(Notes) Promises

Question 1: What is Output?
Explanation:
It will output `"start"`, then `"1"`, then `"end"`, and finally `"2"`. This is because the Promise is resolved asynchronously, so the `then` callback will be executed after the synchronous code finishes executing.

Question 2: What is Output?
Explanation:
It will output `"start"`, then `"1"`, then `"3"`, then `"end"`, and finally `"2"`. The Promise is still resolved asynchronously, so the `then` callback will be executed after the synchronous code finishes executing.

Question 3: What is Output?
Explanation:
It will output `"start"`, then `"middle"`, then `"1"`, then `"end"`, and finally `"success"`. The Promise inside the function `fn` is resolved asynchronously, so the `then` callback will be executed after the synchronous code finishes executing.

Question 4: What is Output?
Explanation:
It will output `"Error 1"` and `"Success 4"`. Since the Promise is rejected in the `job` function, it will skip the `then` callbacks and go directly to the first `catch` callback.

Question 5: What is Output?
Explanation:
It will output `"success"`, `"error"`, `"Error caught"`, and finally `"success"`. The first Promise resolves successfully, then the second Promise is rejected, triggering the `catch` callback. After that, it continues with the `then` and `catch` callbacks accordingly.

Question 6: What is the Output?
Explanation:
It will output `"success"`, `"error"`, `"Defeat"`, `"error"`, and finally `"Error caught"`. The

first Promise resolves successfully, then the second Promise resolves with a value that doesn't match `'victory'`, triggering the `catch` callback and throwing an error.

Question 7: Promises Chaining

Explanation:

It will output `"First!"`. The second Promise resolves to the first Promise itself, so it simply returns the resolved value of the first Promise.

Question 8: Rewrite this example code using async/await

Explanation:

The code is already rewritten using `async/await`. It uses `async` function to define an asynchronous function and `await` to wait for the Promise to resolve.

Question 9: Solve Promise Recursively

Explanation:

It will output `"Subscribe to Roadside Coder"`, `"Like the Javascript Interview Questions video"`, and finally `"Share the Javascript Interview Questions video"`. The `promRecurse` function recursively processes each Promise in the array and logs the resolved value.

Question 10: Promise Polyfill

Explanation:

The Promise Polyfill is a custom implementation of the Promise class. It provides basic functionalities like `then`, `catch`, `resolve`, and `reject`. The example demonstrates how to create and use a Promise using the custom Polyfill.
