Mercado Libre DataSec Technical Challenge

Instructions

- Create a separate file for each question.
- Commit all the solutions to a single public GitHub repository.
- Solve all the questions using Python 3. Specify in the comments the exact version of Python you used.

---

1. Minesweeper Number of Neighbouring Mines

Your solution for this challenge must be in a file named `solution_minesweeper.py`.

Create a function that takes a list representation of a Minesweeper board, and returns another board where the value of each cell is the amount of its neighbouring mines.

Example

Input:

```
[
  [0, 1, 0, 0],
  [0, 0, 1, 0],
  [0, 1, 0, 1],
  [1, 1, 0, 0]
]
```

- The `0` represents an empty space.
- The `1` represents a mine.

Expected Output:

```
[
  [1, 9, 2, 1],
  [2, 3, 9, 2],
  [3, 9, 4, 9],
  [9, 9, 3, 1]
]
```

Notes

- Since in the output the numbers 0-8 are used to determine the amount of adjacent mines, the number 9 will be used to identify the mines instead.
- You can use the Wikipedia page explaining how Minesweeper works.

---

2. REST API: Best TV Shows in Genre

Your solution for this challenge must be in a file named `solution_best_in_genre.py`.

Use the HTTP GET method to retrieve information about recent television shows.
Query the API:

```
https://jsonmock.hackerrank.com/api/tvseries
```

The query result is paginated. To access additional pages, append `?page={num}` to the URL where `num` is the page number.

Response JSON Schema

```
{
  "name": "Game of Thrones",
  "runtime_of_series": "(2011–2019)",
  "certificate": "A",
  "runtime_of_episodes": "57 min",
  "genre": "Action, Adventure, Drama",
  "imdb_rating": 9.3,
  "overview": "...",
  "no_of_votes": 1773458,
  "id": 1
}
```

Task Challenge

Given a genre, find the series with the **highest** `imdb_rating`. If there is a tie, return the **alphabetically lower name**.

Function Description

```
def bestInGenre(genre: str) -> str:
    ...
```

- **Parameter:**
  - `genre` (str): the genre to search
- **Return:**
  - `str`: the highest-rated show in the genre, with the lowest name alphabetically if there is a tie

Sample

Input: `Action` **Output:** `Game of Thrones`

Sample Explanation

The 4 highest-rated shows in the `Action` genre are:

- Game of Thrones — 9.3
- Avatar: The Last Airbender — 9.2
- Hagane no renkinjutsushi — 9.1
- Shingeki no kyojin — 8.9

---

3. SQL: Advertising System Failures Report

As part of HackerAd's advertising system analytics, a team needs a list of customers who have a maximum number of failure events (`status = "failure"`) in their campaigns.

Requirements

- For all customers with more than 3 events with `status = 'failure'`, report the customer name and their number of failures.
- The result should be in the following format:

customer	failures
Whitney Ferrero	6

Table Schemas

Table: `customers`

Column	Type	Description
id	SMALLINT	Customer ID
first_name	VARCHAR(64)	Customer first name
last_name	VARCHAR(64)	Customer last name

Column	Type	Description

Table: `campaigns`

Column	Type	Description
id	SMALLINT	Campaign ID
customer_id	SMALLINT	Customer ID
name	VARCHAR(64)	Campaign name

Table: `events`

Column	Type	Description
dt	VARCHAR(19)	Event timestamp
campaign_id	SMALLINT	Campaign ID
status	VARCHAR(64)	Event status

Sample Data

customers

id	first_name	last_name
1	Whitney	Ferrero
2	Dickie	Romera

campaigns

id	customer_id	name
1	1	Upton Group
2	1	Roob, Hudson and Rippin
3	1	McCullough, Rempel and Larson
4	1	Lang and Sons
5	2	Ruecker, Hand and Haley

events

dt	campaign_id	status
2021-12-02 13:52:00	1	failure
2021-12-02 08:17:48	2	failure
2021-12-02 08:18:17	2	failure
2021-12-01 11:55:32	3	failure
2021-12-01 06:53:16	4	failure
2021-12-02 04:51:09	4	failure

dt	campaign_id	status
2021-12-01 06:34:04	5	failure
2021-12-02 03:21:18	5	failure
2021-12-01 03:18:24	5	failure
2021-12-02 15:32:37	1	success
2021-12-01 04:23:20	1	success
2021-12-02 06:53:24	1	success
2021-12-02 08:01:02	2	success
2021-12-01 15:57:19	2	success
2021-12-02 16:14:34	3	success
2021-12-02 21:56:38	3	success
2021-12-01 05:54:43	4	success
2021-12-02 17:56:45	4	success
2021-12-02 11:56:50	4	success
2021-12-02 06:08:20	5	success

Expected Output:

customer	failures
Whitney Ferrero	6

---

4. Go CLI: Text Summarizer with GenAI

Your solution for this challenge must be in a file named `solution_summarizer.go`.

Create a Go command-line application that summarizes the contents of a text file using a free, public GenAI API (such as HuggingFace Inference API or any other public endpoint).

Requirements

- The CLI must be written in Go.
- The CLI must accept:
  - `--input` or a positional argument: path to the text file to summarize.
  - `--type` or `-t`: summary type, one of `short`, `medium`, or `bullet`.
- The CLI must call a free, public GenAI API for summarization (e.g., HuggingFace Inference API).
- The prompt sent to the API should be engineered to match the summary type:
  - `short`: a concise summary (1-2 sentences)
  - `medium`: a paragraph summary
  - `bullet`: a list of bullet points
- The CLI should output the summary to stdout.
- The CLI should handle API errors gracefully and print user-friendly messages.
- Document the Go version used in your code comments.

Functionality Example

```
go run solution_summarizer.go --input article.txt --type bullet
```

or

```
go run solution_summarizer.go -t short article.txt
```

Sample Output

```
- Point 1: ...
- Point 2: ...
- Point 3: ...
```

or

```
This article discusses ...
```

Evaluation Criteria

- Correctness of CLI argument parsing.
- Proper use of Go HTTP client for API calls.
- Prompt engineering: how the summary type is reflected in the prompt.
- Error handling and code clarity.
- Documentation/comments in the code.

API Suggestions

- You may use the HuggingFace Inference API (free tier, no key required for some models) or any other public GenAI summarization endpoint.
- Provide a link to the API documentation in your code comments.