1. User and Posts Fetch with Promise Chaining

Write a function that:

- Fetches user data from `https://dummyjson.com/users/1`.
- Uses the fetched user's `id` to fetch their posts from `https://dummyjson.com/users/1/posts`.
- Logs the user's `firstName` and the titles of their posts.
- Use **Promise chaining** to implement this.

---

2. Error Handling with Retry Logic (Promises Only)

Write a function that:

- Fetches a post from `https://dummyjson.com/posts/1`.
- Simulate a failure by using an incorrect URL (e.g., `https://dummyjson.com/post/1`).
- Logs an error message if the fetch fails.
- Retries the fetch up to 3 times.
- Use Promises (not `async/await`) to handle retries and error handling.

---

3. Sequential vs Parallel Fetches

Write two functions to fetch data from the following APIs:

1. `https://dummyjson.com/users`
2. `https://dummyjson.com/posts`
3. Fetch the data **sequentially**, ensuring the second fetch starts only after the first fetch is completed.
4. Fetch the data **in parallel** using `Promise.all` to execute both fetches at the same time.
- Compare and log the time taken for both approaches.

---

4. Retry Logic with Exponential Backoff

Write a function that:

- Fetches a post from `https://dummyjson.com/posts/1`.
- Implements retry logic:
    - Retry the fetch operation up to 3 times if the request fails.
    - Double the delay between retries (e.g., 1 second, 2 seconds, 4 seconds).
- Logs whether the data fetch succeeded or failed after all retries.
