iFollow Challenge

In this challenge, you will find some tasks which will test your computer science, computer vision, ROS, automatic control, creativity and technical organizational skills. Answers both in France and English are accepted. Make an effort to provide some thoughts about the problems and how to begin to solve them even if you don't really know the right/complete answers. Be clear and always explain your reasoning.

1. **[Algorithmics & logistics]** You work for a logistics company and you are entrusted with the task of optimizing shipments. You have to charge goods into a cargo truck so that you transport the maximum total value per shipment. Each one of the 7 items (A through G) has a value and a volume (c.f. Table below) and the cargo truck has a maximum volume (capacity) of 15 units of volume. There is no partial volume (e.g., you cannot ship 60% of product C).

	A	B	C	D	E	F	G
value	7	9	5	12	14	6	12
volume	3	4	2	6	7	3	5

   a. Find the subset of products of maximum total value such that the sum of their volumes is at most 15 (they all fit into the truck).

   b. Now imagine that instead of 7 products and a little 15 [units of volume] truck you have thousands of products and hundreds of units of volume in your cargo truck. How does the method you used to solve the previous question scale with the growth of the number of items and total volume capacity? Is it efficient regarding computational time? Is it efficient regarding memory use? Use Big O notation for those complexity analysis. If your previous approach is not very efficient try to provide one that is.

c. Code your solution for solving the problem for an arbitrary value of **capacity** and arbitrary arrays of **values** and **volumes**.

Bonus: Time your solution and that from Google OR-Tools (https://developers.google.com/optimization/bin/knapsack) for the same dataset and show you can beat Google!

3. **[ROS basics]** It is well known that ROS C++ is faster than ROS Python. Is it really true? Create a ROS catkin package implementing 4 combinations of ROS publisher/subscriber illustrating the difference in performance using a graph/picture or any other visual tool you prefer.

    a. C++ publisher and python subscriber

    b. C++ publisher and C++ subscriber

    c. Python publisher and python subscriber

    d. Python publisher and C++ subscriber

    The implementation could involve a simple string publisher to a topic with your favourite name. Make sure the publisher is dynamically reconfigurable to change the publishing rate online. Measure the subscriber loop rate for different publisher rate to illustrate the performance difference

    A. Find out the point when the measured value is different from real values in each of the combinations (a,b,c,d) from a set of different publishing rates.

    B. Make some comments about what you understand from this performance difference.

    C. Explain the implementation difference between roscpp and rospy resulting in performance difference.

    Deliverable:

    - A private git repo with your performance test package
    - Comments or explanations or answers in a README.md file.

4. **[ROS Computer Vision]:**

A. Apply canny edge filter to the images grabbed from your web camera or your favourite videos. Create a ros package to apply the canny edge filter and publish the filtered images so that it can be seen in rqt or any other visualizer. **cv_camera** from ROS could be used to grab the images to quickly go forward in the task. Use any programming language you wish.

Deliverables:

- A private git repo with your performance test package
- Comments or explanations or answers in a README.md file.

5. **[Tele-operated robot with a joystick]** A gamer controller input can be represented as two arrays, one of buttons values (booleans) and one of axes values (floats). Each joystick of a controller is represented by two axes (X, Y). You have to use one of the controller's joystick to control a differential drive mobile robot movements in the 2D plane. Its wheels have a maximum rotation velocity of W_{max} (rad/s).

   A. Write a snippet of Python or C++ code to convert any possible value of X and Y of that joystick to sensible rotation velocities (rad/s) for right and left wheels of that differential drive mobile robot.

   B. Write another snippet for the same purpose but this time the footprint of the mobile robot and the wheels radius (**R**) are known and you want to guarantee that the instantaneous velocity of any point within the robot does not ever exceeds V_{max} | $V_{max} = R\ W_{max}$. The footprint is described by a convex polygon stored as an array of (x, y) tuples, and the center of rotation of the robot coincides with the geometrical center of the footprint.

   C. Since this joystick will be used by a person and that people usually prefer their robots to move forward/backwards more than to turn around themselves, tweak your previous answer in a way that along the axes X and Y the behaviour remains the same as before, but when

going from (X=0, Y=±1) to (X=±1, Y=0) along the joystick outer circle you have a more “slow” transition from the robot moving along a line to it turning around it self.