• U  C •

Universidade de Coimbra
Mestrado em Engenharia Informática
Qualidade e Confiabilidade de Software 2024/25

Assignment 1

In this assignment, we will consider the electronic control system of an automobile (i.e., a car). Nowadays, most of the functionality of a car is implemented using software. This means that the codebase of modern cars reaches millions of lines of code and many microprocessors (sometimes over 100) are required to execute this software. Hence, the architecture of a car is a complex system of interconnected components, where a failure in one of them may cause death, harm or damage to the user or the environment.

Despite the careful design and implementation of these systems, which often employs stringent software development practices, extensive testing and certification, faults (e.g., software bugs, hardware faults) can nevertheless occur. Thus, **fault tolerance** has to be used, depending on the criticality level of each component.

In this assignment, you are asked to:

- Choose among a pool of software functional requirements
- Implement two of those requirements in your language of choice (C/C++, Java, Python)
- Improve your solution, by including fault tolerance, according to the criticality of the task and the faults that you consider likely to occur
- Write a report, describing your work and justifying your decisions

By the end of the assignment, you should:

- Know the different types of fault tolerance approaches and their advantages/disadvantages
- Be able to apply fault tolerance in practice
- Be capable of justifying your design choices based on concrete factors

The assignment expects you to research and learn about the topic of fault tolerance. You can use the slides of the T class as a starting point, however these will not be enough, and you are advised to look for more content outside the materials provided in the course.

The output of this assignment, which you are required to submit as a deliverable, is:

A. Source code of a baseline solution (without fault tolerance)

B. Source code of an improved solution (with fault tolerance)

C. A report, with fewer than 2000 words, in PDF format, which should include:

   - a list of the considered fault tolerance mechanisms (including those that were not implemented, if any)
   - an explanation of the applied fault tolerance mechanisms
   - if applicable, a justification to why some fault tolerance mechanisms were not implemented
   - a justification to why the applied fault tolerance mechanism were chosen
   - a description of the considered fault models

- the division of work/tasks among the members of the group
- the list of references (external resources, websites, books, research papers) that were used for the completion of this assignment

The deadline of this assignment is **April 04** until 23h59, in Inforestudante. However, before, you should submit a document (PDF or TXT) containing the following elements:

- Name and student numbers of the two group elements
- List of 5 software requirements, in order of preference
- List of programming languages that the group is comfortable with (unless discussed with the teacher, this list should include one or more from C, C++, Java and Python)

This document shall be submitted until **March 10**, in Inforestudante, and will be used by the teacher to assign the software requirements to each group, as to ensure an even balance. Each group will be assigned two requirements to implement, according to the preferences provided by the group, and according to the submission date (that is, groups that submit this document earlier will be more likely to get the requirements that they prefer).

Defenses will take place on the week after the deadline, in slots that will be open in Inforestudante (and each group must register for), most of these during the PL classes. Grading is dependent on the **individual** performance during defense and will take into account the quality of the baseline code (to a lesser extent), the quality and adequacy of the improved code, the report and the performance during the defense. Plagiarism (of either code or report) will be penalized according to the rules of the faculty and may result in a fail at the course.

1 Functional requirements specification

1.1 FR1 - Emergency and anti-lock braking system

The system must feature an anti-lock braking system, so that when one wheel loses traction, the braking pressure is reduced. Wheel traction is calculated by comparing vehicle speed (in km/h) against wheel speed (also in km/h), to obtain *wheel slip*. As long as, at least one of the four wheels has a slip higher than a certain value (passed as a parameter), the vehicle is considered to be losing traction and braking pressure must be reduced in half.
Furthermore, the system must feature emergency braking, so that when vehicle speed is equal or higher than 30 km/h and brake pressure is equal or higher than 40%, if there is an increase in brake pressure that is 30% than the average brake pressure, then brake pressure must be forced to 100%.
This functionality must be implemented by a function that receives the required parameters and returns an integer value, ranging from 0 to 100, which represents the brake pressure that should be applied.
For reference, your code must pass the following test cases:

ID	Input	Expected Output
TC1	vehicle_speed: 0 wheel_speeds: [] slip_threshold: 0.5 current_brake_pressure: 0 prev_brake_pressures: []	0
TC2	vehicle_speed: 30 wheel_speeds: [30, 30, 30, 30] slip_threshold: 0.5 current_brake_pressure: 40 prev_brake_pressures: [0]	100
TC3	vehicle_speed: 10 wheel_speeds: [10, 10, 10, 10] slip_threshold: 0.5 current_brake_pressure: 40 prev_brake_pressures: [0]	40
TC4	vehicle_speed: 10 wheel_speeds: [10, 10, 10, 10] slip_threshold: 0.5 current_brake_pressure: 40 prev_brake_pressures: [35, 36, 37, 41, 42]	40
TC5	vehicle_speed: 10 wheel_speeds: [10, 10, 0, 10] slip_threshold: 0.2 current_brake_pressure: 20 prev_brake_pressures: [20, 20, 20]	10
TC6	vehicle_speed: 10 wheel_speeds: [10, 0, 0, 10] slip_threshold: 0.2 current_brake_pressure: 20 prev_brake_pressures: [20, 20, 20]	10

Tabela 1: Test cases for FR1

1.2 FR2 - Cruise control

The system must feature cruise control functionality. The user must be able to set a desired speed (in km/h) and an acceptable error (in km/h), and the system must accelerate or decelerate as to keep within the desired range. To avoid jerky motions, it must be possible to define a maximum acceptable acceleration.
This functionality must be implemented by a function that receives the required parameters and returns a floating-point value, which represents the acceleration (or deceleration) that should be applied. If the current speed is already within the desired speed range, then no acceleration (0) should be returned.
For reference, your code must pass the following test cases:

ID	Input	Expected Output
TC1	current_speed: 100 prev_speed: [100, 100] set_speed: 100 max_accel: 10 error: 10	0
TC2	current_speed: 100 prev_speed: [100, 100] set_speed: 100 max_accel: 10 error: 0	0
TC3	current_speed: 100 prev_speed: [100, 100] set_speed: 120 max_accel: 10 error: 0	-10
TC4	current_speed: 100 prev_speed: [100, 100] set_speed: 80 max_accel: 30 error: 0	20
TC5	current_speed: 100 prev_speed: [100, 100] set_speed: 80 max_accel: 10 error: 21	0

Tabela 2: Test cases for FR2

1.3 FR3 - Tyre pressure warning

The system must control tyre pressure and produce a warning when the pressure in a tyre (in PSI) is lower than a predefined value. To avoid intermittent warnings (i.e., warnings that appear and disappear quickly) when pressures are close to the predefined value (and can vary due to heat and other factors), the system should receive a list of previous pressure readings, calculate the average pressure and consider that value.
Sometimes the tyre pressure may malfunction, when this failure of the sensor is detected via hardware, the reading is set to 0 PSI. If a reading of 0 PSI is received, the system should ignore the reading.
This functionality must be implemented by a function that receives the required parameters (namely, the current and past tyre pressure readings and the target pressure level below which a warning is thrown) and returns a list with the tyres that, ordered from the tyre with the lowest pressure to the tyre with the highest pressure.
For reference, your code must pass the following test cases:

ID	Input	Expected Output
TC1	pressures: [[10], [10], [10], [10]] target_pressure: 5	[]
TC2	pressures: [[10, 10], [10, 10], [10, 10], [10, 10]] target_pressure: 5	[]
TC3	pressures: [[10, 10], [30, 30], [20, 20], [40, 40]] target_pressure: 100	[0, 2, 1, 3]
TC4	pressures: [[10], [1], [10], [10]] target_pressure: 5	[1]
TC5	pressures: [[0, 15], [10, 10], [10, 10], [10, 10]] target_pressure: 8	[]

Tabela 3: Test cases for FR3

1.4 FR4 - Reverse gear proximity sensor

The system must detect and alert the driver to nearby objects when in reverse gear. To do so, the system features various sensors that return the distance (in cm) to an object in a straight line. As the vehicle gets closer to objects, the system will produce an audible beep, which will increase in frequency and tone the closer objects are.
This functionality must be implemented by a function that should receive the distance to objects as obtained from the sensors and a list of beep levels according to the distance to the nearest object. The function should return an integer value, corresponding to the beep level that has to be played in the vehicle's speakers. For this functionality to operate, it requires at least 2 sensors. If the system has less than two active sensors, then the system cannot operate and the function must return -1.
For reference, your code must pass the following test cases:

ID	Input	Expected Output
TC1	distances: [ ] levels: [ ]	-1
TC2	distances: [100, 100] levels: [5, 10, 20, 40, 70]	4
TC3	distances: [30, 100] levels: [5, 10, 20, 40, 70]	2
TC4	distances: [1, 7, 13, 25, 45, 80] levels: [5, 10, 20, 40, 70]	0
TC5	distances: [2, 4, 5, 5] levels: [1]	0

Tabela 4: Test cases for FR4

1.5 FR5 - Object detection in camera

The system must detect objects from the input received from a rear-facing camera. The camera returns a matrix of $w * h$ floating point numbers in the 0 to 1 range (grayscale). The system should group pixels that are connected (vertically and horizontally, but never diagonally) and above a certain threshold value. Table 5 shows an example of a 3 x 3 input that contains 2 different objects if we consider the threshold value to be 0.5.

1.0	1.0	0.0
1.0	1.0	0.0
0.0	0.0	1.0

Tabela 5: Example of a 3 x 3 input matrix

This functionality must be implemented by a function that should receive a matrix, the width and the height of the camera input, and the threshold value. The function should return the count of unique objects (that is, groups of neighboring pixels above the threshold value) in the image.
For reference, your code must pass the following test cases:

ID	Input	Expected Output
TC1	image: [[0.2, 0.2], [0.2, 0.2]] width: 2 height: 2 threshold: 0.1	1
TC2	image: [[0.2, 0.2], [0.2, 0.2]] width: 2 height: 2 threshold: 0.3	0
TC3	image: [[0.2]] width: 1 height: 1 threshold: 0.1	1
TC4	image: [[1.0, 0.0, 0.0], [0.0, 0.0, 0.0], [0.0, 0.0, 1.0]] width: 3 height: 3 threshold: 0.1	2
TC5	image: [[1.0, 1.0, 1.0], [1.0, 1.0, 1.0], [1.0, 1.0, 1.0]] width: 3 height: 3 threshold: 0.1	1

Tabela 6: Test cases for FR5

1.6 FR6 - Climate control

The system must monitor the temperature within the passenger compartment and keep it within the user desired range. When the temperature is too cold, the cooling fans (AC) should be turned down, while when the temperature is too hot, the fans should be turned up. Within the passenger compartment there are various zones (e.g., the front seats, the rear seats) and each zone has a set of cooling fans associated to it.
This functionality must be implemented by a function that receives the desired temperature (in Celsius), the allowable error margin, the temperature readings from the sensors of the various zones inside the car and a list with the current duty cycle (a percentage from 0% to 100%) of each fan of every zone of the passenger compartment. If the temperature reading of a zone is outside the allowable range (i.e., desired temperature plus or minus the allowable error) then the fans of that zone should be increased (if the temperature is too hot) or decreased (if the temperature is too low) as follows:

- if the average duty of the fans in the zone is 0%, increase or decrease by 20%;
- otherwise, increase or decrease by 10%

The function should return the new duty cycle of the fans.
For reference, your code must pass the following test cases:

ID	Input	Expected Output
TC1	desired_temp: 25 error: 3 temps: [23] fans: [[0, 0, 0]]	[[0, 0, 0]]
TC2	desired_temp: 25 error: 3 temps: [20, 18] fans: [[10, 10], [5, 5]]	[[20, 20], [15, 15]]
TC3	desired_temp: 25 error: 3 temps: [20, 18] fans: [[10, 0], [5, 5]]	[[20, 10], [20, 20]]
TC4	desired_temp: 15 error: 3 temps: [20, 18] fans: [[10, 0], [0, 0]]	[[0, 0], [0, 0]]
TC5	desired_temp: 30 error: 3 temps: [20, 18] fans: [[95, 70], [100, 0]]	[[100, 80], [100, 10]]

Tabela 7: Test cases for FR6