Functions

Impure Functions:

1. **Initialize Movie Ratings** Create a function `initializeRatings` that initializes an empty array to store movie ratings. Each rating will eventually be an array with two elements: a string representing the movie title and a number representing the rating. This function should set a global array variable `movieRatings` to an empty array.

2. **Add a Movie Rating** Write a function `addRating` that takes two parameters: `movieTitle` (a string) and `rating` (a number). This function should add a new sub-array to the `movieRatings` array with the movie title and rating.

3. **Find a Movie Rating** Create a function `findRating` that takes one parameter, `movieTitle` , and logs the rating of the movie. If the movie is not found, it should log `Movie was not found` .

4. **Update a Movie Rating** Write a function `updateRating` that takes two parameters: `movieTitle and newRating` . It should update the rating of the specified movie. If the movie does not exist in the list, it should print a message indicating so.

5. **Remove a Movie Rating** Develop a function `removeRating` that takes one parameter, `movieTitle` , and removes the movie from the `movieRatings` array. If the movie is not found, it should print a message indicating so.

6. **List All Movies and Ratings** Create a function `listMovies` that prints all movies and their ratings in the format "Movie Title - Rating".

7. **Find Highest Rated Movie** Create a function `highestRatedMovie` that finds and prints the title of the movie with the highest rating. If there are no movies, it should print a message indicating so.

Pure Functions:

1. **Average Rating:**

- Write a pure function `calculateAverageRating` that takes an array of numbers representing movie ratings (1-5) and returns the average rating.
- **Input**: `[3, 4, 5, 3, 5]`
- **Expected Output**: `4`

2. **Filter Movies by Rating:**

- Create a pure function `filterByRating` that takes an array of movie ratings and a minimum rating value. It should return a new array containing only the ratings that are equal to or higher than the minimum rating.
- **Input**: `([3, 4, 5, 2, 1, 5], 4)`
- **Expected Output**: `[4, 5, 5]`

3. **Highest Rated Movie:**

- Write a pure function `findHighestRating` that takes an array of movie ratings and returns the highest rating in the array.
- **Input**: `[3, 4, 5, 3, 5]`
- **Expected Output**: `5`

4. **Ratings Above Threshold:**

- Craft a pure function `ratingsAboveThreshold` that takes an array of ratings and a threshold rating. It should return a new array with ratings that are strictly greater than the threshold.
- **Input**: `([3, 4, 5, 2, 1, 5], 3)`
- **Expected Output**: `[4, 5, 5]`

5. **Count Movies in Rating Range:**

- Design a pure function `countInRatingRange` that takes an array of ratings and two numbers representing the lower and upper bounds of a rating range (inclusive). The function returns the count of movies within that rating range.
- **Input**: `([3, 4, 5, 2, 1, 5], 4, 5)`
- **Expected Output**: `4`

6. **Filter Unique Ratings:**

- Develop a pure function `uniqueRatings` that takes an array of movie ratings and returns a new array of ratings without duplicates, preserving their original order in the input array.
- **Input**: `[5, 3, 4, 3, 5, 4, 5]`
- **Expected Output**: `[5, 3, 4]`

7. **Merge Ratings:**

- Write a pure function `mergeRatings` that combines two arrays of movie ratings into one array, removing any duplicates, and returns the new array. Ensure it does not modify the input array.
- **Input**: `([5, 3, 4], [2, 3, 5])`
- **Expected Output**: `[5, 3, 4, 2]`

8. **Ratings Differential:**

- Create a pure function `ratingsDifferential` that takes two arrays of movie ratings and calculates the difference between the average ratings of the two arrays. You can use the function you created in exercise 1 in this section to calculate the ratings of each array.
- **Input**: `([4, 4, 4, 4, 5], [3, 3, 3, 3])`
- **Expected Output**: `1.25` (The average difference between the two sets of ratings)

9. **Filter Ratings by Multiple Criteria:**

- Write a pure function `filterRatingsByCriteria` that takes an array of ratings and multiple criteria (e.g., greater than a value, less than a value), and returns a new array of ratings that meet all criteria.
- **Input**: `([1, 2, 3, 4, 5], greaterThan=2, lessThan=5)`
- **Expected Output**: `[3, 4]`

Bonus Exercises

1. **Sort Movie Ratings:**

- Develop a pure function `sortRatings` that takes an array of movie ratings and returns a new array with the ratings sorted from lowest to highest. Implement the sorting algorithm using loops and without the `sort` method. Ensure it does not modify the input array.

- **Input**: `[5, 3, 4, 2, 1]`
- **Expected Output**: `[1, 2, 3, 4, 5]`

2. **Average Rating of Top N Movies:**

- Develop a pure function `averageOfTopN` that takes an array of movie ratings and an integer `N`. It should return the average rating of the top `N` rated movies. You can use the function you created in exercise 1 in this section in order to sort the arrays.
- **Input**: `([3, 1, 5, 4, 2], 3)`
- **Expected Output**: `4` (The top 3 ratings are 5, 4, and 3)

3. **Create Rating Frequency Map:**

- Write a pure function `ratingFrequency` that takes an array of movie ratings and returns an array where each element is a sub-array with two elements: the rating and the number of times it occurs in the input array.
- **Input**: `[3, 4, 3, 5, 4, 5, 5]`
- **Expected Output**: `[[3, 2], [4, 2], [5, 3]]`

4. **Normalize Ratings:**

- Create a pure function `normalizeRatings` that takes an array of ratings (1-5) and scales them to a new range (e.g., 0-10), returning a new array of normalized ratings.
- **Input**: `([1, 2, 3, 4, 5], 0, 10)`
- **Expected Output**: `[0, 2.5, 5, 7.5, 10]`