CHALMERS LINDHOLMEN

Institutionen för data- och elektroteknik

TENTAMEN

KURSNAMN	**Parallellprogrammering**
PROGRAM: namn **åk / läsperiod**	**Dataingenjörslinjen** **Inbyggda system Åk 3 / Lp I**
KURSBETECKNING	**LEU290 0199**
EXAMINATOR	**Roger Johansson**
TID FÖR TENTAMEN	**Fredag 14 januari 8:30-12:30**
HJÄLPMEDEL	**Inga**
ANSV LÄRARE: namn **Telnr** **besöker tentamen kl**	**Anders Franzén** **031 - 772 5715** **c:a 9:45 och 11:00**
DATUM FÖR ANSLAG av resultat samt av tid och plats för granskning	**Resultat anslås senast fredag 28 januari 2000** **Granskning sker hos Anders Franzén**
ÖVRIG INFORM.	▪ **Motivera val du gjort väl. Vissa uppgifter kanske inte tillhandahåller alla uppgifter för att lösa problemet. I så fall får du själv göra rimliga antaganden om dessa detaljer. Tala om vilka antaganden du gör. Om dina lösningar endast fungerar under vissa förutsättningar måste du klart tala om vilka dessa förutsättningar är.** ▪ **Poängavdrag kan ges för onödigt långa, komplicerade, eller ostrukturerade lösningar.** **Totalt antal poäng: 60**

NAMN (tentand): _____

Uppgift 1: Ge en kort förklaring av var och en av följande termer:

(a) deadlock
(b) livelock
(c) liveness
(d) safety
(e) synkronisering

(10p)

Uppgift 2: Betrakta de fyra **select**-satserna:

(a)

```
select
  accept A;
or
  delay 10.0;
end select;
```

(b)

```
select
  accept A;
else
  delay 10.0;
end select;
```

(c)

```
select
  accept A;
or
  delay 5.0;
  delay 5.0;
end select;
```

(d)

```
select
  accept A;
  delay 5.0;
or
  delay 5.0;
end select;
```

Beskriv skillnaderna mellan dessa, i de fall det finns några. (4p)

Uppgift 3: Visa hur man med hjälp av *task* och *rendezvous* kan åstadkomma deadlock mellan två task. (5p)

Uppgift 4: Betrakta nedanstående skyddade objekt:

```
protected Barrier is
   entry Wait;
private
   entry Waiting;
   Needed : Natural := 3;
   Release : Boolean := False;
end Barrier;

protected body Barrier is
   entry Wait when True is
   begin
```

```
      Needed := Needed-1;
      if Needed = 0 then
         Release := True;
      end if;
      requeue Waiting;
   end Wait;

   entry Waiting when Release is
   begin
      Needed := Needed+1;
      if Needed = 3 then
         Release := False;
      end if;
   end Waiting;
end Barrier;
```

(a) Beskriv i detalj vad som händer när följande sekvens av händelser inträffar.

Process A: `Barrier.Wait;`
Process B: `Barrier.Wait;`
Process C: `Barrier.Wait;`
Process D: `Barrier.Wait;`

De fyra händelserna kommer med så långt mellanrum att allting som skulle kunna inträffa som följd av en av händelserna inträffar innan nästa anrop av `Barrier.Wait`. Antag att det skyddade objektet inte har använts tidigare i programmet. (6p)

(b) Skriv om det skyddade objektet utan **`requeue`**. Det nya skyddade objektet skall ha samma funktionalitet som `Barrier`. (8p)

Uppgift 5: Vanligen implementeras semaforer i Ada med skyddade objekt. Ursprungligen fanns dock inte skyddade objekt i Ada, utan man var tvungen att använda task istället.

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

(a) Implementera heltalssemaforer utgående från ovanstående specifikation. Bortse från eventuella problem med terminering. (10p)

(b) Varför får man problem med terminering? Förklara! (2p)

(c) Utvidga programmet med en lösning på problemet med terminering. (3p)

Uppgift 6: En ny smal enfilig tunnel norr om staden tillåter bara trafik i en riktning i taget. För att styra trafiken har ett trafikljus installerats. Men eftersom rödljuskörningarna är så vanliga har kommunstyrelsen bestämt att alla bilar skall utrustas med en autopilot. Då en bil närmar sig tunneln skall det inte gå att köra in i den innan tunnelns styrsystem har gett klartecken.

(a) Implementera ett skyddat objekt som kontrollerar trafiken genom tunneln. Varje bil som kommer norrifrån anropar `Enter_North`, och då anropet tas emot kör bilen in i tunneln. När bilen lämnar tunneln anropas `Leave_South`. På samma sätt anropas `Enter_South` respektive `Leave_North` av bilar söderifrån. Flera bilar med samma körriktning skall få befinna sig i tunneln samtidigt. En bil som kommer norrifrån motsvaras av tasket

```
task body Car is
begin
   Tunnel_Control.Enter_North;
   Tunnel_Control.Leave_South;
end Car;
```

Utgå från följande specifikation:

```
protected Tunnel_Control is
   entry Enter_North;
   entry Enter_South;
   entry Leave_North;
   entry Leave_South;
end Tunnel_Control;
```

(8p)

(b) Stadens styrande är dock missnöjda med lösningen. De vill att norrgående bilar skall få företräde för södergående, i ett försök att minska biltrafiken i centrum. Förändra det skyddade objektet för att möta deras krav.

(4p)