OHJ-4010 Rinnakkaisuus Tentti 1.2. 2010

Tentissä ei saa käyttää ylimääräistä kirjallista materiaalia, laskimia, tietokoneita tai muita lunttausvälineitä.

Tentin tarkistaa Johannes Koskinen.

1. Tehtävä

Ohessa on kuusi vaihtoehtoa lukija-kirjoittajaongelmassa esiintyvän kirjoittajan nälkiintymisen ratkaisemiseksi. Käytössä ovat semaforit *write_mutex*, *mutex* ja *extra*, jotka kaikki on alustettu arvoon 1. Lukijoilla on lisäksi yhteinen kokonaislukumuuttuja *read_count*, joka on alustettu nollaksi. Vaihtoehtoiset algoritmit lukijan varausalgoritmeiksi ovat A-C ja vapautusalgoritmeiksi D-F.

Kirjoittajan algoritmi

```
-- varaus
extra.p();
write_mutex.p();
-- tee tässä välissä kirjoitus
-- vapautus
extra.v();
write_mutex.v();
```

A

```
extra.p();
mutex.p();
read_count := read_count+1;
if read_count = 1 then
  write_mutex.p();
 extra.v();
end if;
mutex.v();
```

B

```
extra.p();
extra.v();
mutex.p();
if read_count = 0 then
 read_count := read_count+1;
 write_mutex.p();
end if;
mutex.v();
```

C

```
extra.p();
mutex.p();
extra.v();
read_count := read_count+1;
if read_count = 1 then
 write_mutex.p();
end if;
mutex.v();
```

D

```
mutex.p();
read_count := read_count-1;
if read_count = 0 then
 write_mutex.v();
else
 extra.v();
end if;
mutex.v();
```

E

```
mutex.p();
read_count := read_count-1;
if read_count = 0 then
 write_mutex.v();
end if;
mutex.v();
```

F

```
mutex.p();
if read_count = 1 then
 write_mutex.v();
end if;
read_count := read_count-1;
mutex.v();
```

Muodostetaan lukijan lukitusalgoritmeista (A, B, C) ja vapautusalgoritmeista (D, E, F) algoritmiparit AD, AE, AF, BD, BE, BF, CD, CE, CF. Mitkä näistä yhdistelmistä toimii oikein, mitkä väärin? Semaforien odotetaan toimivan siten, että jonoa puretaan saapumisjärjestyksessä. Mikäli yhdistelmä toimii oikein ja tehokkaasti, siitä riittää perusteluksi OK. Mikäli algoritmissa on tehokkuusongelma tai virhe, se pitää perustella lyhyesti malliin "ratkaisu lukkiutuu joskus".

2. Tehtävä

Kuvaa aterioivien filosofien ongelma. Selitä, kuinka sen avulla voidaan esittää lukkiutuminen ja nälkiintyminen.

3. Tehtävä

Adan tehtävätyypillä (task type) voidaan toteuttaa semafori. Ohessa on tällaisen tehtävätyypin määrittelyosa.

```
task type Semaphore is
   entry Start (alkuarvo : integer);
   entry P;
   entry V;
end Semaphore;
```

Tarkoitettu käyttö on siis sellainen, että aluksi kutsutaan semaforin porttia Start, jolla annetaan semaforille alkuarvo. Tämän jälkeen sovellukset kutsuvat semaforin portteja P ja V normaaliin tapaan. V-operaatio voi vapauttaa minkä tahansa odottajista, eli jonotusta ei tarvitse tehdä minkään tietyn algoritmin mukaan.

Tehtäväsi on kirjoittaa Semaphore-tehtävätyypin runko-osa.

4. Tehtävä

Ohessa on yksi ratkaisu tuottaja-kuluttaja-ongelmaan. Tuottajaprosessi kutsuu aliohjelmaa Produced, kun se on tuottanut yhden alkion. Vastaavasti kuluttajaprosessi kutsuu aliohjelmaa Consumed, kun se haluaa uuden alkion kulutettavaksi.

```
type T is private; -- Sisältö ei siis kuulu meille
Buffer_size : constant := 100;
buffer : array 1..Buffer_size of T;
in_index : integer := 1;
out_index : integer := 1;

-- Tätä kutsutaan ennen kuin aliohjelmia
-- Produced ja Consumed aletaan käyttää
procedure Initially is
begin
   null;  -- Tässä versiossa ei tehdä mitään.
end;

procedure Produced (item : T) is
begin
   buffer (in_index) := item;
   in_index := (in_index mod Buffer_size) + 1;
end;

procedure Consumed (item : out T) is
begin
   item = buffer (out_index);
   out_index = (out_index mod Buffer_size) + 1;
end;
```

Oleta, että käytettävissäsi on edellisessä tehtävässä esitelty semafori. Saat siis uuden semaforin käyttöösi kirjoittamalla

```
uusi_semafori : semaphore;
```

Lisää välttämättömät (turhat katsotaan virheeksi) semaforit ja semaforikutsut, jos

a) Puskuri ei voi koskaan täyttyä; tuottajia ja kuluttajia on vain yksi.
b) Puskuri ei voi koskaan täyttyä: tuottajia ja kuluttajia on monta.
c) Puskuri voi täyttyä; tuottajia on kaksi ja kuluttajia on yksi.