DOUGLAS COLLEGE

COMMERCE AND BUSINESS ADMINISTRATION

CSIS 3380: Full Stack Development with JavaScript

Assignment 1

(5 % towards your Final Grade)

Note: The assignment is to be completed individually. Any form of cheating or sharing of work may have serious consequences.

Instructions: You will build a Movie Data Processor App using advanced JavaScript features and functional programming principles. Use ES6 modules, arrow functions, destructuring, spread/rest operators, and functional array methods. All operations should not mutate the original dataset unless specified.

Setup:

a. Download the assignment folder **movie-processor** and rename it as **movie-processor_yourname**.
b. Inside it, locate **movies.js** (dataset) and **index.html**.
c. Create:
   - index.js → main entry point for coding JavaScript features
   - utils.js → helper functions
d. Link the index.js to index.html
e. Module setup:
   - In index.js, import the movie dataset from movies.js.
   - In utils.js, export all helper functions for reuse in index.js.

Step 1: Basic Data Handling

a. Import **movies** dataset in index.js. Use const wherever possible and use let only for reassignment.

b. Write an arrow function in index.js called **printMovieStats** for
   - Printing the number of movies and the dataset span (from earliest to latest year). Use a template literal to print the results. (**Sample output in the console**: The dataset contains 10 movies spanning from 1972 to 2019.)
   - Call this function immediately after importing the dataset.

Step 2: Advanced JavaScript Features

a. Object Destructuring:
   - **In utils.js,** create an arrow function named **findFirstHighRatedMovie** which finds the first movie with a rating higher than 8.8. Use object destructuring to extract title, year, and rating. (**Sample output in the console**: The Dark Knight (2008) has an IMDb rating of 9.0.).
     - Note: If there are multiple movies above 8.8, destructure only the first one.
   - Call findFirstHighRatedMovie(movies) in **index.js**.

b. Array Destructuring

- **In utils.js,** create an arrow function named **sortMoviesByRating** which sorts movies in descending order by rating. Use **array destructuring** to extract:
    - topMovie1 → highest rated
    - topMovie2 → second highest rated
    - otherTopMovies → the rest of the movies.
- Display all three variables in the console.
- Call sortMoviesByRating(movies) in index.js.

c. Function Argument Destructuring

- In **utils.js**, create a function **displayMovieSummary** that takes a movie object. Use parameter destructuring to extract title, year, and genre from the passed-in movie object. Print the summary (**Sample output in the console**: Inception (2010) – Genre: Sci-Fi)
- **In index.js**, call displayMovieSummary(movie) for each movie using **forEach**.

d. **Object literal enhancement / restructuring**:

- In utils.js, create a function **createMovieObject** that accepts title, year, genre, rating, duration. The function should return a new movie object using object literal enhancement.
- **Next, in index.js, c**all createMovieObject to create a new movie object. Check if a movie with the same title already exists in the movies array. If it doesn't exist, create a new array **newMovies** with the new movie added. Otherwise, keep the array unchanged.

e. Spread operator:

i. Combine Arrays:

- **In index.js**, assume you have an array called **newReleases** containing 2–3 new movie objects (same format as the dataset).
- Create a new array allMovies by merging movies (or newMovies if you created it earlier) with newReleases using the spread operator.

ii. Clone the Array:

- **In index.js**, clone **allMovies** to **allMoviesClone**.
- Add a new movie to the clone.
- Print both arrays to verify that the original **allMovies** is unchanged.

iii. Split Top and Remaining Movies with Spread

- In index.js, create an arrow function named **getTopAndRemainingMovies**. Inside the function, sort the allMovies array by rating in descending order (highest rated first).
- Use array destructuring with the spread operator to split the sorted array into two groups:
    - topMovies → the first 3 movies (highest rated)
    - remainingMovies → the rest of the movies
- Print both arrays to the console in a clear format.

iv. Collect Function Arguments Using Rest Parameters:

- **In utils.js**, write a function called **logSelectedMovies** that uses the rest parameter (...movieTitles) to accept multiple titles.
- Print the titles. (**Sample output in the console**: You selected: Inception, Interstellar, Parasite)
- In index.js, call this function with the **titles** of topMovies.

Step 3: Functional Programming

Implement a set of **query functions** using only functional array methods (map, filter, reduce, find, sort, etc.). These functions must **not** use loops (for, while), only functional methods.

a. **In utils.js**, write the following arrow functions:

- **addWatchedFlag()**: returns a **new array** of movies where each movie object has an additional key: watched: false. Use map to create the new array. Do not mutate the original objects — instead, return new objects with the spread operator.
- **getMoviesAfter(year): r**eturns all movies released **after the given year**. Use filter.
- **getAverageRating(): r**eturns the **average rating** of all movies. Use reduce to compute the total and divide by the number of movies.
- **findMovie(title):** returns the **first movie** that matches the title. Use find.
- **areAllAbove(rating)**: accepts a rating threshold. Returns true if **all movies** have a rating above that threshold, otherwise false. Use every.
- **getMoviesByGenre(genre): a**ccepts a genre (e.g., "Sci-Fi"). Returns all movies of that genre. Use filter.
- **getLongestMovie()**: returns the movie with the **maximum duration**. Use reduce to compare movies.

b. **In index.js, i**mport all query functions from utils.js. Call each function with sample inputs and print results clearly using template literals. Examples:

- console.log(`Movies with watched flag: ${addWatchedFlag()}`);
- console.log(`Movies after 2010: ${getMoviesAfter(2010)}`);
- console.log(`Average Rating: ${getAverageRating().toFixed(2)}`);
- console.log(`Find 'Inception': ${findMovie("Inception")}`);
- console.log(`Are all movies rated above 7.0? ${areAllAbove(7.0)}`);
- console.log(`Sci-Fi Movies: ${getMoviesByGenre("Sci-Fi")}`);
- console.log(`Longest Movie: ${getLongestMovie()}`);

Submission:

Compress/zip your Assignment 1 folder "**movie-processor_yourname**" and upload it to Blackboard. NO LATE SUBMISSION is allowed.