1. Write a program that validates if a user-submitted username is valid. The username must be between **5 and 15 characters long** and contain **only** alphanumeric characters (letters `a-z`, `A-Z` and digits `0-9`).

HINT: Regex Pattern: `^[a-zA-Z0-9]{5,15}$`

How it works:

- `^` and `$` anchor the match to the absolute start and end of the string.
- `[a-zA-Z0-9]` ensures only valid alphanumeric letters or numbers are accepted.
- `{5,15}` enforces the minimum length of 5 and maximum of 15.

2. Given a snippet of raw HTML text, extract and print all the individual HTML opening and closing tags (e.g., , , <p>)

HINT: Regex Pattern : `</?[a-zA-Z0-9]+>`

How it works:

< matches the starting bracket literal.
/? matches zero or one forward slash / (to safely capture both opening tags and closing tags like).
[a-zA-Z0-9]+ matches one or more alphanumeric characters for the tag name.
> matches the closing bracket literal.

3. You have a log file or user input containing North American phone numbers formatted as XXX-XXX-XXXX. For privacy compliance, replace the first two blocks of numbers with asterisks (e.g., transform 123-456-7890 into ***-***-7890)

HINT: Regex Pattern : `\d{3}-\d{3}-(\d{4})`

How it works:

\d{3}- matches exactly 3 digits followed by a hyphen.
The second \d{3}- matches the next 3 digits and a hyphen.
(\d{4}) uses parentheses to form a capturing group for the final 4 digits so we can reference them later.
Replacement String: "***-***-$1" where $1 inserts the exact content matched in the 4-digit capturing group.