Exercise 6.1: Response Time Analysis

The following process set is given:

<table>
<thead>
<tr>
<th>Process</th>
<th>Exec. Seq.</th>
<th>Release Time</th>
<th>T</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>EEEQVQEE</td>
<td>6</td>
<td>40</td>
<td>10</td>
</tr>
<tr>
<td>L2</td>
<td>QEQV</td>
<td>4</td>
<td>35</td>
<td>15</td>
</tr>
<tr>
<td>L3</td>
<td>EVWE</td>
<td>4</td>
<td>30</td>
<td>20</td>
</tr>
<tr>
<td>L4</td>
<td>EWVQQE</td>
<td>0</td>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>

The priorities should be assigned according to the deadline monotonic scheme. E, Q, V, and W stand for one unit of time of execution with Q, V, and W representing mutually exclusive access to resources Q, V, and W, respectively. A sequence QQ is to be read as “the process accesses resource Q for two units of time of execution” and not as “the process accesses resource Q twice for one unit of time each”. During these two units of time, no other process can lock resource Q. For the sequence QEQV, resource Q is unlocked for use by another process after the sequence QQ finishes. The period $T$ indicates when the process happens to become runnable periodically (if it is not yet running), and the relative deadline $D$ indicates when the process has to finish.

Determine both the theoretical and actual response time values of each process and indicate whether all the deadlines have been met. Depict the system behaviour using a time line diagram. Where applicable differentiate between true and priority blocking while also showing dynamic process priorities. The following uniprocessor schemes are to be considered:

a. Fixed Priority Scheme

b. Simple Priority Inheritance Scheme

c. Original Ceiling Priority Protocol (OCPP)

d. Immediate Ceiling Priority Protocol (ICPP)
Exercise 6.2: Rendezvous

a. Ada select Statement for Entry Calls
   Consider the following Ada task, which realizes a basic rendezvous. Task T declares one entry. The body of task T (not shown here) will contain the corresponding accept statement, and rendezvous code.

   task T is
   entry Call;
   end T;

   Below are 2 client code fragments A and B, which call the entry of task T:

   Code fragment A:
   select
     T.Call;
   or
     delay 10.0;
     ... -- code that takes 2 seconds to execute
   end select;

   Code fragment B:
   select
     T.Call;
   else
     delay 10.0;
     ... -- code that takes 2 seconds to execute
   end select;

   Explain which statements are executed in each fragment under the following assumptions:
   1) T.Call is available when the select is executed.
   2) T.Call is not available when the select is executed, but does become available in the next:
      a) 14 seconds
      b) 8 seconds
      c) 2 seconds
A rendezvous with T.Call takes 5 seconds to execute.

b. **Rendezvous and Termination**

The sieve of Eratosthenes computes all prime numbers in a certain range. It works as follows: in order to determine if \( N \) is a prime number, all (previously established) prime numbers smaller than \( N \) are tested in turn, and as soon as one divides \( N \) (without any remainder), clearly it is not prime so it is discarded. If all primes smaller than \( N \) have been tried without having to discard \( N \), it obviously is a prime number.

A concurrent implementation of this algorithm based on the Rendezvous mechanism for synchronization and communication (which computes the prime numbers up to 100) is given at:

http://www.iste.uni-stuttgart.de/fileadmin/user_upload/iste/ps/Lehre/SS2015/V_RTP/a6/sieve.adb

i) Explain how the algorithm works. How many tasks are created and how do they communicate? Show snapshots of the program state to visualise the behaviour of the algorithm.

ii) The given implementation does not terminate. Why is that? Modify the code so that the program terminates after all prime numbers have been computed.

Hint: You have to add a select statement for the accept call.

Note: Please look at the code before the exercise session since there will not be enough time to explain it in detail in class.
Exercise 6.3: Multiple Choice Questions

Are the following statements true or false? Please also provide a reason.

a. A specific parameter evaluation order is prescribed by every programming language.

b. If several execution sequences of the same statement are allowed, it is possible that some of those lead to runtime errors while others do not.

c. Forward error recovery is generally much easier to implement than backward recovery.

d. For a language in which all variables are statically sized (e.g., C or Java), the size of an activation record on the stack can be statically determined.

e. When writing a program, one should define exception handlers only when necessary because they cause a significant run-time overhead during program execution.

f. Under transient overload of a system, hard deadlines are allowed to be missed occasionally under worst-case (WCET) assumptions as long as they are met under average-case (ACET) assumptions.

g. Any non-determinism of timing characteristics due to concurrency in hard real-time applications is to be avoided at all costs.

h. In Ada 2012 the functionality of protected objects can be achieved by only using conventional abstract data types, procedures with and without preconditions and functions.

i. Cumulative drift occurs because a task cannot always be scheduled to run immediately when the periodic delay expires.

j. The (relative) delay statement in Java may cause local drift, but cannot cause cumulative drift.

k. Under a Simple Priority Inheritance protocol, nested monitor calls may cause priority inversion.

l. If processes on a uni-processor system cannot be blocked for any reason (other than mutually exclusive access to a resource), OCPP for processes and resources will guarantee immediate availability of shared resources for executing processes.

m. Period displacement is the effect that a series of delays of equal duration in a task will eventually add up to much more than the sum of the delays because the task cannot always be scheduled immediately to run when the delay expires.