Week-2: Tasks List

- Project Proposal (title, team-members, Users, modules/screens) dueDate:15/9/2023
- An Introduction to Flutter (chapter 3)
- Flutter - Installation
- Flutter: Creating Your First App

  https://docs.flutter.dev/get-started/install
  https://docs.flutter.dev/reference/flutter-cli

- **Practice Dart – Exercises**

https://hackmd.io/@kuzmapetrovich/S1x90jWGP#Practice-Dart---exercises-for-beginners

Exercise 11

Write a program that takes a list of numbers for example

```
a = [5, 10, 15, 20, 25]
```

and makes a new list of only the first and last elements of the given list. For practice, write this code inside a function.

Exercise 12

Write a program that asks the user how many Fibonnaci numbers to generate and then generates them. Take this opportunity to think about how you can use functions.

Make sure to ask the user to enter the number of numbers in the sequence to generate.

Exercise 13

Write a program (function) that takes a list and returns a new list that contains all the elements of the first list minus all the duplicates.

Exercise 14

Write a program (using functions!) that asks the user for a long string containing multiple words. Print back to the user the same string, except with the words in backwards order.

For example, say I type the string:

```
  My name is Michele
```

Then I would see the string:

```
  Michele is name My
```

Exercise 15

Write a password generator in Dart. Be creative with how you generate passwords - strong passwords have a mix of lowercase letters, uppercase letters, numbers, and symbols. The passwords should be random, generating a new password every time the user asks for a new password. Include your run-time code in a main method.

Ask the user how strong they want their password to be. For weak passwords, pick a word or two from a list.

Exercise 16

Create a program that will play the "cows and bulls" game with the user. The game works like this:

- Randomly generate a 4-digit number. Ask the user to guess a 4-digit number. For every digit the user guessed correctly in the correct place, they have a "cow". For every digit the user guessed correctly in the wrong place is a "bull."
- Every time the user makes a guess, tell them how many "cows" and "bulls" they have. Once the user guesses the correct number, the game is over. Keep track of the number of guesses the user makes throughout the game and tell the user at the end.

Exercise 17

Time for some fake graphics! Let's say we want to draw game boards that look like this:

```
 --- --- ---
|   |   |   |
 --- --- ---
|   |   |   |
 --- --- ---
|   |   |   |
 --- --- ---
```

This one is 3x3 (like in tic tac toe).

Ask the user what size game board they want to draw, and draw it for them to the

screen using Dart's print statement.

Exercise 18

As you may have guessed, we are trying to build up to a full tic-tac-toe board.
For now, we will simply focus on checking whether someone has WON the game, not worrying about how the moves were made.

If a game of Tic Tac Toe is represented as a list of lists, like so:

```
game = [[1, 2, 0],
        [2, 1, 0],
        [2, 1, 1]]
```

where a 0 means an empty square, a 1 means that player 1 put their token in that space, and a 2 means that player 2 put their token in that space.

Your task: *given a 3 by 3 list of lists that represents a Tic Tac Toe game board, tell whether anyone has won, and tell which player won, if any.*

A Tic Tac Toe win is 3 in a row - either in a row, a column, or a diagonal. Don't worry about the case where TWO people have won - assume that in every board there will only be one winner.

Exercise 19

In a previous exercise we explored the idea of using a list of lists as a "data structure" to store information about a tic tac toe game. In a tic tac toe game, the "game server" needs to know where the Xs and Os are in the board, to know whether player 1 or player 2 (or whoever is X and O) won.

*There has also been an exercise (**17**) about drawing the actual tic tac toe gameboard using text characters.*

*The **next logical step** is to deal with handling user input. When a player (say player 1, who is X) wants to place an X on the screen, they can't just click on a terminal. So you are going to approximate this clicking simply by asking the user for a coordinate of where they want to place their piece.*

Exercise 20

In 3 previous exercises, we built up a few components needed to build a Tic Tac Toe game in Dart:

1. Draw the Tic Tac Toe game board
2. Checking whether a game board has a winner
3. Handle a player move from user input

The next step is to put all these three components together to make a two-player Tic Tac Toe game!

Your challenge in this exercise is to use the functions from those previous exercises all together in the same program to make a two-player game that you can play with a friend. There are a lot of choices you will have to make when completing this exercise, so you can go as far or as little as you want with it.

Here are a few things to keep in mind:

- You should keep track of who won - if there is a winner, show a congratulatory message on the screen.
- If there are no more moves left, don't ask for the next player's move!

Keep in mind, the current solution is not just a copy pase of functions from the previous exercises, but rather a rework of them