CHALMERS LINDHOLMEN

Institutionen för Data- och elektroteknik

TENTAMEN

KURSNAMN	**Parallellprogrammering**
PROGRAM: namn åk / läsperiod	**Dataingenjörsprogrammet** **DAI3 / Omtentamen april 2001**
KURSBETECKNING	**LEU 290 0199 (4p)**
EXAMINATOR	**Alois Goller**
TID FÖR TENTAMEN	**Tisdagen den 17/4 2001 kl 13.30 – 17.30**
HJÄLPMEDEL	**Inga**
ANSV LÄRARE: namn telnr besöker tentamen kl	**Anders Franzen** **772 5715** **Ca 15.00**
DATUM FÖR ANSLAG av resultat samt av tid och plats för granskning	**Fredagen den 4/5** **Granskning hos Alois Goller**
ÖVRIG INFORM. (ex.vis antal frågor, uppgifter, poäng o dyl)	

NAMN (tentand): _____

Uppgift 1: Ge en förklaring till "Manager-Worker"-modellen. Beskriv fördelar och nackdelar. (4p)

Uppgift 2: Skriv ett Manager-Worker-program i pseudokod som beräknar

$$\sum_{i=0}^{N} \frac{1}{i^2}$$

(4p)

Uppgift 3: Visa med ett kodexempel i Ada hur enkelt det är att åstadkomma deadlock. (4p)

Uppgift 4: Betrakta de fyra **select**-satserna:

(a)

```
select
  T.Call;
  Flag := A;
or
  delay 10.0;
  Flag := B;
  delay 2.0;
end select;
```

(b)

```
select
  T.Call;
  Flag := A;
else
  delay 10.0;
  Flag := B;
  delay 2.0;
end select;
```

(c)

```
select
  T.Call;
  Flag := A;
then abort
  delay 10.0;
  Flag := B;
  delay 2.0;
end select;
```

(d)

```
select
  delay 10.0;
  Flag := A;
then abort
  T.Call;
  Flag := B;
  delay 2.0;
end select;
```

Antag att exekvering av rendezvous med `T.Call` tar 5 sekunder. Visa hur `Flag` kan få värdena `A` respektive `B` då **select**-satserna avslutats. Om något fall inte är möjligt, förklara varför. (16p)

Uppgift 5: Följande kodfragment illustrerar två olika sätt att i Ada implementera heltalssemaforer:

```
package Semaphore_Package_1 is
   protected type Semaphore(Initial: Natural := 1) is
      entry Wait;
```

```
      procedure Signal;
   private
      Value : Natural := Initial;
   end Semaphore;
end Semaphore_Package_1;
```

```
package Semaphore_Package_2 is
   task type Semaphore(Initial: Natural) is
      entry Wait;
      entry Signal;
   end Semaphore;
end Semaphore_Package_2;
```

(a) Visa hur paketet Semaphore_Package_1 kan implementeras. (8p)

(b) Visa hur paketet Semaphore_Package_2 kan implementeras. (8p)

(c) Diskutera för- och nackdelar med de olika lösningarna. (4p)

Uppgift 6: I problemet "Dinerande filosofer" finns det många möjliga lösningar för att kontrollera åtkomsten till ätpinnarna. En lösning ser ut så här:

```
type Phil_ID is new Integer range 1..5;

protected Chopstick_Control is
   entry Take(Left, Right: Phil_ID);
   procedure Drop(Left, Right: Phil_ID);
private
   Free : array (Phil_ID) of Boolean := (others => True);
   entry Retry(Left, Right: Phil_ID);
   Waiting : Natural := 0;
end Chopstick_Control;

protected body Chopstick_Control is
   entry Take(Left, Right: Phil_ID) when True is
   begin
      if Free(Left) and Free(Right) then
         Free(Left)  := False;
         Free(Right) := False;
      else
         requeue Retry;
      end if;
   end Take;
```

```
   procedure Drop(Left, Right: Phil_ID) is
   begin
      Free(Left)  := True;
      Free(Right) := True;
      Waiting := Retry'Count;
   end Drop;

   entry Retry(Left, Right: Phil_ID) when Waiting>0 is
   begin
      Waiting := Waiting-1;
      if Free(Left) and Free(Right) then
         Free(Left)  := False;
         Free(Right) := False;
      else
         requeue Retry;
      end if;
   end Retry;
end Chopstick_Control;
```

(a) Ett problem med ovanstående lösning är dock att filosofer kan utsättas för svält. Visa hur. (4p)

(b) Förklara hur en lösning på problemet med svält kan se ut, utan att förändra det publika gränssnittet. (8p)