📝 Basic Regex Problems

1️⃣ Validate a Username

- A valid username:
  - Can only contain **letters (a-z, A-Z), numbers (0-9), and underscores (_)**
  - Must start with a letter
  - Must be **between 5 to 15 characters long**

🔹 **Example Inputs & Outputs**

✅ `"user_123"` → **Valid**

❌ `"123user"` → **Invalid** (starts with a number)

❌ `"us"` → **Invalid** (too short)

```java
import java.util.regex.*;
public class UsernameValidator {
    public boolean isValidUsername(String username) {
        return username.matches("^[a-zA-Z][a-zA-Z0-9_]{4,14}$");
    }
}
```

2️⃣ Validate a License Plate Number

- License plate format: **Starts with two uppercase letters, followed by four digits.**
- Example: `"AB1234"` is **valid**, but `"A12345"` is **invalid**.

```
import java.util.regex.*;

public class LicensePlateValidator {
    public boolean isValidLicensePlate(String plate) {
        return plate.matches("^[A-Z]{2}\t{4}$");
    }
}
```

3️⃣ Validate a Hex Color Code

- A valid **hex color**:
    - Starts with a **#**
    - Followed by **6 hexadecimal characters** (`0-9`, `A-F`, `a-f`).

🔹 **Example Inputs & Outputs**

✅ `"#FFA500"` → **Valid**

✅ `"#ff4500"` → **Valid**

❌ `"#123"` → **Invalid** (too short)

```
import java.util.regex.*;

public class HexColorValidator {
    public boolean isValidHexColor(String color) {
        return color.matches("^#[0-9A-Fa-f]{6}$");
    }
}
```

🔍 Extraction Problems

4️⃣ Extract All Email Addresses from a Text

- 🔹 **Example Text:**

"Contact us at support@example.com and info@company.org"

- 🔹 **Expected Output:**

support@example.com

info@company.org

```
import java.util.regex.*;
import java.util.*;

public class EmailExtractor {
    public List<String> extractEmails(String text) {
        List<String> emails = new ArrayList<>();
        Matcher matcher =
Pattern.compile("[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}").matcher(text
);
        while (matcher.find())
            emails.add(matcher.group());
        return emails;
    }
}
```

5️⃣ Extract All Capitalized Words from a Sentence

- **Example Text:**

"The Eiffel Tower is in Paris and the Statue of Liberty is in New York."

- **Expected Output:**

Eiffel, Tower, Paris, Statue, Liberty, New, York

```
import java.util.regex.*;
import java.util.*;

public class CapitalizedWordExtractor {
    public List<String> extractCapitalizedWords(String text) {
        List<String> words = new ArrayList<>();
        Matcher matcher = Pattern.compile("\\b[A-Z][a-z]+\\b").matcher(text);
        while (matcher.find())
            words.add(matcher.group());
        return words;
    }
}
```

6️⃣ Extract Dates in `dd/mm/yyyy` Format

- **Example Text:**

"The events are scheduled for 12/05/2023, 15/08/2024, and 29/02/2020."

- **Expected Output:**

12/05/2023, 15/08/2024, 29/02/2020

```
import java.util.regex.*;
import java.util.*;

public class DateExtractor {
    public List<String> extractDates(String text) {
```

```
        List<String> dates = new ArrayList<>();
        Matcher matcher =
Pattern.compile("\\b\\d{2}/\\d{2}/\\d{4}\\b").matcher(text);
        while (matcher.find())
            dates.add(matcher.group());
        return dates;
    }
}
```

7️⃣ Extract Links from a Web Page

- **Example Text:**

"Visit https://www.google.com and http://example.org for more info."

- **Expected Output:**

https://www.google.com, http://example.org

```
import java.util.regex.*;
import java.util.*;

public class LinkExtractor {
    public List<String> extractLinks(String text) {
        List<String> links = new ArrayList<>();
        Matcher matcher = Pattern.compile("https?://\\S+").matcher(text);
        while (matcher.find())
            links.add(matcher.group());
        return links;
    }
}
```

🔄 Replace and Modify Strings

8️⃣ Replace Multiple Spaces with a Single Space

- 🔹 **Example Input:**

"This is an example with multiple spaces."

- 🔹 **Expected Output:**

"This is an example with multiple spaces."

```
public class SpaceReplacer {
    public String replaceMultipleSpaces(String text) {
        return text.replaceAll("\\s+", " ");
    }
}
```

9️⃣ Censor Bad Words in a Sentence

- Given a **list of bad words**, replace them with ****.

- 🔹 **Example Input:**

"This is a damn bad example with some stupid words."

- 🔹 **Expected Output:**

"This is a **** bad example with some **** words."

```
import java.util.regex.*;

public class BadWordCensor {
    public String censorBadWords(String text, String[] badWords) {
        for (String word : badWords)
            text = text.replaceAll("\\b" + word + "\\b", "****");
        return text;
    }
}
```

🔥 Advanced Problems

🔟 Validate an IP Address

- A valid **IPv4 address** consists of **four groups of numbers (0-255) separated by dots**.

```
import java.util.regex.*;

public class IPAddressValidator {
    public boolean isValidIP(String ip) {
        return
ip.matches("((25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.){3}(25[0-5]|2[0-4][0-9]|
[01]?[0-9][0-9]?)");
    }
}
```

1️⃣1️⃣ Validate a Credit Card Number (Visa, MasterCard, etc.)

- A **Visa** card number starts with 4 and has **16 digits**.
- A **MasterCard** starts with 5 and has **16 digits**.

```
import java.util.regex.*;

public class CreditCardValidator {
    public boolean isValidCreditCard(String card) {
        return card.matches("^4\\d{15}$") || card.matches("^5\\d{15}$");
    }
}
```

1️⃣2️⃣ Extract Programming Language Names from a Text

- **Example Text:**

"I love Java, Python, and JavaScript, but I haven't tried Go yet."

- **Expected Output:**

Java, Python, JavaScript, Go

```
import java.util.regex.*;
import java.util.*;

public class ProgrammingLanguageExtractor {
    public List<String> extractLanguages(String text) {
        List<String> langs = new ArrayList<>();
        Matcher matcher =
Pattern.compile("\\b(Java|Python|JavaScript|Go)\\b").matcher(text);
        while (matcher.find())
            langs.add(matcher.group());
        return langs;
    }
}
```

1️⃣3️⃣ Extract Currency Values from a Text

- **Example Text:**

"The price is $45.99, and the discount is 10.50."

- **Expected Output:**

$45.99, 10.50

```
import java.util.regex.*;
import java.util.*;

public class CurrencyExtractor {
    public List<String> extractCurrencyValues(String text) {
        List<String> values = new ArrayList<>();
```

```
        Matcher matcher =
Pattern.compile("\\$?\\d+(\\.\\d{2})?").matcher(text);
        while (matcher.find())
            values.add(matcher.group());
        return values;
    }
}
```

1️⃣4️⃣ Find Repeating Words in a Sentence

- **Example Input:**

`"This is is a repeated repeated word test."`

- **Expected Output:**

`is, repeated`

```
import java.util.regex.*;
import java.util.*;

public class RepeatingWordFinder {
    public Set<String> findRepeatingWords(String text) {
        Set<String> repeated = new HashSet<>();
        Matcher matcher = Pattern.compile("\\b(\\w+)\\s+\\1\\b").matcher(text);
        while (matcher.find())
            repeated.add(matcher.group(1));
        return repeated;
    }
}
```

1️⃣5️⃣ Validate a Social Security Number (SSN)

🔹 **Example Input:**

`"My SSN is 123-45-6789."`

🔹 **Expected Output:**

✅ `"123-45-6789"` is **valid**

❌ `"123456789"` is **invalid**

```java
import java.util.regex.*;

public class SSNValidator {
    public boolean isValidSSN(String ssn) {
        return ssn.matches("\\d{3}-\\d{2}-\\d{4}");
    }
}
```
