Problem Statement

Log Analysis Utility Using Regular Expressions in C#

Scenario

A software company maintains **application log files** that contain system messages, error details, and security-sensitive information such as **password references**.
To automate log validation and analysis, you are required to design a **Log Parsing Utility** using **C# and Regular Expressions**.

The utility must:

- Validate log severity headers
- Split log entries using special delimiters
- Detect and count password occurrences inside quoted text
- Remove unwanted end-of-line markers
- Highlight weak passwords found in log lines

Technical Specifications

1. Namespace Name

LogProcessing

---

2. Class Name

LogParser

---

3. Data Members (Fields)

Data Member Name	Access Modifier	Data Type	Purpose

validLineRegexPattern	private readonly	string	Stores regex to validate log severity
splitLineRegexPattern	private readonly	string	Stores regex used to split log lines
quotedPasswordRegexPattern	private readonly	string	Stores regex to find passwords inside quotes
endOfLineRegexPattern	private readonly	string	Stores regex to remove end-of-line markers
weakPasswordRegexPattern	private readonly	string	Stores regex to detect weak passwords

---

4. Method Definitions & Responsibilities

Task 1: Validate Log Line Format

Method Signature

public bool IsValidLine(string text)

Parameters

- `text` – A single log line

Return Type

- `bool`

What the Method Should Do

- Check whether the log line **starts with a valid severity level**:
  - `[TRC]`, `[DBG]`, `[INF]`, `[WRN]`, `[ERR]`, `[FTL]`

Expected Outcome

- Returns `true` for valid log lines
- Returns `false` for invalid or malformed log lines

Task 2: Split Log Line Using Delimiters

Method Signature

public string[] SplitLogLine(string text)

Parameters

- `text` – A log entry containing special delimiters

Return Type

- `string[]`

What the Method Should Do

- Split the log line using delimiters such as:
  - `<***>`, `<====>`, `<^*>`

Expected Outcome

- Returns an array of log segments after splitting

Task 3: Count Quoted Password Occurrences

Method Signature

public int CountQuotedPasswords(string lines)

Parameters

- `lines` – A block of log text

Return Type

- `int`

What the Method Should Do

- Count how many times the word **password** appears **inside double quotes**
- Case-insensitive match

Expected Outcome

- Returns the total count of quoted password occurrences

Task 4: Remove End-of-Line Markers

Method Signature

public string RemoveEndOfLineText(string line)

Parameters

- `line` – A log line containing end-of-line markers

Return Type

- `string`

What the Method Should Do

Remove text matching the pattern:
end-of-line<number>

-

Expected Outcome

- Returns a clean log line without the marker

Task 5: Identify and Label Weak Passwords

Method Signature

public string[] ListLinesWithPasswords(string[] lines)

Parameters

- `lines` – An array of log lines

Return Type

- `string[]`

What the Method Should Do

Detect weak passwords matching:
password followed by alphanumeric characters

-
- If found, prefix the password to the line
- If not found, prefix with `--------`

Expected Outcome

Input Line	Output
`User password123 failed login`	`password123: User password123 failed login`
`System started successfully`	`--------: System started successfully`

Summary of Student Responsibilities

Students must:

- Create the specified **namespace and class**
- Declare all **private readonly regex fields**
- Implement all **methods with exact signatures**
- Use **System.Text.RegularExpressions**
- Ensure correct return values and string processing
