Problem 1 – Scheduling

You are given a code segment as shown below:
1. \( x = i_1 - i_2; \)
2. \( y = i_3 + i_4; \)
3. \( o_1 = x \times y; \)
4. \( z = i_5 \times i_6 \times y \)
5. \( o_2 = 7 + i_7 + x; \)
6. \( w = i_8 \times i_9 \times i_{10}; \)
7. \( o_3 = w + x \times 5 + z; \)

a) Draw the task graph representing the dependencies in the code.
b) Show how the code can be scheduled with:
   i) ASAP (no resource constraint)
   ii) ALAP (no resource constraint)
   iii) List (Assume there is only 1 adder and 2 multipliers)

Problem 2 – Finite State Machine (FSM)

Convert the following StateChart into an equivalent FSM.
Problem 3 – SDL
Convert the following FSM into an equivalent SDL representation.

Problem 4 – Real-Time Operating Systems
Analyze one task/thread scheduling algorithm used in each of the systems listed below (if there are multiple algorithms available, pick one). Explain why the particular scheduler were chosen and what applications it works best for.
   a. Android (Lollipop+)
   b. RTLinux
   c. TinyOS
   d. Apple iOS
   e. VxWorks

Problem 5 – Real-Time I/O
Identify the challenges of implementing real-time I/O on an embedded platform. In particular, make sure to address:
   a. Battery/energy
   b. Connectivity
   c. Computing/storage constraints
   d. Implementing on top of a RTOS (hint: think about what an RTOS does when it cannot complete a task)
For each of the above, discuss solutions in research and/or in real implementations.
Problem 6 – Real-Time I/O

One of the challenges with real-time I/O is handling communication over heterogeneous communication media. TTP/C introduced a distributed approach to the hard/soft real-time communication implemented in TTP/A. However, it has not seen widespread use outside of automotive applications, where the hardware is relatively homogeneous and controllable. How might you leverage this in the larger space of real-time embedded applications, particularly those where general purpose computing is also available (i.e. the Internet of Things or the smart grid). Specifically, how can you leverage features of RTOS (which can be implemented on the general-purpose computing nodes) to provide a distributed RTIO system? Pay particular attention to hard- and soft-deadline applications that exist in such a system.