Grep Interview

Here are some **scenario-based** `grep` interview questions with solutions that dive deeper into practical use cases:

---

1. Scenario: Searching Logs for Errors

Question: You have a large log file, and you need to extract all lines that contain the word "error" (case insensitive). How would you do this with `grep`?

Solution:
Use the `-i` option to perform a case-insensitive search.

```
grep -i "error" logfile.log
```

---

2. Scenario: Finding a Pattern in Multiple Files

Question: You are given a directory with multiple `.txt` files. How do you find all occurrences of the word "failed" in any `.txt` file within this directory and its subdirectories?

Solution:
Use the `-r` option for recursive search and limit the search to `.txt` files.

```
grep -r "failed" *.txt
```

---

3. Scenario: Matching Whole Words

Question: In a file, you want to find only the occurrences of the word "cat" as a whole word (not part of another word like "catalog"). How would you achieve this?

Solution:
Use the `-w` option to match whole words.

```
grep -w "cat" filename.txt
```

---

4. Scenario: Displaying Lines Around a Match

Question: You need to search for the word "exception" in a log file, but you also want to include 2 lines before and 2 lines after each match for context. What command would you use?

Solution:
Use the `-C` option to specify the number of context lines.

```
grep -C 2 "exception" logfile.log
```

---

5. Scenario: Search for Multiple Patterns

Question: You are required to search a file for lines containing either "error" or "warning". How would you perform this search with `grep`?

Solution:
Use the `-E` option (extended regular expressions) to specify multiple patterns.

```
grep -E "error|warning" logfile.log
```

---

6. Scenario: Counting Occurrences of a Pattern

Question: How would you count the number of lines in a file that contain the word "success"?

Solution:
Use the `-c` option to count the matching lines.

```
grep -c "success" filename.txt
```

---

7. Scenario: Suppressing Output

Question: You want to check if the word "timeout" exists in a configuration file but don't need the actual matching lines—just a success/failure return code. How do you achieve this with `grep`?

Solution:
Use the `-q` option to suppress output and rely on the exit status.

```
grep -q "timeout" config.conf
```

- If the word is found, the exit code will be `0`.
- If not found, the exit code will be `1`.

---

8. Scenario: Search for Inverted Matches

Question: How would you find all lines in a file that do **not** contain the word "test"?

Solution:
Use the `-v` option to invert the match.

```
grep -v "test" filename.txt
```

---

9. Scenario: Display Line Numbers with Matches

Question: You need to locate the lines where the word "server" appears in a file, along with their line numbers. What command would you use?

Solution:
Use the `-n` option to display line numbers.

```
grep -n "server" filename.txt
```

---

10. Scenario: Matching Only Patterns at the Start of a Line

Question: You need to find lines in a file that start with the word "Status". What command would you use?

Solution:
Use the `^` anchor in your pattern to match the start of a line.

```
grep "^Status" filename.txt
```

---

11. Scenario: Extracting IP Addresses from a File

Question: You need to extract all valid IPv4 addresses from a file. How would you do this using `grep`?

Solution:
Use a regular expression for matching IPv4 patterns with the `-o` option to display only the matching part of the line.

```
grep -oE '([0-9]{1,3}\\.){3}[0-9]{1,3}' filename.txt
```

---

12. Scenario: Search for a Word in a Compressed File

Question: How would you search for the word "error" inside a compressed `.gz` log file without decompressing it manually?

Solution:
Use the `zgrep` command to search within compressed files.

```
zgrep "error" logfile.gz
```

---

13. Scenario: Searching with a Binary File

Question: You have a binary file, but you want to search for text in it. How would you instruct `grep` to treat the file as text?

Solution:
Use the `-a` option to force `grep` to treat binary files as text.

```
grep -a "pattern" binaryfile
```

---

14. Scenario: Searching for Empty Lines

Question: How do you find all empty lines in a file using `grep`?

Solution:
Use the following pattern to search for empty lines (lines with no characters).

```
grep "^$" filename.txt
```

---

15. Scenario: Filtering Output to Only Matching Strings

Question: You want to extract only the matching part of a line where the word "user" appears. How do you do this with `grep`?

Solution:
Use the `-o` option to print only the matched string.

```
grep -o "user" filename.txt
```

---

16. Scenario: Excluding Files from Recursive Search

Question: You are performing a recursive `grep` search but want to exclude all `.log` files from the search. How would you do this?

Solution:
Use the `--exclude` option to exclude files with a specific extension.

```
grep -r "pattern" /path/to/search --exclude="*.log"
```

---

17. Scenario: Matching a Pattern at the End of a Line

Question: How do you find lines in a file that end with the word "completed"?

Solution:
Use the `$` anchor to match the end of a line.

```
grep "completed$" filename.txt
```

---

18. Scenario: Search for a Pattern in Multiple Files and Show Filenames

Question: How do you search for the pattern "login" across multiple files but only show the filenames where the pattern occurs?

Solution:
Use the `-l` option to list only the filenames.

```
grep -l "login" *.txt
```

---

19. Scenario: Ignore Binary Files in Search

Question: While searching a directory for the pattern "password", you want to avoid searching through binary files. How would you do this?

Solution:
Use the `-I` option to automatically skip binary files.

```
grep -rI "password" /path/to/search
```

---

20. Scenario: Combine `grep` with `find` to Search Specific File Types

Question: You want to find all `.conf` files in a directory and search for the word "timeout" within those files. How would you combine `find` and `grep` to achieve this?

Solution:
Use `find` to locate `.conf` files and pipe the output to `grep`.

```
find /path/to/search -name "*.conf" | xargs grep "timeout"
```

These scenario-based questions test practical applications of `grep` in real-world tasks and showcase its flexibility in different environments.