UBS

Public

UBS Global Coding Challenge – Hong Kong Edition, 2025

Entry Challenge

The story:
In a galaxy far, far away, on the mystical planet of Greexon, lives an exclusive society of code wizards known as the Greex Guild. The Guild is renowned for their mastery of Gree Expressions, a formal language used to manipulate sequences of characters. Every quarter-century, the Guild seeks out the most exceptional minds to join their coven through a series of challenging tasks.

This year, the Guild has a special mission for prospective candidates. The leader of the Guild, known as the Greexmaster and for their cryptic puzzles, has devised a challenge that will test your ability to formulate Gree Expressions to filter valid character sequences.

The Greexmaster has provided several ancient scrolls containing sample data for you to practice with. Here be dragons! The true test will include dark and mystical data that you must also handle.

The challenge:
Your task is to write a piece of code that generates a Gree Expression. This pattern must match all the valid strings and reject all the invalid ones. The pattern must match the entire string and not just a part of it.

The rules:
1. **No hardcoding**: The Greexmaster forbids hardcoding the examples. You must infer the underlying structure and logic based on the samples provided.
2. **Inference**: You must deduce the hidden patterns and structures from the given samples.
3. **Full match**: The pattern must match the entire string (not just a substring).
4. **Compatibility**: The solution must work in any standard Gree Expression engine.

Practice Scrolls:

Scroll 1:
- Valid Strings: ["abc", "def"]
- Invalid Strings: ["123", "456"]
- Sample Output: ^\D+$

Scroll 2:
- Valid Strings: ["aaa", "abb", "acc"]
- Invalid Strings: ["bbb", "bcc", "bca"]
- Sample Output: ^[a].+$

Scroll 3:
- Valid Strings: ["abc1", "bbb1", "ccc1"]
- Invalid Strings: ["abc", "bbb", "ccc"]
- Sample Output: ^.+[1]$

Scroll 4:
- Valid Strings: ["abc-1", "bbb-1", "cde-1"]
- Invalid Strings: ["abc1", "bbb1", "cde1"]
- Sample Output: ^.+-.+$

Scroll 5:
- Valid Strings: ["foo@abc.com", "bar@def.net"]
- Invalid Strings: ["baz@abc", "qux.com"]
- Sample Output: ^\D+@\w+\.\w+$

Ancient scroll limitations:

1. **Scroll limit**: Each scroll will contain no more than 5 strings, as the ancient parchment can only hold so much information.
2. **String length**: Each string will be no longer than 20 characters, as the Greexmaster's quill has a limited capacity before it needs to be re-inked.
3. **Pattern length**: The Gree Expression must be no longer than 20 characters, as your quill has a limited capacity before it needs to be re-inked.

Your quest:

Write a function called *generate_gree_expression(valid_strings, invalid_strings)* that returns the Gree Expression as a string.
You may write your code in any supported programming language.
You are allowed to make your own assumptions, but you must document them clearly in your code.

The Greexmaster awaits your solution. Prove your worth and earn your place in the Greex Guild. Good luck, and may the code be with you!

Explorer's Diary Entry:

"Day 42 on Greexon. The ancient texts speak of '*gree*,' a term that resonates with our '*regex*.' The Greex Guild's arcane practices in Gree Expressions are both bewildering and enlightening. Their cryptic challenges reveal patterns hidden in the fabric of text. The true essence of '*gree*' remains elusive, yet its power is undeniable. The journey continues, deeper into the enigma."

Note

We only accept implementation in python, java, and javascript.

Any other languages will not be accepted.

Do note that we require your code to be in one single file.

Python

We expect a single file main.py that is able to run the command python main.py.

This file should have a function called "generate_gree_expression" which takes in 2 list of strings as parameters and returns your generated regex string.

Java

We expect a single file Main.java that we can compile and run the command java Main.

This file should have a public function called "generate_gree_expression" which takes in 2 list of strings as parameters and returns your generated regex string.

Javascript

We expect a single file main.js that we can run the command node main.js.

This file should have an exported function called "generate_gree_expression" which takes in 2 list of strings as parameters and returns your generated regex string.

Boilerplate templates

Boilerplate templates for the supported languages have been provided alongside this README.

Submission

Send your code file as zip to global-coding-challenge-hongkong@ubs.com.