**CSCI 344 Notes**

**Higher-Order Functions & Abstraction**

1. **Abstraction in Programming**:
   * Programs can be simplified by abstracting repetitive or complex tasks into functions. This reduces the likelihood of bugs because it allows developers to focus on higher-level logic without getting bogged down by details.
   * Functions like sum and range abstract complex operations like summing a range of numbers, which makes code simpler and more reliable.
2. **Higher-Order Functions**:
   * These are functions that operate on other functions, either by taking them as arguments or returning them. Examples include JavaScript's built-in map, filter, and reduce methods.
   * **Example**: repeat(n, action) abstracts a loop, where action is a function passed as an argument to be executed n times.
3. **Abstracting Repetition**:
   * Functions like repeatLog(n) or repeat(n, action) showcase how actions can be abstracted into higher-order functions, allowing more flexibility in what actions are executed during each iteration.
4. **Array Methods**:
   * JavaScript arrays have many higher-order methods like:
     + **forEach**: Executes a provided function once for each array element.
     + **filter**: Creates a new array with elements that pass a test.
     + **map**: Transforms an array by applying a function to each element.
     + **reduce**: Combines elements of an array into a single value.
   * **Example**: Using filter to find living scripts from a dataset or using reduce to sum an array.
5. **Functions That Modify Other Functions**:
   * Functions like noisy(f) modify the behavior of another function by adding extra functionality (e.g., logging inputs and outputs).
6. **Composability**:
   * Higher-order functions make code composable, allowing developers to chain operations together for powerful data transformations.
   * Example: Filtering scripts, mapping them to names, and reducing their character counts can all be done in a chain of operations.
7. **Data Processing with Higher-Order Functions**:
   * By leveraging higher-order functions, complex data processing tasks (like finding the average year of origin for scripts) can be simplified and expressed in a clear, concise manner.
8. **Recognizing Text Using Character Codes**:
   * JavaScript provides methods like codePointAt to correctly identify Unicode characters, making it possible to write functions like characterScript to detect which script a given character belongs to.

**Exercises:**

1. **Flattening Arrays**:
   * Using reduce and concat to combine nested arrays into a single array.
2. **Custom Loop Function**:
   * Write a loop function that behaves like a for loop but accepts a test function, update function, and body function as arguments.
3. **Every Method**:
   * Implement every using both a loop and the some method. every returns true if all elements pass a test.
4. **Dominant Writing Direction**:
   * Write a function to compute the dominant writing direction in a text based on the scripts used.

**Summary:**

* **Higher-order functions** allow programmers to abstract common patterns, making code more readable and easier to manage.
* Functions like filter, map, and reduce are examples of these abstractions and are powerful tools for data manipulation.
* By abstracting complexity, you create reusable code that is easier to debug, understand, and modify.

Top of Form

Bottom of Form