Further Modula-3

Synchronisation of threads in Modula-3 is achieved through the use of mutexes and condition variables. An alternative scheme would be to use Dijkstra semaphores. A semaphore has a hidden value (usually set to 1 initially) and two atomic operations:

\[ \text{wait} \] (sometimes called \text{P}) decrements the stored value. If the result is negative, the thread is suspended; otherwise it continues.

\[ \text{signal} \] (sometimes called \text{V}) increments the value. If there are any other threads suspended while waiting for the semaphore, one of them is allowed to continue.

Write an interface \texttt{Semaphore} defining an opaque object type \texttt{T} with \texttt{init}, \texttt{signal} and \texttt{wait} methods. \hfill [5 marks]

Sketch an implementation of the \texttt{Semaphore} module giving a concrete revelation of \texttt{T} and implementing appropriate default methods. \hfill [10 marks]

Show how the interface and implementation could be extended to derive a sub-type of \texttt{T} with an extra method, \texttt{try}, which works like \texttt{wait} but returns a \texttt{BOOLEAN} value instead of blocking. In the normal case, \texttt{try} should return \texttt{TRUE} but when the thread would have been suspended, the value in the semaphore is left unchanged and it should return \texttt{FALSE}. \hfill [5 marks]