Hey Satyaprakash !

Instructions

1. You have **75 mins** to solve all the questions
2. You can solve the questions in **any order** you like
3. All the questions weigh the same
4. You can **use any language** of your choice to solve the questions [BTW we use Golang]
5. You will have to upload the code that you used to solve the problem in the **request body of your submission**. Any submission without code in the body will not be accepted
6. You can take help from any internet resources [google, stack overflow, ...]
7. You are allowed to use third party libraries
8. Join our Discord server at https://discord.gg/YKecYphK for any announcements or updates
9. Once you're done answering the questions, you're done! You can sit back and wait for the results

Questions

1. Find the better location *not attempted*

Implement a program to find the better location based on the condition provided. Make an API call to get the city names and a suitable condition. Use the city names to make another API call get the weather details of the city and determine the better location among them depending on the condition.

- **Condition 1:** hot => return the city with higher temperature
- **Condition 2:** cold => return the city with lower temperature
- **Condition 3:** windy => return the city with higher wind value
- **Condition 4:** rainy => return the city with higher rain value
- **Condition 5:** sunny => return the city with lower cloud value
- **Condition 6:** cloudy => return the city with higher cloud value

`GET City1, City2 and Condition` : https://quest.squadcast.tech/api/RA2111027020045/weather
`GET weather details of a city` :

https://quest.squadcast.tech/api/RA2111027020045/weather/get?q=city_name

Submission

Submission URL: `https://quest.squadcast.tech/api/RA2111027020045/submit/weather?answer={your_answer}&extension={extension_used}`

Make a HTTP **POST** Request to your submission URL with the code in the request body

Example

Let's say city1 (Delhi) has a temperature of 10°C and city2 (Mumbai) has a temperature of 20°C. So, city2 is the better location if the condition is hot. (Assuming python was used to solve the question)

And the code for it is:

```
import .......... return answer
```

So, the Submission URL will look like:

`https://quest.squadcast.tech/api/RA2111027020045/submit/weather?answer=Mumbai&extension=py`

With the code in the request body as raw text:

```
import .......... return answer
```

2. Find the IP address written in words *not attempted*

Implement a program to find the hidden IP Address (written in words) in a passage full of random words provided by the below link. You have to parse the **valid** IP address and written it in the correct IPv4 format e.g. 193.92.89.206

`Link to passage` : https://quest.squadcast.tech/api/RA2111027020045/worded_ip

Submission

Submission URL: `https://quest.squadcast.tech/api/RA2111027020045/submit/worded_ip?answer={your_answer}&extension={extension_used}`

Make a HTTP **POST** Request to your submission URL with the code in the request body

Example

If your passage is:

blame detect supplements shop bradley enhancement hamilton table off herb britain its fri henderson verbal consultancy equation moved soup lauderdale lamp broken gently signals represented approx sullivan paid d hash one nine three point nine two point eight nine point two zero six rentcom inflation roles trustee hull button christians accidents ice governmental

Here hidden IP is : `one nine three point nine two point eight nine point two zero six`

After parsing : `193.92.89.206`

And the code for it is:

```
import .......... return answer
```

So, the Submission URL will look like

`https://quest.squadcast.tech/api/RA2111027020045/submit/worded_ip?answer=193.92.89.206&extension=js`

With the code in the request body as raw text:

```
import .......... return answer
```

3. Decrypt the encrypted. *not attempted*

You will be given a paragraph(string) (encrypted using the Fernet method) and a key(string) to decrypt the paragraph. The paragraph contains emojis in-between the words, emojis are in the form of their unicode representation. Your task is to replace the unicode with their corresponding real emojis. Send the decrypted paragraph containing emojis in the query params with key *answer* and the code in the body of the request.

`GET payload` : https://quest.squadcast.tech/api/RA2111027020045/emoji

Submission

Submission URL: `https://quest.squadcast.tech/api/RA2111027020045/submit/emoji?answer={your_answer}&extension={extension_used}`

Make a HTTP **POST** Request to your submission URL with the code in the request body

Example

Let's assume the answer is:

```
ten eight 😄 nine six five 😆
```

Note: Make sure your answer is URL encoded before sending it as a query param.

And the code for it is:

```
import .......... return answer
```

So, the Submission URL will look like

```
https://quest.squadcast.tech/api/RA2111027020045/submit/emoji?answer=ten eight 😄 nine six five 😆&extension=go
```

With the code in the request body as raw text:

```
import .......... return answer
```