Tentamen Concurrency, 24 april 2008

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 (25 %).
(a: 5 %) Specificeer een *barrier* voor N processen (geef een sluitende definitie).
(b: 15 %) Geef een implementatie van een barrier voor N processen met behulp van pthread primitieven (mutexen en conditievariabelen). Deze implementatie moet bestand zijn tegen "spurious wakeups".
(c: 5 %) Transformeer deze implementatie door mutex-abstractie tot een samentelling van twee atomaire commando's.

Opgave 2 (25 %). Beschouw een systeem met twee gedeelde variabelen `x` en `as`, en N processen volgens

```
var x : int := 0 ;
var as[N] : int ;

process Member (self := 0 to N − 1)
    var t : int ;
    do true  →
        TNS
        ⟨ as[self] := t ⟩
        ⟨ await x ≥ t then x := x − t ⟩
    od
end Member .
```

Het programmafragment *TNS* is gegeven. Het wijzigt de gedeelde variabelen `x` en `as` niet. Het eindigt gegarandeerd, en het kan de waarden van de privévariabele t wijzigen (bv. negatief maken).

Implementeer de atomiciteitshaakjes en de samengestelde await statement van dit systeem met behulp van gesplitste binaire semaforen. Processen mogen elkaars privévariabelen niet inspecteren. Er mag geen busy waiting optreden. Elke PV-sectie dient tenminste één zinvolle toekenning aan een gedeelde variabele te bevatten.

Z.O.Z.

Opgave 3 (30 %). Gegeven zijn gehele getallen $1 \leq K \leq N$. Ontwerp een systeem van N processen van de vorm:

```
        process Member(self := 0 to N − 1)
          do true  →
10:           NCS
11:           ingang
12:           voorhal
13:           CS
14:           achterhal
15:           uitgang
          od end Member .
```

De noncritical section *NCS* en de critical section *CS* zijn gegeven.
(a) Implementeer de commando's `ingang`, `voorhal`, `achterhal`, en `uitgang` met gedeelde variabelen, atomiciteitshaakjes en samengestelde **await** statements zo, dat voldaan wordt aan de invariant:

(J0) $\quad q$ **at** 13 $\Rightarrow$ $\#\{r \mid r$ **in** $\{12, 13, 14, 15\}\} = K$.

Als er K processen bij 12, 13, 14, 15 zijn, moeten deze processen terug naar *NCS* voordat nieuwe processen bij 12 worden toegelaten. Processen moeten verder niet onnodig tegengehouden worden.
Aanwijzing: gebruik twee gedeelde variabelen die aangeven hoeveel processen bij 12, 13, 14, 15 zijn en hoeveel processen bij 15. Regel de toelating tot *CS* met een boole'se variabele.
(b) Bewijs dat je oplossing aan (J0) voldoet.

Opgave 4 (20 %). Gegeven een systeem van N processen die uitsluitend door middel van boodschappen communiceren met elkaar en met de *omgeving*. De processen hebben privévariabelen x die soms kunnen veranderen. De *omgeving* vraagt met de boodschap `vraag` soms één van de processen om de som van alle waarden $x.q$ van alle processen. Het proces dat de `vraag` krijgt dient te antwoorden met de boodschap `antwoord`, overeenkomstig het volgende fragment.

```
op vraag[0 : N − 1]() ;
op antwoord(zender, som : int) .

process Deelnemer(zelf := 0 to N − 1)
  var x : int
  do true  →
    in vraag[zelf]()  →
        "bepaal som"
        send antwoord(zelf, som)
    [] andere boodschappen
    ni
  od end Deelnemer .
```

De processen staan in een ring en mogen onderling alleen communiceren door het zenden van boodschappen van p naar $(p+1)$ **mod** N (de "rechter buurman"). Het proces dat de `vraag` krijgt, dient de som $\sum_q x.q$ te bepalen door een boodschap de ring rond te sturen. Implementeer dit. Je hoeft je geen zorgen te maken over tussentijdse wijzigingen van de waarden $x.q$.

Je mag de *omgeving* niet implementeren of gebruiken.