Final exam

NAME: _____________________________________________________

ID: ________________________________________________________

<table>
<thead>
<tr>
<th>Problem</th>
<th>Max. Points</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>T/F</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>Esterel</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>SDF</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>Priority inh</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>Petri net</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>RM/EDF</td>
</tr>
</tbody>
</table>

Total 140

INSTRUCTIONS:

1. Please take a moment to make sure that your test is complete and readable.
2. SHOW YOUR WORK. Partial credit is more easily awarded this way.
3. You may use any handouts available on cse237a schedule web page, everything else is off limits.
4. Please sign the statement below, before you begin.

HONOR CODE:

By signing my name below I hereby certify that I have neither given, nor received assistance in completing this examination.
1. **[20]** Write if the following statements are True (T) or False (F)

<table>
<thead>
<tr>
<th>Part</th>
<th>Statements</th>
<th>T/F</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Example: My CSE237a project is done.</td>
<td>T</td>
</tr>
<tr>
<td>1</td>
<td>Total order on events in distributed systems can be obtained with logical clocks.</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>Kahn process networks typically cause no context switching overhead.</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>DCE is a distributed compiling environment helpful for application development.</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td>AND super state in StateCharts models concurrency.</td>
<td>T</td>
</tr>
<tr>
<td>5</td>
<td>Motes are real-time sensor nodes that run RT-Linux.</td>
<td>F</td>
</tr>
<tr>
<td>6</td>
<td>UML is a serial communication protocol.</td>
<td>F</td>
</tr>
<tr>
<td>7</td>
<td>Causality analysis is for clock synchronization in reactive synchronous models.</td>
<td>F</td>
</tr>
<tr>
<td>8</td>
<td>Addition of delta delay in Verilog prevents nondeterminate behavior.</td>
<td>F</td>
</tr>
<tr>
<td>9</td>
<td>PALOS is a good example of a standard OS with real time extensions.</td>
<td>F</td>
</tr>
<tr>
<td>10</td>
<td>SDL uses message passing.</td>
<td>T</td>
</tr>
<tr>
<td>11</td>
<td>The biggest problem faced by multicore SOC designers is power management.</td>
<td>F</td>
</tr>
<tr>
<td>12</td>
<td>Scratchpad is used as a replacement for L2 cache in real time systems.</td>
<td>F</td>
</tr>
<tr>
<td>13</td>
<td>TI’s C54x has a 40 bit ALU for 40 bit wide operands.</td>
<td>F</td>
</tr>
<tr>
<td>14</td>
<td>EEPROM can be erased at the same granularity as FLASH.</td>
<td>F</td>
</tr>
<tr>
<td>15</td>
<td>SRAM can be integrated onto the same chip with the processors.</td>
<td>T</td>
</tr>
<tr>
<td>16</td>
<td>Arbiter is needed when DMA is used.</td>
<td>F</td>
</tr>
<tr>
<td>17</td>
<td>CAN is a serial communication protocol used for real-time applications.</td>
<td>T</td>
</tr>
<tr>
<td>18</td>
<td>Parallel protocols such as PCI are useful for on chip communication.</td>
<td>F</td>
</tr>
<tr>
<td>19</td>
<td>Aliasing occurs when sampling period is half the highest frequency of signal.</td>
<td>T</td>
</tr>
<tr>
<td>20</td>
<td>PID is an open-loop control technique that is more accurate than PI or PD.</td>
<td>F</td>
</tr>
</tbody>
</table>
2.  
   a)  
   What is the functionality described by this code?  Inputs are: Request, PreviousPassed, Token; Outputs are: Granted, Pass, PassToken. Show two possible execution sequences.

   ```
   loop
     present [ Token or PreviousPassed ] then
       present Request then
         emit Granted
       else
         emit Pass
       end present
     end present
each tick
```
```
  ||
  loop
    present Token then
      await tick;
      emit PassToken
    else
      await tick
    end present
  end loop
end module
```

The bus is a ring on which a bunch of identical stations are hooked. In each instant, the user of the bus can request the bus and he can obtain it or not. A priority mechanism arbitrates simultaneous requests. A token defines the current initial station. At any time, the bus is granted to the first station that asks for it, starting from the initial station in clockwise order. To obtain fairness, the token is moved to the next station in each instant, so that each station is the initial one in turn.
b) [10] Assume the code above describes a module STATION(Request, Granted, 
PreviousPassed, Pass, Token, PassToken). Explain what the following code 
segment does. Show two possible execution sequences.

```
emit Token1
||
run Station1 / 
  STATION [ signal Request1 / Request, 
    Granted1 / Granted, 
    Pass3 / PreviousPassed, 
    Pass1 / Pass, 
    Token1 / Token, 
    Token2 / PassToken ]
||
run Station1 / 
  STATION [ signal Request2 / Request, 
    Granted2 / Granted, 
    Pass1 / PreviousPassed, 
    Pass2 / Pass, 
    Token2 / Token, 
    Token3 / PassToken ]
||
run Station1 / 
  STATION [ signal Request3 / Request, 
    Granted3 / Granted, 
    Pass2 / PreviousPassed, 
    Pass3 / Pass, 
    Token3 / Token, 
    Token1 / PassToken ]
```

This code implements a token passing protocol among three entities on a same ring bus.
3. [30] Consider the Petri net consisting of places (p’s) and transitions (t’s).

a) [10] let $M_0 = [0 \ 0 \ 1]$. What happens in the subsequent operations of the Petri net?

0 0 1
T3
020
T1
110
T1
200
T2
001 – PASS schedule – can be statically scheduled
b) [10] Assume that transition priority order is $t_1$ (highest), $t_2$, $t_3$ (lowest). For the same initial state, $M_0 = [0 2 1]$, show a PASS and the maximum buffer sizes needed for it.

Possible pass is $t_1 \ t_1 \ t_2 \ t_3$

Max buffer sizes are 2,2,2 for each place
c) [10] Given the same assumptions as part b), now design the schedule capable of running on two processors (CPU1 and CPU2) of equal capabilities. Assume that transition execution times for t1-3 are T1=1, T2=2, T3=3. What is the fastest execution time of your parallel schedule assuming you run two repetitions of the schedule?

<table>
<thead>
<tr>
<th>CPU1</th>
<th>t3</th>
<th>t3</th>
<th>t3</th>
<th>t1</th>
<th>t3</th>
<th>t3</th>
<th>t3</th>
<th>t3</th>
<th>t3</th>
<th>t1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU2</td>
<td>t1</td>
<td>t1</td>
<td>t2</td>
<td>t2</td>
<td>t1</td>
<td>t2</td>
<td>t2</td>
<td>t1</td>
<td>t1</td>
<td>t2</td>
</tr>
</tbody>
</table>

The fastest time for two repetitions is 10 time units.
A robot has been designed with three different tasks \( J_A, J_B, J_C \), with increasing priority. The task \( J_A \) is a low priority thread which implements the DC-motor controller, the task \( J_B \) periodically send a "ping" through the wireless network card so that it is possible to know is the system is running. Finally the task \( J_C \), with highest priority, is responsible to check the status of the data bus between two I/O ports, as shown in Figure 6.3. The control task is at low priority since the robot is moving very slowly in a cluttered environment. Since the data bus is a shared resource there is a semaphore that regulates the access to the bus. The tasks have the following characteristics (period \( T_i \), execution time \( C_i \)):

\[
\begin{array}{|c|c|c|}
\hline
J_A & 8 & 4 \\
J_B & 5 & 2 \\
J_C & 1 & 0.1 \\
\hline
\end{array}
\]

Assuming the kernel can handle preemption. Analyze the following possible working condition:

- at time \( t = 0 \), the task \( J_A \) is running and acquires the bus in order to send a new control input to the DC-motors,
- at time \( t = 2 \) the task \( J_C \) needs to access the bus meanwhile the control task \( J_A \) is setting the new control signal,
- at the same \( t \) \( J_B \) is ready to be executed to send the "ping" signal.

a) [10] Show graphically the execution order of tasks. How long does it take for task \( J_C \) to finish execution?
b) [10] How much faster can the task \( J_C \) finish execution? Show your result.

a) The situation is shown in figure below. Let us consider the single instances.

1) The task \( J_C \) requires to be executed. Since it has high priority the CPU preempts the task \( J_A \).
2) Since \( J_C \) needs to access the bus, and this resource is occupied by the task \( J_A \) the task \( J_C \) is stopped. At the same time task \( J_B \) asks for CPU time, and since \( J_C \) is stopped and it has higher priority of \( J_A \), \( J_B \) can be executed. The execution of \( J_B \) prevents \( J_A \) to release the bus.
3) The task \( J_B \) with medium priority has finished to be executed and the CPU is given to the task \( J_A \).
4) Task \( J_A \) finishes to write on the bus, and it releases it.
5) Task \( J_C \) finally can use the bus and it is scheduled by the CPU.

What we can see from this particular situation is that a high priority task as \( J_C \) is blocked by a low priority task. This is priority inversion.

Schedule for the control task \( J_C \) and the task handling the interrupt:
b) A possible way to overcome the problem is to use the priority inheritance protocol. In this case the task \( J_A \) inherits the priority of the task which has higher priority and needs to use the blocked resource. The task \( J_A \) thus acquires high priority and is able to release the resource as soon as possible, so that the high priority task \( J_C \) can be served as soon as possible. The priorities are set to the default one, once the bus is released.
4. [20] Consider two tasks $J_1$ and $J_2$ running on a single CPU.
   a) [10] Model with a Petri net the status of the CPU. You may assume that the two tasks are not preemptive. The CPU can be in one of the three following states: 
   
   \[ \text{CPU} = \{ \text{J1 running, J2 running, idle} \} \]

   b) [10] Suppose that the two tasks $J_1$ and $J_2$ are scheduled using the rate monotonic algorithm, and that it gives high priority to $J_1$ and low priority to $J_2$. A computer engineer tries to model the CPU status when $J_1$ may preempt $J_2$. The design is shown below. Does this Petri net with its initial condition accomplish what the engineer set out to describe? If not, show situations in which it fails. If so, define the maximum buffer sizes at each place and possible transition sequences.

   **Places model the following states:**
   
   - $P_1$: $J_1$ running
   - $P_2$: $J_2$ running
   - $P_3$: Idle
   - $P_4$: $J_2$ preempted
   - $P_5$: $J_2$ not preempted

   **Problems with this Petri net:** Suppose that $J_2$ is running and then preempted by $J_1$. When $J_1$ is then terminated, the mark in $P_1$ must then go to idle state in $P_3$ and will not return to $P_2$. Furthermore, the mark in the preempted state $P_4$ will not return to $P_5$ automatically.
5. [30] A periodic control task $C$ is executed on a CPU, which executes also two other tasks ($A$ and $B$). The tasks have the following characteristics (period $T_i$, deadline $D_i$, exec time $C_i$):

<table>
<thead>
<tr>
<th>Task</th>
<th>$T_i$</th>
<th>$D_i$</th>
<th>$C_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>6</td>
<td>6</td>
<td>0.5</td>
</tr>
<tr>
<td>$B$</td>
<td>12</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>$C$</td>
<td>$x$</td>
<td>$x$</td>
<td>1</td>
</tr>
</tbody>
</table>

a) [5] Suppose 39% of the CPU utilization is reserved for other activities (other than $A$, $B$ and $C$). What is the minimum task period for the control task $C$ to guarantee that all tasks $A$, $B$ and $C$ are schedulable under rate-monotonic (RM) scheduling?

The total CPU utilization should be less than 69% to guarantee that RM scheduling is feasible. Since other tasks are using 39%, there is 30% left for tasks $A$, $B$, and $C$. Hence,

$$0.3 = U = \sum_{i=1}^{3} \frac{C_i}{T_i} = \frac{0.5}{6} + \frac{1}{12} + \frac{1}{x}$$

Task period for $C$ is: $x=15/2$

b) [5] Suppose instead that all of the CPU is available for $A$, $B$ and $C$. What is the minimum task period for the control task $C$ under earliest deadline first (EDF) scheduling?

$$1 = U = \sum_{i=1}^{3} \frac{C_i}{T_i} = \frac{0.5}{6} + \frac{1}{12} + \frac{1}{x}$$

Task period for $C$ is $x=6/5$

c) [10] Assume task $C$ has a period of 3. Show RM schedule for all three tasks. Priority of tasks: $C$; $A$; $B$. The schedule repeats starting at the $12^{th}$ period. Note that two $C$’s and $B$’s really just mean that tasks $B$ & $C$ last 1 time unit, while task $A$ lasts only 0.5 time units.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$C$</td>
<td>$C$</td>
<td>$A$</td>
<td>$B$</td>
<td>$C$</td>
<td>$C$</td>
<td>$A$</td>
<td>$C$</td>
<td>$C$</td>
<td>$C$</td>
<td>$C$</td>
<td>$C$</td>
<td>$C$</td>
</tr>
</tbody>
</table>

d) [10] Same as c) but now show the minimum energy consumption EDF schedule. Assume that when CPU runs at top speed, it consumes 2W, when running at half speed it consumes 1W, when idle it consumes also 1 W. Also assume the time units for tasks are in seconds. How much energy is being consumed in Joules for the first 12 time units?

Right now out of 12 time units, the tasks are idle 6 periods, so the total energy consumption is: $6*2 + 6*1 = 18$ J
A: 6 ½
B: 12 1
C: 3 1
(start time, deadline)
A       (0,6)     (6,12)     (12,18)
B       (0,12)    (12,24)
C       (0,3)     (3,6)      (6,9)      (9,12) (12,15)

Max slack available is 6 time units:
4*c+2*a+b=12  where c=b and a=1/2 b
4x+x+x=12  x=2
We can slow down all computation for total energy consumption of:
12s*1 W = 12 J

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>C</td>
<td>A</td>
<td>C</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>C</td>
<td>CC</td>
<td>AA</td>
<td>BB</td>
<td>BB</td>
<td>CC</td>
<td>CC</td>
<td>CC</td>
<td>CC</td>
<td>AA</td>
<td>CC</td>
<td>CC</td>
</tr>
</tbody>
</table>