The Final Task

1. Do a **Debian installation** of the following libraries **(Hint Slide 20)**
    - ros-foxy-v4l2-camera
    - libopencv-dev
    - python3-opencv
2. Run the camera node with executable v4l2_camera_node in the package v4l2_camera **(Hint Slide 22)**
3. Create a workspace with a **python package** named `camera_test` **and node name** `camera_viewer`. **(Hint Slide 26)**
4. In the `camera_test` folder in the package, copy the script located in the next page in `camera_viewer.py`.
5. Create a subscriber that subscribes to the topic published by the camera node representing the **raw image (Hint Slide 58)**
    - Use a ros2 topic command to find out the topics being published when you run the camera node **(Hint Slide 62)**
    - Use a ros2 topic command to find out the message type of the topic being published **(Hint Slide 62)** **(You can omit the *sensor_msgs/msg/* part of the message type for this parameter)**
    - Set the message queue size to 10
    - Set the callback as the callback function already available in the script.
6. Build your workspace **(Hint Slide 35)**
7. Source your workspace **(Hint Slide 38)**
8. Run the camera node and the camera viewer node. You should now see a popup window displaying your webcam output **(Hint Slide 22)**

```
import rclpy
import cv2
import numpy as np

from rclpy.node import Node
from std_msgs.msg import String
from sensor_msgs.msg import Image

class MinimalSubscriber(Node):

    def __init__(self):
        super().__init__('minimal_subscriber')

        #Add your subscriber here

    def listener_callback(self, msg):
        np_img = np.reshape(msg.data, (msg.height, msg.width, 3)).astype(np.uint8)
        self.display(np_img)

    def display(self, img : np.ndarray):
        cv2.imshow("img", cv2.cvtColor(img, cv2.COLOR_RGB2BGR))
        cv2.waitKey(1)


def main(args=None):
    rclpy.init(args=args)

    minimal_subscriber = MinimalSubscriber()

    rclpy.spin(minimal_subscriber)

    # Destroy the node explicitly
    # (optional - otherwise it will be done automatically
    # when the garbage collector destroys the node object)
    minimal_subscriber.destroy_node()
    rclpy.shutdown()


if __name__ == '__main__':
    main()
```

The Final Task (Part 2)

1. Create a launch folder in the camera test package folder
2. Create a launch file, camera_display.launch.py
3. Copy the following in the launch file

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    ld = LaunchDescription()
    #TODO#complete your code to call both v4l2_camera_node and py_pubsub listener here#

    return ld
```

4. Create two different nodes **(Hint Slide 93)**
    a. The first node will be from the camera_test package with executable camera_viewer.
    b. The second node will be from the v4l2_camera package with executable v4l2_camera_node.
5. Make sure to add each node to the `LaunchDescription` instance with the `add_action` function **(Hint Slide 93)**
6. Build the workspace **(Hint Slide 35)**
7. Source the workspace **(Hint Slide 38)**
8. Launch the launch file. There will be the popup with your webcam video playing. **(Hint Slide 92)**