Travel Planner

Loops - Extra

1. Calculate Total Travel Time

- **Task:** Write a function that calculates the total time it will take to visit all the destinations in the given array. Each destination has a `distance` property (in miles) and a `speed` property (in miles per hour). Assume that there's no time spent at the destinations.
- **Inputs:** An array of destination objects.
- **Example:**
  - Input:
    ```
    [{ name: "Paris", distance: 500, speed: 50 }, { name: "London", distance: 200, speed: 70 }, { name: "New York", distance: 3000, speed: 500 }]
    ```
  - Output: `27.14` hours
- **Tip:** Use a loop to iterate over the destinations, calculate the time it takes to get to each destination (`distance/speed`), and sum these times up.

2. Find Longest Travel Route

- **Task:** Write a function that finds the longest travel route in the given array. Each destination object includes an array of `routes` objects, each with a `distance` property. Return the route object.
- **Inputs:** An array of destination objects.
- **Example:**
  - Input:
    ```
    [{ name: "Paris", routes: [{ name: "Route 1", distance: 300 }, { name: "Route 2", distance: 250 }] }, { name: "London", routes: [{ name: "Route 1", distance: 200 }, { name: "Route 2", distance: 280 }] }]
    ```
  - Output: `{ name: "Route 1", distance: 300 }`
- **Tip:** Use nested loops to iterate over the destinations and their routes, keeping track of the route with the maximum distance.

3. Sort Destinations by Distance

- **Task:** Write a function that sorts an array of destinations in descending order by the total route distance. Each destination has an array of `routes`, each with a `distance` property. Return the sorted array.
- **Inputs:** An array of destination objects.
- **Example:**
  - Input:
    ```
    [{ name: "Paris", routes: [{ distance: 300 }, { distance: 250 }] },
    { name: "London", routes: [{ distance: 200 }, { distance: 280 }] }]
    ```
  - Output:
    ```
    [{ name: "Paris", routes: [{ distance: 300 }, { distance: 250 }] },
    { name: "London", routes: [{ distance: 200 }, { distance: 280 }] }]
    ```
- **Tip:** Use a loop to calculate the total distance for each destination. Use the Array's `sort` method to sort the destinations.

4. Travel Itinerary

- **Task:** Write a function that builds a travel itinerary from a starting destination. The function should return a path as an array of destinations. Each destination has a `name` and `connections` array containing names of destinations it connects to.
- **Inputs:** An array of destination objects, starting destination name.
- **Example:**
  - Input:
    ```
    [{ name: "Paris", connections: ["London"] }, { name: "London",
    connections: ["New York"] }, { name: "New York", connections: [] }]
    ```
    , start: "Paris"
  - Output: `["Paris", "London", "New York"]`
- **Tip:** This problem is more complex and may require a depth-first search or breadth-first search algorithm to solve.

5. Least Number of Connections

- **Task:** Write a function that determines the destination with the least number of connections and returns the destination's name. Each destination object has a `connections` array containing names of destinations it connects to.
- **Inputs:** An array of destination objects.
- **Example:**

  - Input:
    ```
    [{ name: "Paris", connections: ["London", "New York"] }, { name: "London", connections: ["New York"] }, { name: "New York", connections: [] }]
    ```
  - Output: `"New York"`
- **Tip:** Use a loop to iterate over the destinations, and keep track of the destination with the smallest number of connections.