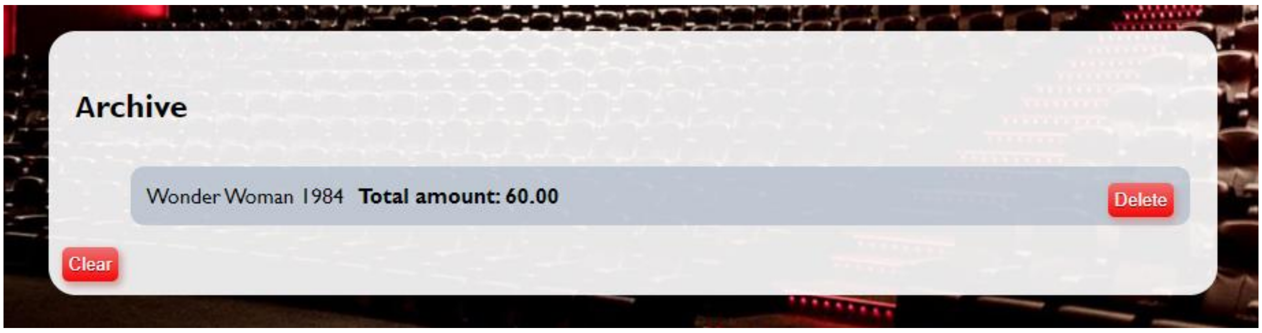
```
▼<section id="archive">
   <h2>Archive</h2>
  ▼<ul>
    ▼<li> == $0
       <span>Wonder Woman 1984</span>
       <strong>Total amount: 60.00</strong>
       <button>Delete</button>
     </li>
   </ul>
   <button>Clear</button>
 </section>
```

Here we have **list item** containing **`span`** element with the name of the movie, **`strong`** element with total profit like "**`Total amount: {total price}`**" **<u>fixed to the second digit</u>** after the decimal point. Add a delete button **`[Delete]`**.

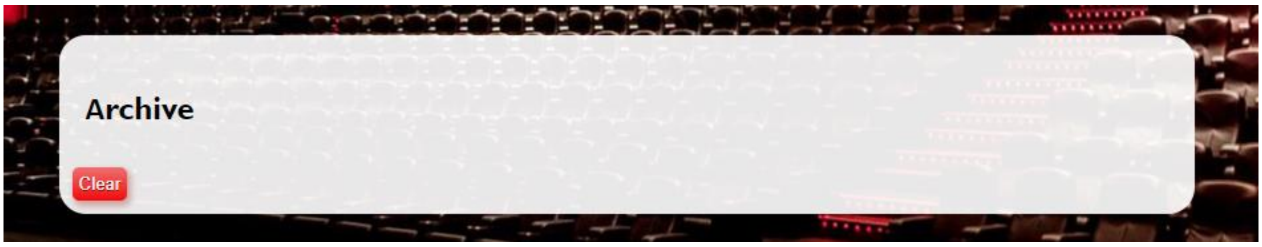
When you click the **`[Delete]`** button, you should **delete the current list item.**



Archive
Wonder Woman 1984 Total amount: 60.00
Delete
Avengers: Endgame Total amount: 120.00
Delete
Clear



Finally, when we click the **`[Clear]`** button **delete** all **`li`** elements from the **archive** section. No matter how many archived movies we have the archive section leaves like this:



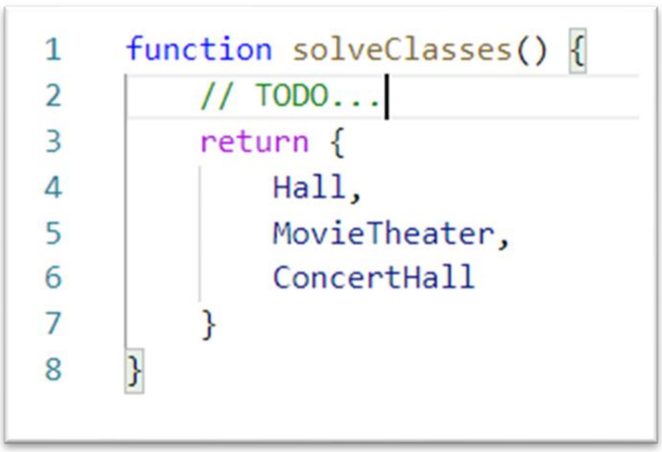
Problem 2. Halls

Your Task

Implement the following classes: **`Hall, MovieTheater, ConcertHall.`**

Submission

Submit your **`solveClasses`** function with the implementation of the tree classes.



```
function solveClasses() {
    // TODO...
    return {
        Hall,
        MovieTheater,
        ConcertHall
    }
}
```

Class Hall

constructor(capacity, name)

Should have these **3** properties:

- **`capacity – number;`**
- **`name –string;`**
- **`events - array;`**

hallEvent(title)

This **function** should receive single **`title`** like **string**, add it to the **`events`** array and **return** a message:

`"Event is added."`

If event is already added to the events array **throw an error** with:

`"This event is already added!"`

close()

This **function** should **clear** the events **array** and return a simple message:

`"{name} hall is closed."`

toString()

This **function** should **return:**

`"{name} hall - { capacity }"`

If there are any **`events`** then add on a new line:

`"Events: {all events joined by ', '}"`

Class MovieTheater

Class **`MovieTheater`** inherits class **`Hall`**.

constructor(capacity, name, screenSize)

Should have these **4** properties:

- `capacity – number;`
- `name – string;`
- `events - array of strings;`
- `screenSize - string,`

close()

This **function** should inherit the **`close()`** method of class **`Hall`** and extend the **returned** message adding this at the end on the same line:

`"All screenings are over."`

toString()

This **function** should extend the **`toString()`** method of class **`Hall`** returning the message with next string on a new line**:**

`"{name} is a movie theater with {screenSize} screensize and {capacity} seats capacity."`

Note: For more information see the examples below!

Class ConcertHall

Class **`ConcertHall`** inherits class **`Hall`**.

constructor(capacity, name)

Should have these **3** properties:

- `capacity – number;`
- `name – string;`
- `events - array of strings;`

hallEvents(title, performers)

This **function** should receive single **`title`** like **string** and **`performers`** – **array** of strings. This method inherit the **`hallEvents()`** method of class **`Hall`** like adding title to the **`events`** array, but also creates new property **`performers`** and **returns** a message:

`"Event is added."`

If event is already added to the events array **throw an error** with:

`"This event is already added!"`

close()

This **function** should inherit the **`close()`** method of class **`Hall`** and extend the **returned** message adding this at the end on the same line:

`"All performances are over."`

toString()

This **function** should extend the **`toString()`** method of class **`Hall.`** If there are any events into the events array, add to the returned message on new line**:**

`"Performers: { all performers joined by ', '}."`

Note: __For more information see the examples below!__

Examples

Sample code usage
```
let classes = solveClasses();
let hall = new classes.Hall(20, 'Main');
console.log(hall.hallEvent('Breakfast Ideas'));
console.log(hall.hallEvent('Annual Charity Ball'));
console.log(hall.toString());
console.log(hall.close());
---------------------------------------------------------------------------------
let movieHall = new classes.MovieTheater(10, 'Europe', '10m');
console.log(movieHall.hallEvent('Top Gun: Maverick'));
console.log(movieHall.toString());
---------------------------------------------------------------------------------
let concert = new classes.ConcertHall(5000, 'Diamond');
console.log(concert.hallEvent('The Chromatica Ball', ['LADY GAGA']));
console.log(concert.toString());
console.log(concert.close());
console.log(concert.toString());
```
Corresponding output
```
Event is added.

Event is added.

Main hall - 20

Events: Breakfast Ideas, Annual Charity Ball

Main hall is closed.

---------------------------------------------------------------------------------
Event is added.

Europe hall - 10

Events: Top Gun: Maverick

Europe is a movie theater with 10m screensize and 10 seats capacity.

---------------------------------------------------------------------------------
Event is added.

Diamond hall - 5000
```

```
Events: The Chromatica Ball
Performers: LADY GAGA.
Diamond hall is closed.All performances are over.
Diamond hall - 5000
```

Problem 3. Movie

```
class Movie {
    // TODO: implement this class...
}
```

Your Task

Write a **`class Movie`**, which implements the following functionality:

Functionality

constructor (movieName, ticketPrice)

Receives **2** parameters at initialization of the class - **`movieName`** and **`ticketPrice`**.

Should have these **3** properties:

- **`movieName`** – property of type **string**;
- **`ticketPrice`** – property of type **number,** may come as a **string**;
- **`screenings`**– initially an empty **array**;

Hint: Here you can add more properties to help you finish the task.

`newScreening(date, hall, description)`

The **`date, hall`** and **`description`** are of type **string**.

- Check if there is already entered screening with the same date and hall and **throw an `Error`**:

  `"Sorry, {hall} hall is not available on {date}"`

- Otherwise this function should **add** the screening to the screenings array and **return:**

  `"New screening of {movieName} is added."`

`endScreening(date, hall, soldTickets)`

- Check if the given **`screening array`** has a screening with the given date and hall, if **NOT throw an `Error`**:

  `"Sorry, there is no such screening for {movieName} movie."`

- Otherwise calculate the current screening **profit** of sold **tickets**, add the profit to the total profit for the movie, also add the **sold tickets** count to the total sold movie tickets, **delete** the screening from the screenings array and **return**:

  `"{movieName} movie screening on {date} in {hall} hall has ended. Screening profit: {currentProfit}"`

toString ()

In the end is the **toString()** method where we **return** the full information of the movie.

- At the first line:

  `"{movieName} full information:"`

- On the **second two** lines comes the collected **profit,** fixed to 0 and count of all **sold tickets:**

  `"Total profit: {profit}$`

  `Sold Tickets: {soldTickets}"`

- If there are screenings into the **screening array**, add on new line:
  `"Remaining film screenings:"`
  Sort screenings **alphabetically** by hall name and add a message for each of them on new line:

  `"{hall} - {date} - {desrtiption}"`

- If there are no screenings into screenings array add this line to the returned message:
  `"No more screenings!"`

Examples

Sample code usage
```let m = new Movie('Wonder Woman 1984', '10.00');```

```
let m = new Movie('Wonder Woman 1984', '10.00');
console.log(m.newScreening('October 2, 2020', 'IMAX 3D', `3D`));
console.log(m.newScreening('October 3, 2020', 'Main', `regular`));
console.log(m.newScreening('October 4, 2020', 'IMAX 3D', `3D`));
console.log(m.endScreening('October 2, 2020', 'IMAX 3D', 150));
console.log(m.endScreening('October 3, 2020', 'Main', 78));
console.log(m.toString());

m.newScreening('October 4, 2020', '235', `regular`);
m.newScreening('October 5, 2020', 'Main', `regular`);
m.newScreening('October 3, 2020', '235', `regular`);
m.newScreening('October 4, 2020', 'Main', `regular`);
console.log(m.toString());
```

Corresponding output

```
New screening of Wonder Woman 1984 is added.
New screening of Wonder Woman 1984 is added.
New screening of Wonder Woman 1984 is added.
Wonder Woman 1984 movie screening on October 2, 2020 in IMAX 3D hall has ended. Screening profit: 1500
Wonder Woman 1984 movie screening on October 3, 2020 in Main hall has ended. Screening profit: 780
Wonder Woman 1984 full information:
Total profit: 2280$
Sold Tickets: 228
Remaining film screenings:
IMAX 3D - October 4, 2020 - 3D
```

```
Wonder Woman 1984 full information:
Total profit: 2280$
Sold Tickets: 228
Remaining film screenings:
235 - October 4, 2020 - regular
235 - October 3, 2020 - regular
IMAX 3D - October 4, 2020 - 3D
Main - October 5, 2020 - regular
Main - October 4, 2020 - regular
```

GOOD LUCK! 😊