ES6 Features - Spread, Destructure, Rest

1. Extract Hero Stats

Given an object representing a hero, use object destructuring to extract and log the hero's name, level, and class.

```
const hero = {
  name: "Aragorn",
  level: 20,
  class: "Ranger",
  race: "Human"
};
```

2. Swap Top Two Heroes

Given an array of hero names, use destructuring to swap the positions of the first two heroes in the array.

```
let topHeroes = ["Gandalf", "Legolas", "Frodo",
"Samwise"];
```

3. Update Hero Levels

Given an object where keys are hero names and values are levels, write a function that takes this object and an updates object. The function should apply the updates and return a new object using the spread operator.

```
const heroLevels = {
  "Aragorn": 20,
  "Legolas": 19
};

const updates = {
  "Aragorn": 21,
  "Legolas": 20
};
```

4. Combine Hero Quests

You have two arrays of quests collected from two different adventurers. Combine these arrays into one, removing duplicates, using the spread operator.

```
const quests1 = ["Find the ring", "Save the king",
"Protect the realm"];
const quests2 = ["Destroy the ring", "Find the lost
city", "Save the king"];
```

5. Extracting Values from Nested Hero Objects

Given a nested object containing hero information including a nested object of equipment, use object destructuring to extract the names of the first two pieces of equipment only.

```
const heroDetails = {
  name: "Legolas",
  equipment: {
    first: "Bow of the Galadhrim",
    second: "Elven dagger",
    third: "Cloak of Lothlórien"
  }
};
```

6. Merge Hero Objects

You have two objects representing details about the same hero: one containing the base info and the other containing additional stats. Merge these objects using the spread operator.

```
const baseInfo = {
  name: "Gandalf",
  class: "Wizard"
};

const additionalStats = {
  level: 25,
  power: 100
};
```

7. Expand Hero Skills

Given an object representing a hero's details, including a skills array, write a function that adds new skills to the hero's skills list. The function should be capable of taking any number of new skills and adding them to the existing list without creating duplicates. Use the spread operator to handle the skills array appropriately.

```
const heroDetails = {
  name: "Legolas",
  level: 20,
  skills: ["Archery", "Sneaking", "Hand-to-hand combat"]
};
```

8. Flatten Quest Log

Imagine you have a quest log where each entry is an array representing quests that a hero plans to complete per month. Write a function that flattens this quest log into a single array of quests, removing any duplicates. Use the spread operator to concatenate arrays and ensure the quest log is a flat array of unique quests.

```
const questLog = [
  ["Defeat the dragon", "Rescue the princess"],
  ["Rescue the princess", "Explore the ancient ruins"],
  ["Explore the ancient ruins", "Discover the hidden
treasure"]
];
```

9. Favorite Quest Extractor

Given an array of adventurer objects where each adventurer has a name and a list of favorite quests, write a function that extracts the first favorite quest for each adventurer. Use the `map` method and destructuring to achieve this.

```
const adventurers = [
  { name: "Bilbo", favorites: ["Find the ring", "Join the
dwarves"] },
  { name: "Frodo", favorites: ["Destroy the ring", "Visit
Rivendell"] },
  { name: "Aragorn", favorites: ["Protect the hobbits",
"Serve the king"] }
];
```

10. Quest Difficulty Updater

Given an array of quests where each quest is an object containing the quest's name and difficulty level, write a function that increases the difficulty of a specific quest by a given amount. Use the `map` method and object destructuring in the function to simplify your code.

```
const quests = [
  { name: "Defeat the dragon", difficulty: 8 },
  { name: "Rescue the princess", difficulty: 5 }
];
```