OHJ-4010 Rinnakkaisuus Tentti 23.1.2006

Tentissä ei saa olla mukana mitään materiaalia.

1. Seuraavassa luettelossa on väitteitä. Vastausvaihtoehdot ovat "aina tosi" (AT), "yleensä tosi" (YT), "yleensä väärin" (YV) ja "aina väärin" (AV). Jos väite on mielestäsi yleensä (useinmiten) tosi, vaikka siihen on jokin poikkeus olemassa, vastaa YT. Jos taas poikkeuksia ei ole, vastaa AT. Kielteiset vastaukset vastaavasti. Arvostelu: oikeasta vaihtoehdosta +2, oikean viereisestä vaihtoehdosta +1, tätä kauempana olevasta vaihtoehdosta -1 ja tätäkin enemmän pielessä olevasta -2. Tyhjät vastaukset 0. Yhteenlasketut pisteet skaalataan välille 0..6 vastaten yhtä tavallista tehtävää.

   Vastaa kysymyspaperiin!

	AT	YT	YV	AV
1. Jos skeduleri ei ole irrottava, sovellusohjelmien poissulkemisrutiineita ei tarvita.				
2. Samaa kriittistä aluetta käyttävät prosessit ovat loogisesti sidoksissa toisiinsa (ns. multitasking-tekniikka).				
3. Turha poissulkeminen on poistettava koodista.				
4. Nälkiintymistä ei esiinny ilman poissulkemista.				
5. Kriittisiä aluetta voidaan suorittaa yhtäaikaisesti, jos ne eivät päivitä samoja muuttujia.				
6. Prosessin jakaminen säikeisiin johtaa uusien kriittisten alueiden syntyyn.				
7. Poissulkemisen toteutuksen alin taso toteutetaan laitteistolla.				
8. Lukkiutumista esiintyy ilman poissulkemista.				
9. Samalla kriittisellä aluella voi olla kaksi ohjelmaa yhtä aikaa, jos ne eivät muuta muuttujien sisältöä.				
10. Monitorissa voidaan odottaa ehdon toteutumista aktiivisella silmukalla (busy wait).				
11. Irrottamaton skeduleri on yksi lukkiutumisen välttämättömistä ehdoista.				
12. Jos oletetaan, että lukkiintuminen on estetty, aterioivia filosofeja tulee olla vähintään kolme, jotta nälkiintyminen saadaan aikaan.				

2. Ohessa on yksi lukitusrutiiniehdokas.

   ```
   lock_variable : integer := 0;

   procedure Lock (my_number : integer) is

   begin
     loop
       while lock_variable /= 0 loop
         null; -- tyhjä silmukka
       end loop;
       lock_variable := my_number;
       for j in 0 .. 10000 loop
         null; -- toinen tyhjä silmukka
       end loop;
       exit when lock_variable = my_number; -- ehdollinen poistuminen
     end loop;
   end Lock;
   ```

   Lukitusrutiini | **Vapautusrutiini**

   ```
   procedure Unlock is
   begin
     lock_variable := 0;
   end Unlock;
   ```

   Vastaa *lyhyesti* seuraaviin kysymyksiin:

   a. Voidaanko algoritmia käyttää sellaisenaan monisuoritinkoneen suorittimien välisenä poissulkemisrutiinina? Jos voi, onko käytöllä joitain rajoituksia? Jos ei voi, tee algoritmista käyttökelpoinen, jos se on mahdollista *pienellä* korjauksella.
   b. Voidaanko algoritmia käyttää sellaisenaan yksisuoritinkoneessa prosessien tai säikeiden välisenä poissulkemisrutiinina? Jos voi, onko käytöllä joitain rajoituksia? Jos ei voi, tee algoritmista käyttökelpoinen, jos se on mahdollista *pienellä* korjauksella.

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

   ```
   task type Semaphore is

     entry Start (initial_value : integer);
   ```