Design and Integration of Embedded Systems exam SOLUTION **January 13, 2021.**

Available time: 60 minutes

Maximum points: 24. Minimum points needed: 12 points

Please start solving each exercise on a separate page, writing first your name and Neptun code.

Exam topics: System Design (12 points)

1. You are working for an embedded software company. Your company executes project planning and control, requirements management and tracking, and many other processes, but it does not have configuration management process.

   a. Which approach of CMMI could you use to add this process area to the company workflow? **(1 point)**

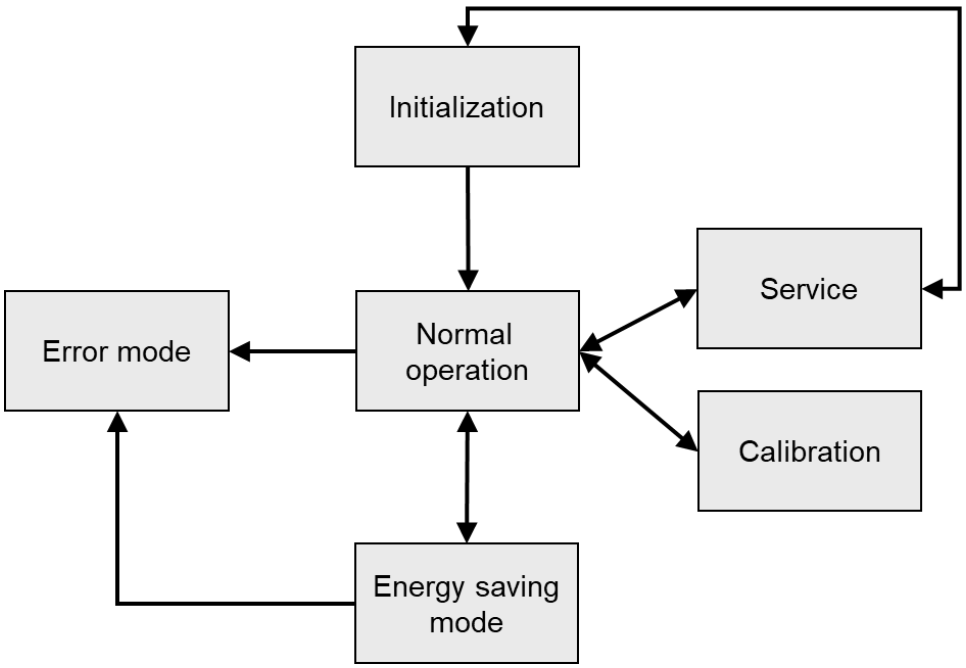
   Because the company has processes, the continuous representation is recommended.

   b. Which level would be your goal to achieve in a year, and why? **(1 point)**

   Any of the first 3 capability levels is acceptable with good reasoning.

2. You are the software architect of an embedded system company. You company is designing a wireless, battery powered, non-safety-critical outdoor sensor unit (to measure temperature, wind speed etc.). You need to plan the main software operating states of the device. Draw the main software operating states, their connection, and give their purpose in one or two sentences. **(2 points)**

   Something like this (not all of the states are needed for full points):



- Initialization: Setup the sensors, and communication parameters
- Normal: Normal measurements
- Energy saving: Set the sensor and the microcontroller to power save state to endure battery life
- Service: Enabling software update, or other service type settings
- Calibrations: If there are sensors that need calibration time to time, we have to support it.
- Error: After some kind of error happens, go to a safe, error signaling state.

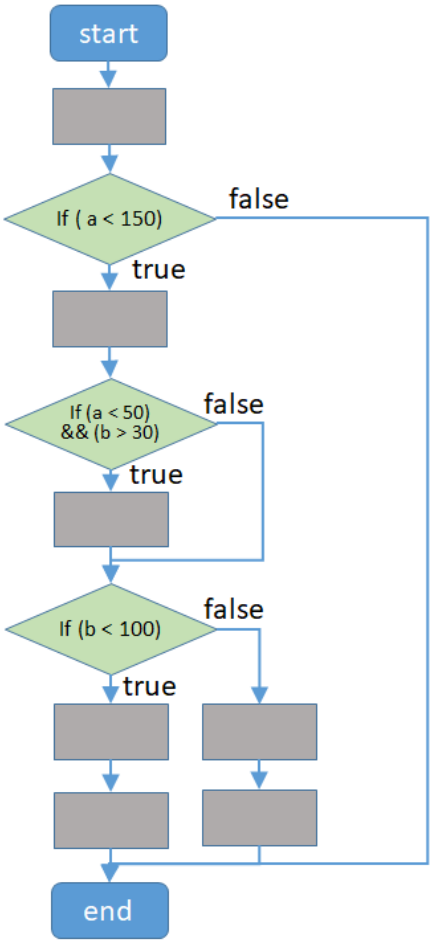
3. As a tester in an embedded system company, you have to test the following software module. The software module is part of a safety critical system. The module has no memory, it does not use a global variable.

```
// returns the result of the calculation, return value range 0 - 255
uint8_t safetyCriticalCalc(uint8_t a,     // first parameter of the calculation 0 - 255
                uint8_t b) // second parameter of the calculation value range 0 - 255
```

a. How many test cases do you need to perform an Exhaustive testing? **(1 point)**

The module has no memory, therefore we need to test the input combinations: 256 * 256 = 65,536 test cases.

b. You got the source code of the module, which has the following flowchart (on the right). What *types* of coverage tests could be executed on this module, and minimum how many test cases do these coverage test types need to achieve a 100% coverage? Give the parameter values **a** and **b** for each *test case*! (No more than 3 different coverage types are needed, you don't have to describe more even if you know more.) **(6 points)**



We can use for example statement, branch, and MC/DC coverage.

Statement coverage example test cases (min. 2):

Test Case	A	B
1	30	50
2	30	120

Branch coverage example test cases (min. 3):

Test Case	A	B
1	30	50
2	200	don't care
3	80	120

MC/DC coverage example test cases (min. 4):

Test Case	A	B
1	200	don't care
2	30	50
3	80	120
4	30	12

c. Which coverage type would you select for testing a highly safety critical module and why? **(1 point)**

Usually MC/DC coverage is used, but anything stronger than statement coverage can be accepted.

Exam topics: Safety Critical Systems (12 points)

4. We have to provide fault tolerance in case of *transient hardware faults* of a microcontroller that executes our application software. In order to do this, we have only an external non-volatile (flash) memory module; the microcontroller is able to write and read data to/from this memory.

   a. Describe *two possible methods* that can be used in this case to implement the *recovery* that is necessary for fault tolerance. **(2 points)**

   Two possible methods:

   - Backward recovery: The state of the application is saved regularly to the external non-volatile memory (as checkpoint). After detecting an error, the latest checkpoint is loaded from the external memory and the application continues operation from the restored state. If it turns out that the restored state is faulty, then a previous checkpoint is loaded. The external non-volatile memory is used as a checkpoint memory.
   - Forward recovery: After detecting an error, selective correction is applied: depending on the result of the error detection and damage assessment, a new predefined state is set from which the operation can be continued. In this case the external non-volatile memory is not used.

   b. Give *at least one characteristic* of the application that determines which one (out of the described two methods) can be applied! **(1 point)**

   Possible characteristics of the software application that may determine the selection:

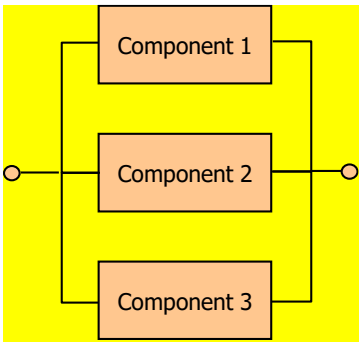
   - If it is possible to define a state (depending on the detected error and estimated damage) from which the application can be safely continued (e.g., reading input sensors and recomputing the state variables in the controller) then forward recovery can be applied.

     If there is no such state (e.g., in case of a stateful application) then backward recovery has to be applied.
   - If the size of the external non-volatile (flash) memory module is too small to store a checkpoint belonging to the application, then backward recovery cannot be used.
   - The execution time of forward recovery can be guaranteed, while in the case of backward recovery the execution time is longer and depends on the (number of) checkpoint intervals.

5. There is a system that consists of 3 components. Any of these components is sufficient to provide the correct service of the system. A faulty component does not provide any service.

   a. Draw the *reliability block diagram* of the system! **(1 point)**

   The reliability block diagram of the system:



   b. Give the *formula* of the reliability function $r(t)$ of the system if the reliability functions $r_i(t)$ of the components ($i=1, 2, 3$) are known. **(1 point)**

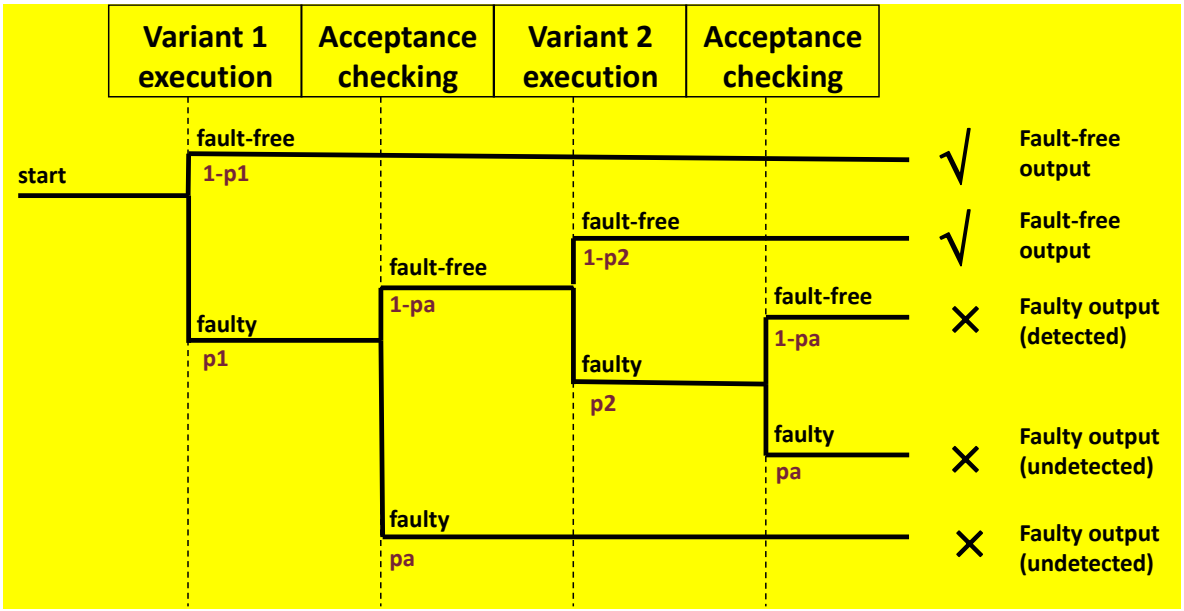
   The formula:

$$r(t) = 1 - (1- r_1(t))*(1- r_2(t))*(1- r_3(t))$$

6. There is a software system that implements the *recovery blocks* (RB) scheme including two *variants* (diverse components used to solve the same task) and an *acceptance checker* component. The two variants give faulty output with probability *p1* (primary variant) and *p2* (secondary variant). In case of a fault-free output of a variant, the acceptance checker always accepts it as fault-free. However, in case of a faulty output of a variant, the acceptance checker (due to its faulty implementation) accepts it as fault-free with probability *pa*.

   a. Construct the *event tree* belonging to this system. The initial event is the starting of the system, the events are the executions of the involved components (that may be either fault-free or faulty). **(3 points)**

   The event tree belonging to this system:



   b. Indicate in case of each scenario, whether the *output of the system* is fault-free or faulty. **(2 points)**

   The scenarios are indicated above as √ (fault-free) or × (faulty).

   c. Provide the *formula* that can be used to compute the probability of a faulty output of the system. **(2 points)**

   The probability of a faulty output of the system (3 scenarios):

   $$p1*(1-pa)*p2*(1-pa) + p1*(1-pa)*p2*pa + p1*pa$$

   In case of low fault probabilities it can be approximated as:

   $$p1*p2 + p1*p2*pa + p1*pa \approx p1*p2 + p1*pa$$