Tentamen Concurrency, 21 november 2000

Tijdsduur 3 uur. Gesloten boek.

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.

Opgave 1. Gegeven is een aantal client processen volgens de declaratie

```
const K: int   # K > 0

process Client (self := 1 to K)
   do true ->
      NCS
      AS
   od end Client
```

`NCS` staat als gebruikelijk voor een niet-critische sectie die niet hoeft te eindigen. `AS` staat voor "await service". Er is één proces Server dat als enige de service kan verlenen en wel door het aanroepen van

```
procedure Serve (q: int)
```

De aanroep `Serve(q)` mag alleen gedaan worden als proces `q` **at** `AS` staat. De client `q` mag `AS` pas verlaten als `Serve(q)` is uitgevoerd.

Implementeer `AS` en het proces Server met shared variables. Je mag hierbij gebruik maken van de constructie $\langle$ **await** B $\rangle$ voor boole'se expressies B en van atomaire tellers. Er dient de volgende voortgangrelatie te gelden

$$q \textbf{ at } \texttt{AS} \quad o\!\!\rightarrow \quad q \textbf{ at } \texttt{NCS} \ .$$

Als alle clients **at** `NCS` zijn en blijven, dient de Server binnen een begrensd aantal ronden voor een **await** statement te gaan wachten.

Geef een zo volledig mogelijk stel invarianten om de correctheid van je oplossing te ondersteunen.

Opgave 2. Gegeven zijn N processen, genummerd van 1 tot N, die met elkaar communiceren dmv asynchrone boodschappen. Elk proces q kan alleen boodschappen sturen naar zijn buren, dwz naar elementen van de lijst `buren`$[q]$, die geen meervoudige elementen heeft. We gebruiken hiertoe de globale declaraties

```
type Intlist = ptr rec (value: int; nx: Intlist)
var buren[N]: Intlist
var cntburen[N]: int
```

Er geldt voor alle q en r

$$\begin{array}{l} q \textbf{ not in } \texttt{buren}[q] \ , \\ r \textbf{ in } \texttt{buren}[q] \quad \equiv \quad q \textbf{ in } \texttt{buren}[r] \ , \\ \texttt{cntburen}[q] \quad = \quad \textit{lengte}\ (\texttt{buren}[q]) \ . \end{array}$$

Elk proces is met elk ander proces via nul of meer tussenliggende processen verbonden.

Gevraagd wordt het volgende te implementeren. Proces *start* (= 1, bv) begint met van standaardinvoer een integer te lezen. Het zendt dit getal naar al zijn buren. Elk proces dat een getal ontvangt zendt het door naar zijn buren en druk het op den duur af. Zorg dat alle processen het getal precies één keer afdrukken en dat als het getal N keer is afgedrukt, alle processen geëindigd zijn en er geen boodschappen meer in omloop zijn.

Z.O.Z.

Opgave 3. Het ontvangen van boodschappen in een shared memory systeem. Stel, dat een process (*Consumer*) boodschappen moet kunnen ontvangen van een aantal andere processen (Producers). We voorzien Consumer daartoe van een circulaire buffer `buf` met K posities. We gebruiken de operator $\oplus$ voor optelling modulo K.

We gebruiken gedeelde variabelen `nr` voor het aantal bezette posities in de buffer, `free` voor de index waar de eerstvolgende boodschap geplaatst zal worden, `read` voor de index waar de eerstvolgende boodschap gelezen zal worden, en een boole's array `wr` om aan te geven dat een boodschap geplaatst is.

Alle processen hebben een privévariabele *item* van het type boodschap en x of y voor een positie.

many Producers :
do *true* $\rightarrow$
produce item ;
$\langle$ **await** `nr` $\neq K$ **then**
$x :=$ `free` ; `free` $:=$ `free` $\oplus 1$;
`nr` $:=$ `nr` $+ 1$ $\rangle$;
`buf`$[x] :=$ *item* ;
od .

a single Consumer :
do *true* $\rightarrow$
$\langle$ **await** `wr`[`read`] **then**
$y :=$ `read` ; `read` $:=$ `read` $\oplus 1$;
`wr`$[y] :=$ *false* $\rangle$;
item $:=$ `buf`$[y]$;
$\langle$ `nr` $:=$ `nr` $- 1 \rangle$;
consume item
od .

Opdracht (a) In het bovenstaande programma is één toekenning weggelaten. Voeg deze op de juiste plaats toe.

Opdracht (b) Bepaal geschikte initiële waarden voor de gedeelde variabelen.

Men zou nu de correctheid kunnen bewijzen, maar dat wordt niet gevraagd.

Opdracht (c) Geef een correcte implementatie van het systeem volgens onderdeel (a), met behulp van gesplitste binaire semaforen. Zorg dat de body van de **await** opdracht van de consumer gelijktijdig uitgevoerd kan worden met de body van de **await** opdracht van elk van de producers. Dit kan als `nr` en `free` bewaakt worden door de ene gesplitste binaire semafoor en `read` en `wr` door de andere.