Basic Data Structures Activities Part 2

These activities will continue your practice with some of the tools we discussed in the lecture, as well as give you some practice with open-ended problems involving arrays and objects.

1. From Pairs

Write a function called `fromPairs` that takes an array of key-value pairs and turns it into an object.

```
console.log(
  fromPairs([
    ["a", 1],
    ["b", 2],
  ])
); // { a: 1, b: 2 }
console.log(
  fromPairs([
    [7, "apples"],
    [2, "socks"],
  ])
); // { 7: "apples", 2: "socks" }
```

2. Replace Item

Write a function called `replaceItem` that removes 1 item from a given array at a given index and inserts a given new item.

```
console.log(replaceItem(["a", "b", "c"], 1, "z")); // ["a", "z", "c"]
console.log(replaceItem(["Cowboys", "Eagles", "Raiders"], 2, "Broncos")); //
["Cowboys", "Eagles", "Broncos"]
```

3. A Piece of the Array

Write a function called `partial` that returns a portion of a given array from a given starting index to a given stopping index.

```
console.log(partial([10, 11, 12, 13, 14, 15], 1, 3)); // [11, 12]
console.log(partial(["vanilla", "chocolate", "strawberry", "peach"], 0, 2)); //
["vanilla", "chocolate"]
```

NOTE: The next two problems are more open-ended than what you've worked on so far in the basic data structures problem sets. You may need to combine concepts from the current lecture (arrays and objects) with what you've learned in earlier units or draw from your resources (like loops and

conditionals). You may not be able to solve these using just one built-in method. Be sure to think through the logic!

4. Group By Category

Write a function called `groupByCategory` that takes an array of objects, each representing an item with a name and category, and returns an object that groups all item names under their respective categories.

```
console.log(
  groupByCategory([
    { name: "Apple", category: "Fruit" },
    { name: "Carrot", category: "Vegetable" },
    { name: "Banana", category: "Fruit" },
    { name: "Broccoli", category: "Vegetable" },
  ])
); // { Fruit: ["Apple", "Banana"], Vegetable: ["Carrot", "Broccoli"] }

console.log(
  groupByCategory([
    { name: "Wrench", category: "Tools" },
    { name: "Hammer", category: "Tools" },
    { name: "Notebook", category: "Stationery" },
    { name: "Pen", category: "Stationery" },
    { name: "Drill", category: "Tools" },
  ])
); // { Tools: ["Wrench", "Hammer", "Drill"], Stationery: ["Notebook", "Pen"] }

console.log(
  groupByCategory([
    { name: "Shirt", category: "Clothing" },
    { name: "Apple", category: "Food" },
    { name: "Laptop", category: "Electronics" },
  ])
); // { Clothing: ["Shirt"], Food: ["Apple"], Electronics: ["Laptop"] }
```

5. Highest Scoring Student

Write a function called highestScoringStudent that takes an array of student objects and returns the name of the student with the highest score. If two or more students share the highest score, return an array containing the students' names.

Return values

- The name of the highest-scoring student as a string, if only one.
- An array of names (strings) if multiple students share the highest score.

```
console.log(
  highestScoringStudent([
    { name: "Alex", score: 82 },
```

```
    { name: "Bob", score: 91 },
    { name: "Charlie", score: 88 },
  ])
); // Bob

console.log(
  highestScoringStudent([
    { name: "Karl", score: 72 },
    { name: "Lena", score: 89 },
    { name: "Mark", score: 67 },
    { name: "Nina", score: 89 },
  ])
); // ["Lena", "Nina"]

console.log(
  highestScoringStudent([
    { name: "George", score: 70 },
    { name: "Helen", score: 70 },
    { name: "Ivan", score: 70 },
  ])
); // ["George", "Helen", "Ivan"]
```