UNIVERSITY
OF SKÖVDE

School of

WRITTEN EXAMINATION

Course: Concurrent Programming

Examination

Course code: IT404G	Credits for written examination: 4.5
Date: 2024-02-23	Examination time: 5 hours

Examination responsible: Richard Senington

Teachers concerned: Birgitta Lindström

Aid at the exam/appendices

Other

Instructions

- ☐ Take a new sheet of paper for each teacher.
- ☐ Take a new sheet of paper when starting a new question.
- ☒ Write only on one side of the paper.
- ☒ Write your name and personal ID No. on all pages you hand in.
- ☒ Use page numbering.
- ☒ Don´t use a red pen.
- ☒ Mark answered questions with a cross on the cover sheet.

Grade points

Examination results should be made public within 18 working days

Good luck!

Total number of pages

Grading

There are five main questions in the written exam corresponding to the course objectives. Each main question consists of a set of three sub-questions, which are graded pass or fail. To pass the exam, you need to ***pass at least one sub-question for each of the main questions****. The more sub-questions you pass, the higher your grade will be. The detailed grading scheme is published in Canvas. For your convenience, each section lists the relevant examination criterion.*

Main question 1

Examination criterion: *Redogöra för olika frågor som måste hanteras i program med samtidigt exekverande processer, inklusive tävlan om resurser och ömsesidig uteslutning*

Sub-question 1a

Define the two properties *liveness* and *absence of unnecessary delay* in your own words. Make sure that your answer is detailed enough to separate the two properties.

Sub-question 1b

Consider the following psudo-code for a simple lock between 2 threads. Will this work? If not, why not? Don't forget to motivate your answer with respect to all three safety properties, mutual exclusion, absence of deadlock and absence of unnecessary delay.

```
bool flag1 = false
bool flag2 = false
```

```
int turn = 1
void lock(bool myflag,bool itsflag, int me)
  myflag = true
  turn = me
  while (itsflag and turn = me){
     skip // No-op
  }
}
```

```
void unlock(){
  flag=false;
}
```

Sub-question 1c

Schedules are used to wake and suspend processes in concurrent programs. Describe *unconditionally fair* schedules and *weakly fair* schedules, including the limits of each.

Main question 2

Examination criterion: *Identifiera, beskriva och diskutera klassiska synkroniseringsproblem mellan parallella processer såsom synkronisering av läsare och skrivare eller av producenter och konsumenter*

Sub-question 2a

In the below pseudo code please answer the following questions; what is the classic problem, is mutual exclusion ensured, is deadlock possible, is starvation possible? Please note that

- the function "find_channel_for" will map an integer to a channel, e.g. if you pass 1 it would return Chan1.
- that the function try_pop is not fully defined. If it cannot remove an element, please assume the code will not give null-pointers but will act appropriately, though this sketch does not explicitly say so.

```
chan Chan1,Chan2,Chan3,Chan4;
int[] ww,rw,wa,ra;
```

```
Process 1{
 Loop{
   Chan4.send([“r”,1])
   Chan1.recieve()
   // ACT
   Chan4.send([“er”,1])
  // ACT
 }
}
```

```
Process 2{
 Loop{
   Chan4.send([“r”,2])
   Chan2.recieve()
   // ACT
   Chan4.send([“er”,2])
  // ACT
 }
}
```

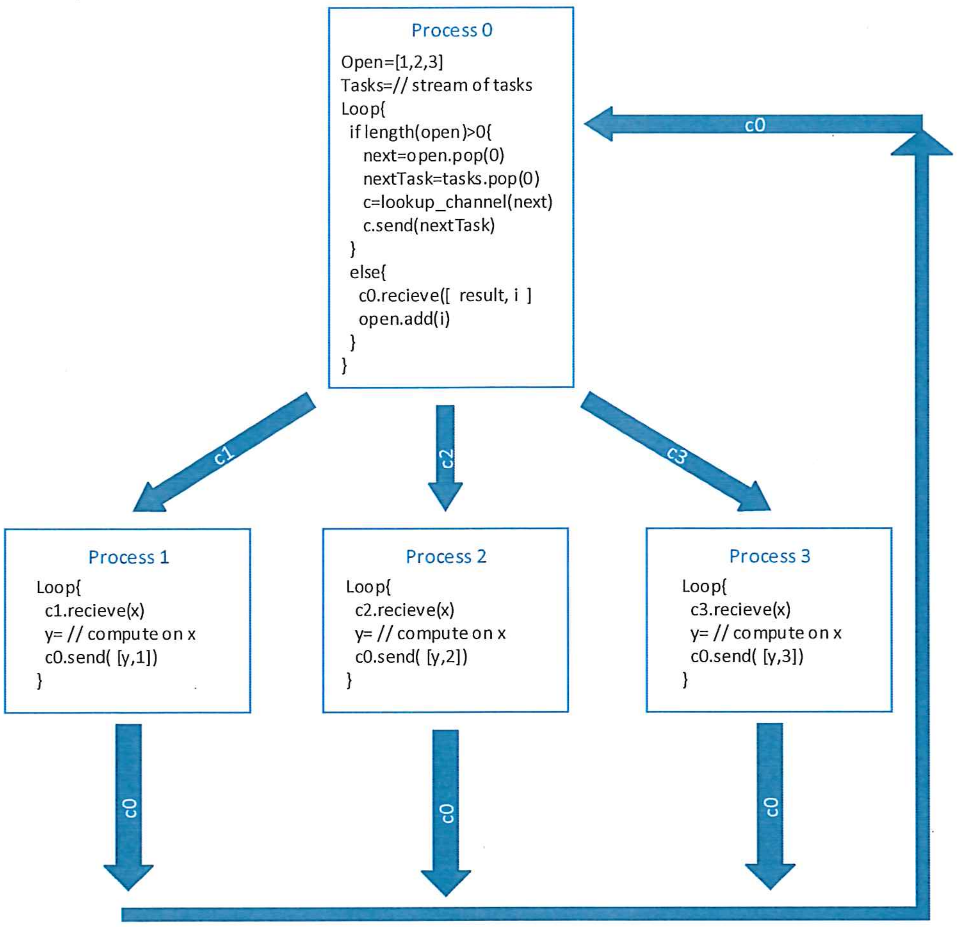
```
Process 3{
 Loop{
   Chan4.send([“w”,3])
   Chan3.recieve()
   // ACT
   Chan4.send([“ew”,3])
  // ACT
 }
}
```

```
Process 4{
 Loop{
  Chan4.recievie[t,p]
  if t==”ew”{ wa.remove(p) }
   elif t==”er”{ ra.remove(p) }
   elif t==”r”{ rw.add(p)}
   elif t==”w” {ww.add(p)}
  if length(wa)>0{ // noop }
  elif length(ra)>0{
   nr=try_pop(rw)
   c=find_channel_for(nr)
   ra.add(nr)
   c.send(“go”)
  }
  elif length(ww)>0{
   nw=try_pop(ww)
   c=find_channel_for(nw)
   wa.add(nw)
   c.send(“go”)
  }
  elif length(rw)>0{
   nr=try_pop(rw)
   c=find_channel_for(nr)
   ra.add(nr)
   c.send(“go”)
  }
 }
}
```

Sub-question 2b

Describe the worker-manager classic problem model, including how the system can recover from mistakes or breakdowns in the worker processes. Describe how you might implement this using binary semaphores.

Sub-question 2c



Process 0

```
Open=[1,2,3]
Tasks=// stream of tasks
Loop{
 if length(open)>0{
   next=open.pop(0)
   nextTask=tasks.pop(0)
   c=lookup_channel(next)
   c.send(nextTask)
 }
 else{
   c0.recieve([ result, i ]
   open.add(i)
 }
}
```

Process 1

```
Loop{
  c1.recieve(x)
  y= // compute on x
  c0.send( [y,1])
}
```

Process 2

```
Loop{
  c2.recieve(x)
  y= // compute on x
  c0.send( [y,2])
}
```

Process 3

```
Loop{
  c3.recieve(x)
  y= // compute on x
  c0.send( [y,3])
}
```

Consider the system of processes seen in the diagram above. What classic problem is this? What architectural pattern is this following? What is the purpose of the messages sent on channel c0?

Parallella processer G1F, IT404G. 2024-01-10

Main question 3

***Examination criterion:** Beskriva för- och nackdelar med olika tekniker för att lösa synkroniseringsproblem, inklusive semaforer, monitorer och tekniker för meddelandeöverföring*

Sub-question 3a

Monitors are similar to Conditional Critical Regions. Please describe each. Compare and contrast the two mechanisms with respect to support for *mutual exclusion* and *efficiency*.

Sub-question 3b

Message passing can be either asynchronous or synchronous. Describe each of these. Compare and contrast the two mechanisms with respect to support for *mutual exclusion* and *efficiency*. Describe with examples where you could use each.

Sub-question 3c

Compare and contrast remote procedures calls (RPC) and rendezvous methods of concurrent programming in distributed systems with respect to support for *mutual exclusion* and *efficiency*. Where could you use each?

Main question 4

Examination criterion: *Använda grundläggande tekniker såsom semaforer och meddelandeöverföring för att lösa synkronisering och kommunikation i program med parallella processer*

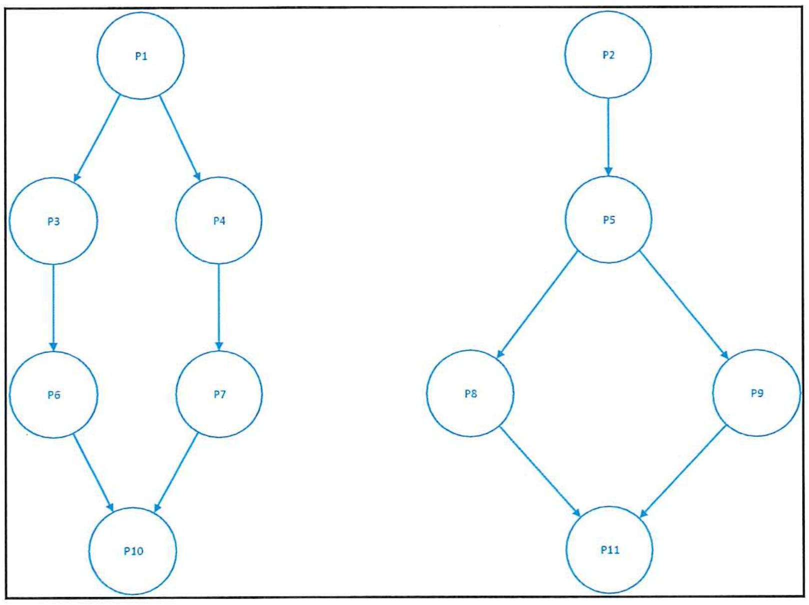
Sub-question 4a

There is a turn based computer game, and it can be played by many players at the same time. The programmers have used a channel abstraction to model the connections between the player processes and a peer-peer based architecture.

Draw a diagram of the processes and the channels linking them and then write some pseudo code for each process to indicate how they will proceed. Assume there are 3 players!

Sub-question 4b

What follows is a diagram of a set of processing nodes. Synchronization will be achieved using semaphores. Please state the number of semaphores required, their initial states and write pseudocode for each process. Please note that the arrows mean that the later process should not begin until ALL the earlier connected processes have completed.

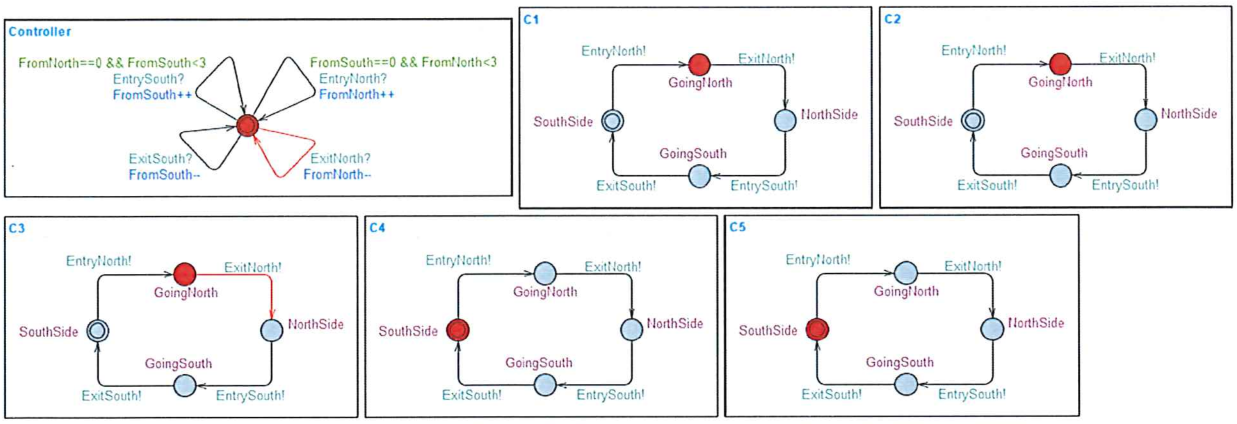


Sub-question 4c

A programmer has found an instance of the dining philosophers classic problem in an application they are writing. They have decided to use a monitor as a controller for the processes. Implement a monitor for this controller, with 2 methods; one to take a fork (indicated by an integer parameter) and one to release a fork (indicated by an integer parameter). You may assume 5 "philosopher processes are active and so there are 5 forks. The system will need 1 condition variable for each fork.

Main question 5

Examination criterion: Modellera och verifiera egenskaper hos program med parallella processer, inklusive progression, frånvaro av låsning och ömsesidig uteslutning



Consider the above model, which model a solution to the old bridge problem. It has a controller process with two local variables, FromNorth and FromSouth. Both of these are initiated to 0. The model also has five instances of car processes. Cars in state GoingNorth or GoingSouth are on the old bridge. **Note** that the system uses the inbuilt mechanisms ! and ? for synchronizations (i.e., not the emulated semaphore or asynchronous message passing mechanisms from assignment). Hence, two processes will synchronize directly with each other on e.g., channel ExitNorth in the above figure.

Sub-question 5a

Explain the role of the controller and describe what it does in detail.

Sub-question 5b

Formulate an uppaal query to verify that if there can be at most three cars on the bridge at the same time.

Sub-question 5c

Formulate an uppaal query to verify that if there are cars on the bridge going south, there cannot be cars on the bridge going in the opposite direction.