Problem P1 [15 points]:
Is the following statement true or false: If \( T_i = k_i \cdot T_{i-1} \) where \( k_i = 2, 3, \ldots \), then a set of \( n \) tasks is RM-schedulable if \( C_1/T_1 + C_2/T_2 + \ldots + C_n/T_n \leq 1 \). If you believe it to be true, prove it. If you think it is false, provide a counter example. \( \text{(Hint: N&S conditions are given by the response time test.)} \)

Problem P2 [10,5 points]
Is the following task set schedulable when it is required that the computation represented by task A be given the highest priority since they have the highest criticality. If not, how would you transform this task set to make it schedulable?

<table>
<thead>
<tr>
<th>Task</th>
<th>T</th>
<th>C</th>
<th>Criticality</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>60</td>
<td>10</td>
<td>High</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>3</td>
<td>Low</td>
</tr>
<tr>
<td>C</td>
<td>8</td>
<td>2</td>
<td>Low</td>
</tr>
</tbody>
</table>

Problem P3 [25 points]
What is a transaction in Mars operating system? Describe the mechanisms and assumptions used by the Mars operating system to guarantee time of the real-time transactions. Explain how the time-stamping in Mars operating system controls the measurement error in reading the time of one component by another component.

Problem P4 [15, 15, 15]: Max Data Rate on Polled Waiting Loops
Polled waiting loops test the status of device before transferring a data byte. For either input or output, the time required to poll a device determines the minimum time per transfer and thus the maximum data rate. To speed up this rate, the polled waiting loop functions can be programmed in assembly, as shown below:

```c
#define SDP 0x2F8 /* Serial Data Port */
#define SSP 0x2FD /* Serial Status Port */
#define RX_READY (l <<0) /* bit 0: 1 = input data ready */

Serial_Input (void) {
    While ((inportb(SSP) & RX_READY) == 0) {};
    Return inportb(SDP);
}
```

_Serial_Input:
MOV DX, 02FDh
ST1: IN AL, DX
TEST AL, 00000001b
JZ ST1
MOV DX, 02F8h
IN AL, DX
MOVZX EAX, AL
RET

Since the function _Serial_Input makes no access to memory except to fetch instructions and to pop 4 bytes of return address information off the stack. And it does to IO transfers. Assume:
   a. the IO device is ready (so that no looping occurs)
   b. all opcodes occupy a single byte of memory
   c. all instructions are read off the cache
   d. data bus to memory is 4 bytes wide, we retrieve 4 bytes of data per memory read

Answer the following:
   a. How many bytes of data is needed from memory (including opcodes, operands, stack info) for each iteration of the _Serial_Input? How many (minimum) memory cycles are needed to fetch this data? Assuming a memory cycle time of 60 ns, what is the total time needed for memory transfers, exclusive of IO transfers?
   b. Estimate additional time needed for IO transfers. Assume IO device resides on an adapter card installed on a 33-MHz PCI bus (i.e., 30 ns per IO read)
   c. What is the maximum data-transfer rate supported by our function?

**Problem P5 [20 points]: Real-Time Memory Management**

Suppose three processes are running in an interrupt-only system where a single interrupt based on three prioritized interrupts is generated. Let T1, T2 and T3 be three tasks as follows:

```
procedure T1;
  begin
    application1;
    application2;
  end;
procedure T2;
  begin
    application2;
    application3;
  end;
procedure T3;
  begin
    application3;
    application4;
  end;
```
Suppose T1 is running when it is interrupted by T2 during application 2. Later T2 is then interrupted by T3 during application 3. Assuming runtime stacks are used, that is, each task has its own runtime stack. Show the contents and their pointers for the various stacks.

*Hint: Use a stack for main tasks whose contents will list tasks. Runtime stacks of the tasks will list applications.*