Tentamen Concurrency, 20 januari 2009

Tijdsduur 3 uur. Gesloten boek tentamen.

Voorzie alle in te leveren bladen van je naam, en nummer ze. Schrijf op het eerste blad het aantal ingeleverde bladen. Werk netjes, formuleer scherp en zorgvuldig. Schrijf duidelijk leesbaar.

Als het tentamen is nagekeken, kun je het komen inzien bij Wim H. Hesselink, Bernoulliborg kamer 374.

Opgave 1 (20 %).
(a) Specificeer *wederzijdse uitsluiting* (mutual exclusion) voor een systeem van N processen.
(b) Definieer de begrippen *semafoor* en *binaire semafoor*.
(c) Definieer het begrip *mutex*. Wat is voor het gebruik het belangrijkste verschil tussen mutexen en semaforen?
(d) Implementeer de wederzijdse uitsluiting van onderdeel (a) met behulp van semaforen.
(e) Implementeer de wederzijdse uitsluiting van onderdeel (a) met behulp van mutexen.

Opgave 2 (40 %). Beschouw een begrensde stapel voor een systeem van concurrent processen met twee procedures

procedure *push*(x : *Item*) ;
procedure *pop*() : *Item* .

De aanroep *push*(x) plaatst x bovenop de stapel. De aanroep *pop*() haalt het bovenste element van de stapel en levert dit op. Als de stapel leeg is, moet de procedure *pop* wachten tot een ander proces iets gepusht heeft. De stapel kan N items bevatten. Als de stapel N items bevat, moet *push* wachten tot een ander proces iets gepopt heeft.
(a) Geef hiervan een arrayimplementatie met behulp van gedeelde variabelen, atomiciteitshaakjes en (enkelvoudige of samengestelde) **await** statements.
(b) Zet de implementatie van onderdeel (a) om in een implementatie met standaard Java-primitieven. Je mag *niet* gebruik maken van de Java-classen Semaphore en Mutex.
(c) Zet de implementatie van onderdeel (a) om in een implementatie met gesplitste binaire semaforen.

Z.O.Z.

Opgave 3 (40 %). Gegeven zijn $N > 1$ processen, alle in een oneindige lus

$$\textbf{do } \mathit{true} \;\; \rightarrow \;\; \mathit{TNS} \; ; \; \texttt{intro} \; ; \; \mathit{ACT} \; ; \; \texttt{exit} \;\; \textbf{od} \;\; ,$$

waarbij *TNS* en *ACT* gegeven commando's zijn die gegarandeerd eindigen.

We stellen nu de volgende synchronisatie-eisen. Elk process q heeft een Boole'se privévariabele $bb.q$. Er dient altijd *ten hoogste* één proces r te zijn waarvoor $bb.r$ geldt. Er dient *precies* één dergelijk proces te zijn, wanneer *ACT* wordt uitgevoerd door welk proces dan ook. Als q *TNS* uitvoert dient $bb.q = \mathit{false}$ te zijn. Dit wordt uitgedrukt in de invarianten

(J0) $\quad bb.q \;\wedge\; bb.r \;\Rightarrow\; q = r \;,$
(J1) $\quad q \textbf{ at } \mathit{ACT} \;\Rightarrow\; (\exists\, r : bb.r) \;,$
(J2) $\quad q \textbf{ at } \mathit{TNS} \;\Rightarrow\; \neg\, bb.q \;.$

Gegeven is een geheel getal $K > 1$. Een proces r waarvoor $bb.r$ geldt, dient $bb.r$ waar te houden tot tenminste K processen *ACT* uitgevoerd hebben en moet daarna de gelegenheid krijgen *TNS* weer uit te voeren.

Om dit voor elkaar te krijgen declareren we de gedeelde variabelen:

$$\begin{array}{l}\textbf{var } \texttt{cc} : \mathit{Bool} := \mathit{true} \\ \textbf{var } \texttt{cnt} : \mathit{Int} := 0 \\ \textbf{var } \texttt{n} : \mathit{Int} := 0\end{array}$$

De processen worden geïmplementeerd volgens

```
          process Member(self := 0 to N − 1)
              var bb : Bool := false
              do true →
10:               TNS
11:               ⟨ await cc then ... ⟩
12:               ⟨ bb := (n = 0) ; n ++ ⟩
13:               ACT
14:               ...
15:               if bb →
16:                   await n ≥ K
17:                   ...
18:                   await ?
19:                   ⟨ bb := false ; n := 0 ⟩
20:                   ...
                  fi
              od end Member .
```

We eisen hiervoor de invariant

(J3) $\quad \texttt{cnt} = \#\{q \mid q \textbf{ in } \{12, 13, 14\} \} \;.$

We stellen de voortgangseis:

(vtg) $\quad q \textbf{ in } \{12, \ldots, 20\} \quad o\!\!\rightarrow \quad pc.q = 10 \;.$

(a) Vul commando's in voor de puntjes in 11, 14, 17 en 20, en een conditie voor het vraagteken in 18, zodanig dat aan deze eisen voldaan wordt en dat er verder niet onnodig gewacht wordt.
(b) Toon aan dat je oplossing voldoet aan de invarianten (J0), (J1), (J2) en (J3). Je mag hiertoe andere invarianten invoeren en bewijzen.
(c) Laat zien dat je oplossing voldoet aan de voortgangseis (vtg).