4

Phone Number Validation: Describe the regular expression pattern that can be used to validate a phone number format. Explain the significance of each component in the pattern.
Explanation: Provide a regular expression pattern that validates phone numbers and break down the components of the pattern, explaining what each part does.

```
^\(?\d{3}\)?[-.\s]?\d{3}[-.\s]?\d{4}$
```

Explanation of Each Component:

1. `^`: This symbol marks the beginning of the string, ensuring that the phone number starts at the start of the input.
2. `\(?\d{3}\)?`: This part validates the area code of the phone number. It allows an optional opening parenthesis "(", followed by three digits (0-9), and an optional closing parenthesis ")".
3. `[-.\s]?`: This section allows for various common phone number separators such as hyphens, dots, or spaces. The question mark makes this part optional.
4. `\d{3}`: This part matches the next three digits of the phone number, which is the exchange code.
5. `[-.\s]?`: Similar to component 3, this part allows for separators and is also optional.
6. `\d{4}`: This segment matches the last four digits of the phone number, which is the subscriber number.
7. `$`: This symbol marks the end of the string, ensuring that the phone number ends at this point in the input.

In summary, this regular expression pattern validates the following aspects of a phone number:

- It starts from the beginning of the input.
- It allows for optional parentheses around the area code.
- It accommodates common separators like hyphens, dots, or spaces between parts of the phone number.
- It ensures the area code consists of three digits.
- It ensures the exchange code consists of three digits.
- It requires the subscriber number to be four digits.
- It validates the phone number format up to the end of the string.

2

URL Validation: Describe the regular expression pattern that can be used to validate a URL format. Explain the significance of each component in the pattern. Explanation: Offer a regular expression that validates URLs and provide an explanation of its components, clarifying their roles in the validation process.

```
^(https?|ftp):\/\/[^\s/$.?#].[^\s]*$
```

Explanation of Each Component:

1. `^`: This symbol denotes the start of the string, ensuring that the URL starts at the beginning of the input.
2. `(https?|ftp)`: This part matches the scheme of the URL, which can be either "http," "https," or "ftp." The question mark after "s" in "https?" makes the "s" character optional.
3. `://`: This is a literal "://" sequence, which separates the scheme from the rest of the URL.
4. `[^\s/$.?#]`: This section matches the domain or hostname of the URL. It ensures that the domain doesn't contain spaces, slashes, question marks, or hash symbols, which are typically not allowed in domain names.
5. `.*`: This part matches the rest of the URL after the domain. The asterisk (*) means zero or more characters. It allows for paths, query parameters, and fragments.
6. `$`: This symbol marks the end of the string, ensuring that the URL ends at this point in the input.

In summary, this regular expression pattern validates the following aspects of a URL:

- It starts from the beginning of the input.
- It checks for the URL scheme, which can be "http," "https," or "ftp."
- It enforces the "://" separator between the scheme and the rest of the URL.
- It ensures that the domain or hostname doesn't contain spaces, slashes, question marks, or hash symbols.
- It allows for various components of the URL after the domain, such as paths, query parameters, and fragments, if present.
- It validates the URL format up to the end of the string.

Credit Card Number Validation: Describe the regular expression pattern that can be used to validate a credit card number format. Explain the significance of each component in the pattern.
Explanation: Present a regular expression pattern for checking the validity of credit card numbers and describe the role of each component within the pattern.

```
^(4[0-9]{15}|5[1-5][0-9]{14}|6(?:011|5[0-9]{14}))$
```

URL Validation: Describe the regular expression pattern that can be used to validate a URL format. Explain the significance of each component in the pattern. Explanation: Offer a regular expression that validates URLs and provide an explanation of its components, clarifying their roles in the validation process.

```
^(https?|ftp):\/\/[^\s/$.?#].[^\s]*$
```

Explanation of Each Component:

1. `^`: This symbol denotes the start of the string, ensuring that the URL starts at the beginning of the input.
2. `(https?|ftp)`: This part matches the scheme of the URL, which can be either "http," "https," or "ftp." The question mark after "s" in "https?" makes the "s" character optional.
3. `://`: This is a literal "://" sequence, which separates the scheme from the rest of the URL.
4. `[^\s/$.?#]`: This section matches the domain or hostname of the URL. It ensures that the domain doesn't contain spaces, slashes, question marks, or hash symbols, which are typically not allowed in domain names.
5. `.*`: This part matches the rest of the URL after the domain. The asterisk (*) means zero or more characters. It allows for paths, query parameters, and fragments.
6. `$`: This symbol marks the end of the string, ensuring that the URL ends at this point in the input.

In summary, this regular expression pattern validates the following aspects of a URL:

- It starts from the beginning of the input.
- It checks for the URL scheme, which can be "http," "https," or "ftp."
- It enforces the "://" separator between the scheme and the rest of the URL.
- It ensures that the domain or hostname doesn't contain spaces, slashes, question marks, or hash symbols.
- It allows for various components of the URL after the domain, such as paths, query parameters, and fragments, if present.
- It validates the URL format up to the end of the string.

Credit Card Number Validation: Describe the regular expression pattern that can be used to validate a credit card number format. Explain the significance of each component in the pattern.
Explanation: Present a regular expression pattern for checking the validity of credit card numbers and describe the role of each component within the pattern.

```
^(4[0-9]{15}|5[1-5][0-9]{14}|6(?:011|5[0-9]{14}))$
```

Explanation of Each Component:

1. `^`: This symbol marks the start of the string, ensuring that the credit card number starts at the beginning of the input.
2. `4[0-9]{15}`: This part matches credit card numbers starting with a "4" (Visa). It's followed by exactly 15 digits (0-9).
3. `|`: The pipe symbol acts as an "OR" operator, allowing multiple credit card patterns to be checked.
4. `5[1-5][0-9]{14}`: This section matches credit card numbers starting with a "5" (MasterCard). It's followed by a second digit between "1" and "5," and then exactly 14 more digits (0-9).
5. `|`: Another "OR" operator to include more credit card patterns.
6. `6(?:011|5[0-9]{14})`: This part matches credit card numbers starting with a "6" (Discover or UnionPay). It allows for two subpatterns. The first one is "011," and the second one is "5" followed by 14 more digits (0-9).
7. `$`: This symbol marks the end of the string, ensuring that the credit card number ends at this point in the input.

In summary, this regular expression pattern validates the following aspects of a credit card number:

- It starts from the beginning of the input.
- It checks for common credit card prefixes for Visa, MasterCard, and Discover/UnionPay.
- It verifies that the credit card number consists of 16 digits.
- It allows for various valid credit card patterns through the "OR" operator.

find the Regular Expression Pattern for matching (TLD)domain names ending in ".co.uk" (UK-specific).

\b\w+\.co\.uk\b

Explanation of Each Component:

1. `\b`: This word boundary anchor ensures that the match occurs at a word boundary, preventing partial matches within longer words.
2. `\w+`: This part matches one or more word characters (letters, digits, or underscores), representing the domain name.
3. `\.co\.uk`: This section matches the specific TLD ".co.uk." The backslashes before the dots (periods) are used to escape them because dots are special characters in regular expressions.