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	**Torsdag 21 oktober 8:30-12:30**
HJÄLPMEDEL	**Inga**
ANSV LÄRARE: namn **Telnr** **besöker tentamen kl**	**Anders Franzén** **031 - 772 5715** **9:45 och 11:00**
DATUM FÖR ANSLAG av resultat samt av tid och plats för granskning	**Resultat anslås senast torsdag 4 november 1999** **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: Processer använder *meddelandeskickning* för kommunikation med andra processer. Olika programspråk tillhandahåller olika former av meddelandeskickning.

(a) Visa hur man kan implementera asynkron kommunikation med indirekt namngivning i Ada. (3p)

(b) Visa hur man i ett språk med asynkron kommunikation och assymetrisk direkt namngivning kan implementera remote invocation. (2p)

Uppgift 2: Vad menar man när man säger att en sekvens av satser är en kritisk region? Visa med ett exempel faran i att låta två processer samtidigt exekvera i en kritisk region. (5p)

Uppgift 3: Parallella program är mer komplexa än sekventiella program. Detta återspeglas i de krav parallella program skall uppfylla för att anses vara korrekta.

(a) Vilket krav tillkommer, förutom de som kan appliceras på sekventiella program? (2p)

(b) Ge ett exempel på varför definitionen av korrekthet för sekventiella program är otillräcklig. (3p)

Uppgift 4: Vad menas med liveness och safety? Exemplifiera! (4p)

Uppgift 5: Visa hur rendezvous mellan task kan användas i följande sammanhang:

(a) För att synkronisera två processer. (2p)

(b) För att implementera ömsesidig uteslutning. (3p)

Uppgift 6: Betrakta följande programsnuttar

```
select
  Ett_Task.En_Ingång;
or
  delay 10.0;
end select;
```

```
select
  Ett_Task.En_Ingång;
else
  delay 10.0;
end select;
```

```
select
  Ett_Task.En_Ingång;
or
  delay 5.0;
  delay 5.0;
end select;
```

```
select
  Ett_Task.En_Ingång;
then abort
  delay 10.0;
end select;
```

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

Uppgift 7: I problemet dinerande filosofer måste filosoferna plocka upp ätpinnarna på ett sådant sätt att filosoferna inte råkar ut för deadlock eller livelock. Ett sätt att åstadkomma detta är att låta filosoferna ta upp båda pinnarna i en atomär operation. Skriv ett skyddat objekt som implementerar detta, utgående från följande kodskelett:

```
type Filosof_ID is new Integer range 1..5;
protected Ätpinnar is
  entry Ta_Upp (Vänster, Höger : Filosof_ID);
  procedure Lägg_Ned (Vänster, Höger : Filosof_ID);
private
  - plats för egna deklarationer
end Åtpinnar;

task type Filosof(Vänster, Höger : Filosof_ID);
task body Filosof is
begin
  loop
    Tänk;
    Ätpinnar.Ta_Upp(Vänster, Höger);
    Ät;
    Ätpinnar.Lägg_Ned(Vänster, Höger);
  end loop;
end Filosof;

F1 : Filosof(1, 2);
F2 : Filosof(2, 3);
F3 : Filosof(3, 4);
F4 : Filosof(4, 5);
F5 : Filosof(5, 1);
```

(10p)

Uppgift 8: Betrakta följande skyddade objekt:

```
protected type Händelse is
  entry Vänta;
  entry Släpp;
private
  entry Återställ;
  Inträffat : Boolean := False;
end Event;

protected body Event is
```

```
  entry Vänta when Inträffat is
  begin
    null;
  end Vänta;

  entry Släpp when True is
  begin
    if Vänta'Count > 0 then
      Inträffat := True;
      requeue Återställ;
    end if;
  end Släpp;

  entry Återställ when Vänta'Count = 0 is
  begin
    Inträffat := False;
  end Återställ;
end Händelse;
```

(a) Beskriv de möjliga följder av händelser som kan inträffa (dvs förändringar av lokala variabler och ingångsköerna) när följande sekvens av anrop tas emot:

```
Process A:  Händelse.Vänta;
Process B:  Händelse.Vänta;
Process C:  Händelse.Släpp;
Process D:  Händelse.Vänta;
```

(6p)

(b) Skriv om det skyddade objektet så att det har samma funktion, men utan att använda **requeue** (eller någon annan entitet). (8p)

Uppgift 9: Monitorer använder *villkorsvariabler* för att sköta synkronisering inuti monitorerna.

(a) Implementera villkorsvariabler, utgående från följande kodskelett:

```
protected type Condition is
  entry Wait;
  procedure Send;
private
  -- Plats för egna deklarationer
end Condition;
```

(5p)

(b) Går det på detta sätt att implementera monitorer med villkorsvariabler i Ada? Motivera ditt svar. (3p)