Here are detailed answers to your technical interview questions:

\*\*1) Concurrency vs. Parallelism:\*\*

- \*\*Concurrency\*\* is the ability of a system to handle multiple tasks simultaneously by interleaving their execution. It doesn't necessarily mean that tasks run in parallel but rather share resources efficiently.

- \*\*Parallelism\*\*, on the other hand, is the execution of multiple tasks in parallel, typically on multiple CPU cores, to achieve faster results. It implies true simultaneous execution.

\*\*2) Real-life example of Parallelism:\*\*

- Rendering a movie: When rendering a movie, different frames can be processed in parallel by multiple CPU cores or computers, speeding up the rendering process.

\*\*3) `git fetch` vs. `git pull`:\*\*

- `git fetch` retrieves changes from a remote repository and updates your local repository's remote-tracking branches but doesn't merge them into your current branch. It's useful to see changes before deciding to merge.

- `git pull` is a combination of `git fetch` followed by `git merge`. It fetches changes from the remote repository and automatically merges them into your current branch.

\*\*4) Handling a Git Merge Request (MR):\*\*

- Initialize a Git repository: `git init`

- Add files: `git add <file>`

- Commit changes: `git commit -m "Commit message"`

- Create a new branch: `git checkout -b feature-branch`

- Make changes, commit them, and push to your remote repository.

- Create the MR on your Git hosting platform.

- Review the changes, resolve conflicts if any.

- Approve and merge the MR.

\*\*5) Subversion vs. Git:\*\*

- Subversion (SVN) is a centralized version control system, while Git is distributed.

- SVN tracks file changes, while Git tracks changes at the commit level.

- Git provides better branching and merging capabilities.

- Git allows offline work, whereas SVN requires a network connection for most operations.

\*\*6) Method Overriding vs. Overloading:\*\*

- \*\*Method Overriding\*\*: Involves creating a new implementation of a method in a subclass with the same name, return type, and parameters as a method in the superclass. It is used to provide a specific implementation in the subclass.

- \*\*Method Overloading\*\*: Involves creating multiple methods in the same class with the same name but different parameter lists (number, type, or order). It is used to define multiple behaviors for methods with the same name.

\*\*7) Polymorphism:\*\*

- Polymorphism is a fundamental concept in OOP that allows objects of different classes to be treated as objects of a common superclass. It enables flexibility and extensibility in code.

\*\*8) Sync vs. Async:\*\*

- \*\*Synchronous (Sync)\*\*: Execution happens sequentially, and each operation blocks until it completes. It's simpler but can lead to performance bottlenecks.

- \*\*Asynchronous (Async)\*\*: Operations don't block; they run concurrently, and results are handled when available. It's used to improve system responsiveness and scalability.

\*\*9) Node.js vs. Spring Boot:\*\*

- \*\*Node.js\*\*: Known for its non-blocking, event-driven architecture. Suitable for building real-time applications and microservices. Ideal for I/O-intensive tasks.

- \*\*Spring Boot\*\*: A Java framework designed for building production-ready, enterprise-level applications. Offers a wide range of libraries and tools.

\*\*10) Quick Sort Code:\*\*

- Here's a basic implementation of the Quick Sort algorithm in Python:

```python

def quick\_sort(arr):

if len(arr) <= 1:

return arr

else:

pivot = arr[0]

less\_than\_pivot = [x for x in arr[1:] if x <= pivot]

greater\_than\_pivot = [x for x in arr[1:] if x > pivot]

return quick\_sort(less\_than\_pivot) + [pivot] + quick\_sort(greater\_than\_pivot)

```

\*\*11) Find Unique Pairs with Sum 10:\*\*

- Here's a Python code to find unique pairs of elements in the array `arr` whose sum equals 10:

```python

def find\_unique\_pairs(arr, target\_sum):

seen = set()

unique\_pairs = []

for num in arr:

complement = target\_sum - num

if complement in seen and (num, complement) not in unique\_pairs and (complement, num) not in unique\_pairs:

unique\_pairs.append((num, complement))

seen.add(num)

return unique\_pairs

arr = [8, 7, 2, 2, 2, 5, 3, 1]

target\_sum = 10

result = find\_unique\_pairs(arr, target\_sum)

print(result) # Output: [(8, 2), (7, 3)]

```

\*\*12) Linux Command to Find Files/Directories with "abc" in Their Name:\*\*

- To find files or directories containing "abc" in their name within the current directory and its subdirectories, you can use the `find` command:

```bash

find . -name "\*abc\*"

```

\*\*13) Linux Command to Find the Word "abc" in a File:\*\*

- To search for the word "abc" in a file (e.g., `file.txt`), you can use the `grep` command:

```bash

grep "abc" file.txt

```

\*\*Second Technical Round\*\*:

1. \*\*Project Discussion\*\*:

- Provide a detailed overview of your project, including its purpose, architecture, technologies used, and your specific role and contributions.

2. \*\*Deployment in AWS\*\*:

- Explain the steps involved in deploying your application on AWS, including setting up instances, configuring databases, and ensuring scalability and security.

3. \*\*Web Server vs. App Server\*\*:

- Differentiate between web servers (e.g., Apache, Nginx) and application servers (e.g., Tomcat, WildFly) and explain their roles in web application deployment.

4. \*\*Handling Payment Issues\*\*:

- Describe your approach for handling payment issues where a user's money is deducted but the payment is stuck. Discuss how you would handle it programmatically with third-party payment gateways, including error handling and retries.

5. \*\*Project Database Design\*\*:

- Explain the database design for your project, including data models, relationships, and any optimizations made for performance.

6. \*\*Normalization\*\*:

- Define normalization and its types (e.g., 1NF, 2NF, 3NF). Explain to what extent you applied normalization principles in your project's database design.

7. \*\*Doubly Linked List Example\*\*:

- Provide a real-life example of where a doubly linked list might be used, aside from image viewers and train scheduling systems. For example, it can be used in browser history.

8. \*\*Puzzle: Finding the Lightest Ball\*\*:

- This classic puzzle involves finding the lightest ball among eight identical balls, given a balance scale and a limited number of weighings. It typically requires a divide-and-conquer strategy.

Please use these explanations and code examples as references in your interviews, and feel free to ask for further clarification on any topic. Good luck with your interviews!