CS_M13

CRITICAL SYSTEMS

(*Attempt 2 questions out of 3*)

Question 1

(a) Explain the notion of *liveware*, and why it can be used to improve the fault tolerance of a critical system.

[5 marks]

(b) Many techniques, languages and standards for critical systems were originally developed for military purposes. Determine two reasons why military applications dominate the area of critical systems.

[4 marks]

(c) Determine *two* areas of critical systems in which *weight* is a particular problem. When weight is a problem, what is the difficulty with achieving a high degree of fault tolerance? What can be done in order to reduce this problem?

[6 marks]

(d) Explain the concept of a watchdog timer. What serious limitations exist when using a watchdog timer in the landing system of a space shuttle?

[5 marks]

(e) One can apply the fault models considered for hardware mistakes to software as well. What could a *single-stuck-at* fault mean in a software system?

[5 marks]

Question 2

(a) Explain the three notions of *verification*, *validation* and *testing*. What are the differences between these three notions?

[7 marks]

(b) Describe, using a diagram, the notion of a *triple modular redundancy system*. Furthermore, explain the notion of *N-version programming*.

[5 marks]

(c) For the new generation of space shuttle, it has been suggested to use triple modular redundancy in combination with N-version programming. This means that, instead of using identical modules, one uses modules which are different with respect to N-version programming. What complications would you expect for such an arrangement as opposed to a conventional triple modular redundancy arrangement? What would be the advantages of such an arrangement?

[6 marks]

(d) A test engineer wants to test a system for the submission of module marks (measured as percentages, eg 50%) at Swansea University. He wants to use equivalence partitioning and boundary value analysis. In his test he assumes one fixed student and module, and tests whether the entering of marks for this student and module works correctly. Determine suitable partitions and typical tests to be performed. Justify your answer.

[7 marks]

Question 3

(a) Ada was the language chosen as the basis for the SPARK Ada system. Give three reasons why it was a good choice to take Ada rather than C as a basis for the SPARK approach, even though C is much more common in programming.

[6 marks]

(b) In Java, objects are created dynamically on the heap. Why is this a problem when using Java in critical systems? What could one do in order to avoid this problem?

[4 marks]

(c) Consider the following incorrect program written in SPARK Ada which is supposed to exchange the values of `X` and `Y`:

```
procedure Exchange (X, Y: in out Float)
--# derives ??;
--# post Y = X~ and X = Y~;
is
begin
  X := Y; Y:= X;
end Exchange;
```

The data flow analysis of SPARK Ada will report one error for this program. Which one? Correct the body of the program so that it passes the data flow analysis. (You might need to introduce an auxiliary variable.)

[5 marks]

(d) Determine a suitable `derives` clause so that the program passes the information flow analysis of SPARK Ada.

[5 marks]

(e) Express the post condition by using the initial values of the variables only. Is the resulting formula provable? Explain your answer.

[5 marks]