

The Australian National University
Final Examination – September 2011

COMP 2310 / COMP 6310

Concurrent and Distributed Systems

Study period: 15 minutes
Time allowed: 3 hours
Total marks: 100
Permitted materials: None

Questions are **not** equally weighted – sizes of answer boxes do **not** necessarily relate to the number of marks given for this question.

All your answers must be written in the boxes provided in this booklet. You will be provided with scrap paper for working, but only those answers written in this booklet will be marked. Do not remove this booklet from the examination room. There is additional space at the end of the booklet in case the boxes provided are insufficient. Label any answer you write at the end of the booklet with the number of the question it refers to.

Greater marks will be awarded for answers that are simple, short and concrete than for answers of a sketchy and rambling nature. Marks will be lost for giving information that is irrelevant to a question.

Student number:

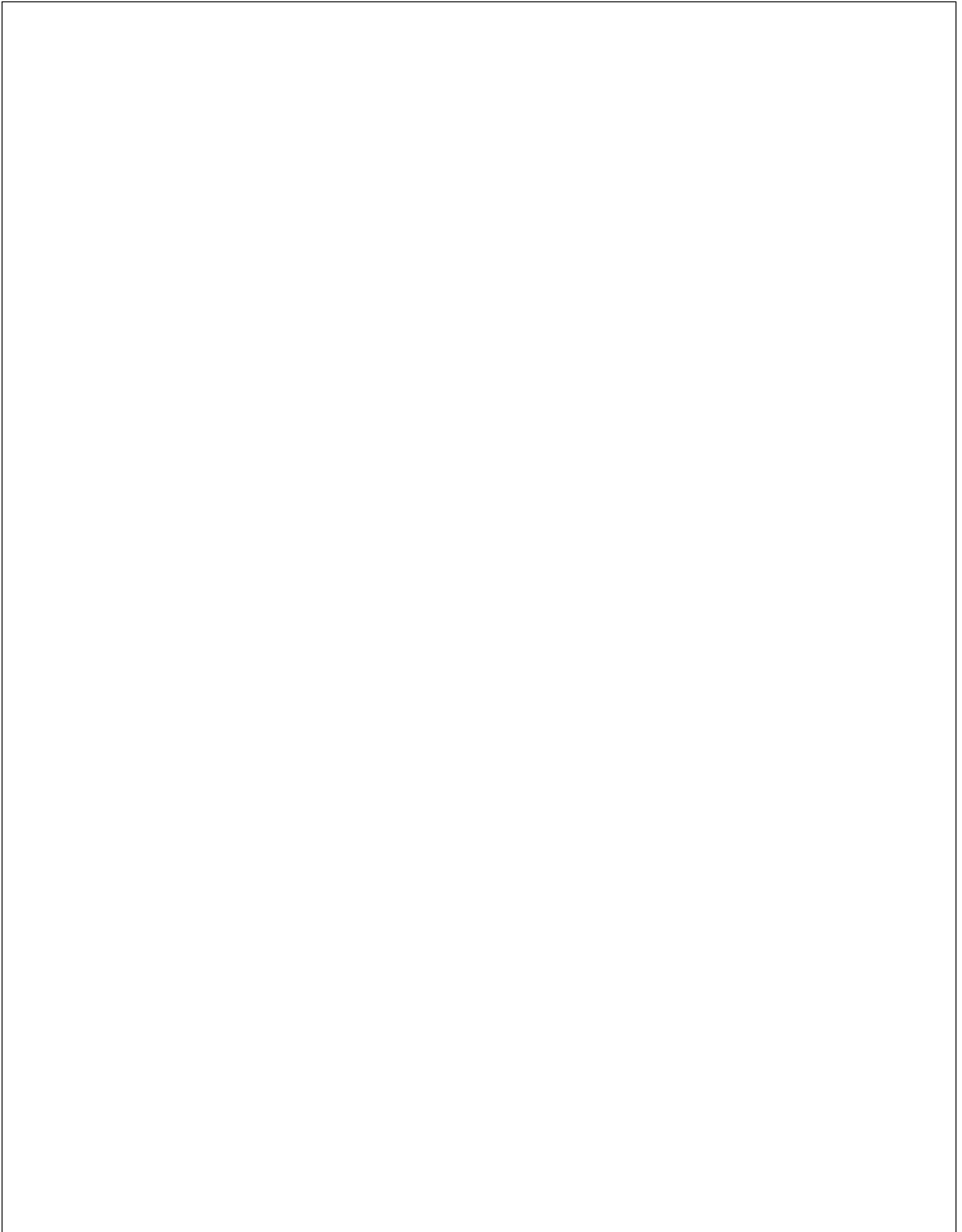
The following are for use by the examiners

<i>Q1 mark</i>	<i>Q2 mark</i>	<i>Q3 mark</i>	<i>Q4 mark</i>	<i>Q5 mark</i>	<i>Q6 mark</i>
----------------	----------------	----------------	----------------	----------------	----------------

Total mark

1. [16 marks] General concurrency

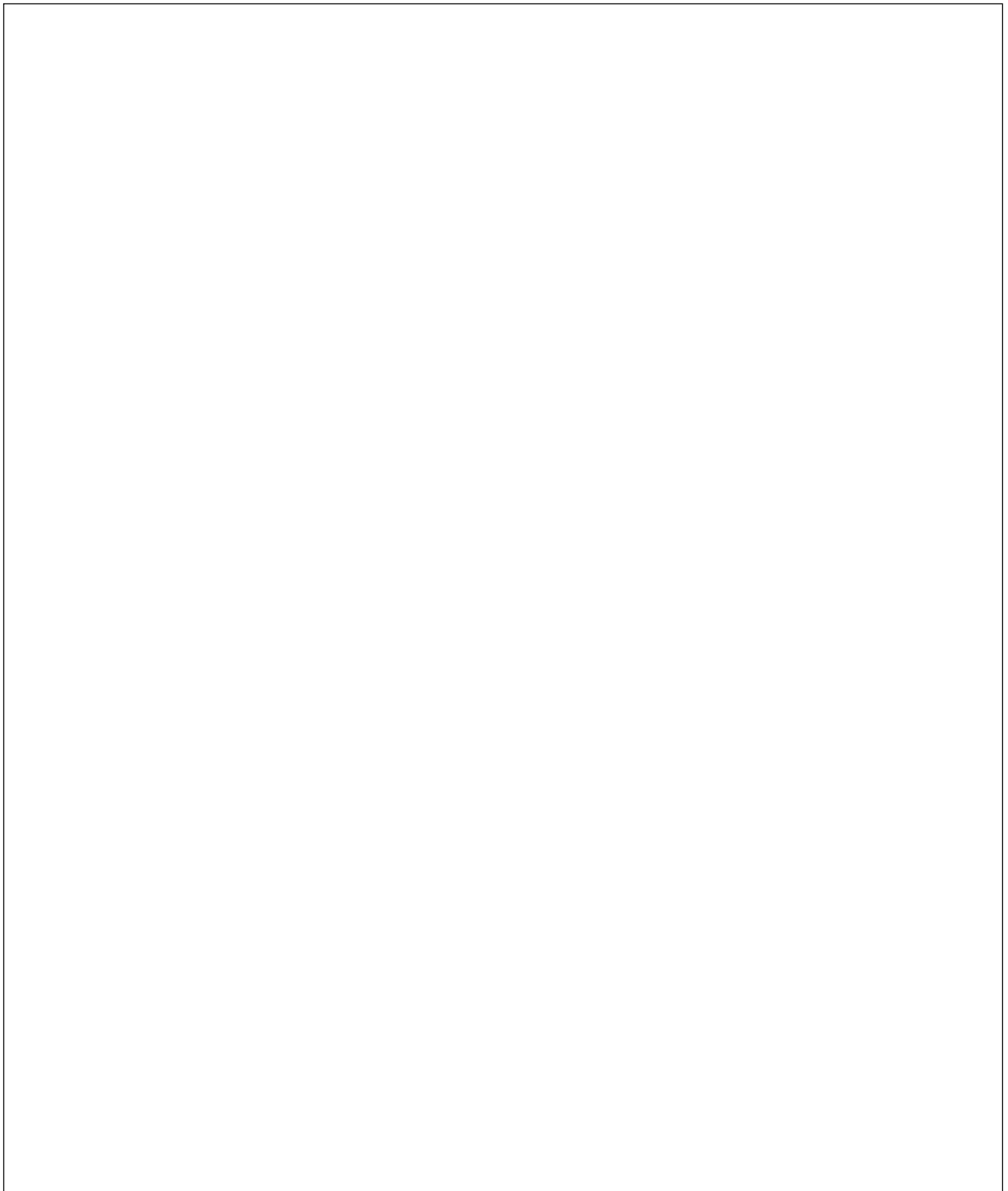
- (a) [5 marks] Define “Process”, enumerate its states and depict the potential transitions between those states. Label the transitions with events which trigger those transitions.



(b) [5 marks] Which of the following architectures require or are supportive for concurrent programming?

Pipelines, Vector processors, Hyper-threading,
Out of order execution, Multiple cores, Virtual memory

Give precise reasons.



- (c) [6 marks] Programming languages support concurrency in various ways. Pick three languages or libraries which you are familiar with and state briefly in which form they support concurrent programming. You only need to provide sufficient detail to make the languages / libraries distinguishable. Therefore, do not choose languages / libraries which are identical in terms of concurrency.

2. [6 marks] Mutual exclusion

- (a) [6 marks] Define the term “semaphore”. Does the implementation of a semaphore require special hardware support or can a semaphore be implemented in any high-level programming language? Give precise reasons.

3. [26 marks] Synchronization

- (a) [8 marks] Ten processes are part of a concurrent calculation. Each individual calculation will also provide a boolean value which indicates the success or failure of that individual calculation. Only if all ten processes complete their individual calculations successfully, then the whole calculation is declared successful. Program a mechanism which can be used by those ten processes such that all of them will wait for the overall result and know about an overall success or an overall failure. Use a synchronization primitive and language (including pseudocode) of your choice.

(b) [2 marks] What is the “requeue” facility (as in Ada) and why is it required? Name an example of a concurrent system which requires “requeue” and explain why it is needed there.

(c) [16 marks] Read the following Ada program carefully. The program is syntactically correct and will compile without warnings. See questions on the following pages.

```

with Ada.Text_IO; use Ada.Text_IO;

procedure Chain_Processor is

  n : constant Positive := 10;
  subtype Counters is Natural range 0 .. n;
  subtype Stages is Natural range 1 .. n;

  type Stage;
  type Link is access Stage;

  task type Stage (Next_Stage : Link := null) is
    entry Pipe (x_in : Natural;
               y_in : Stages := Stages'Last;
               z_in : Positive := 1);
  end Stage;

  task body Stage is

    x : Natural;
    y : Counters;
    z : Positive;

  begin
    loop
      select
        accept Pipe (x_in : Natural;
                    y_in : Stages := Stages'Last;
                    z_in : Positive := 1) do
          x := x_in;
          y := y_in - 1;
          z := z_in * x_in;
        end Pipe;
      or
        terminate;
      end select;

      if y = 0 then
        Put_Line (" -> result:" & Positive'Image (z));
      else
        Put (" ->" & Positive'Image (z));
        Next_Stage.all.Pipe (x, y, z);
      end if;
    end loop;
  end Stage;

  Pipeline : array (Stages) of Link;

begin
  Pipeline (Pipeline'Last) := new Stage;
  for Id in reverse Pipeline'First .. Pipeline'Last - 1 loop
    Pipeline (Id) := new Stage (Pipeline (Id + 1));
  end loop;

  declare
    Entry_Stage : Stage renames Pipeline (Pipeline'First).all;
  begin
    Entry_Stage.Pipe (2);
    Entry_Stage.Pipe (10, 2);
    Entry_Stage.Pipe (2, 8);
  end;
end Chain_Processor;

```


(i) [2 marks] How many tasks are implemented in this program? Name them.

(ii) [4 marks] Are the tasks in any way synchronized? Explain your answer. If they are synchronized then explain by which means and for how long exactly.

(iii) [3 marks] Is there a (potential) deadlock or raised exception in this program? Explain your answer.

(iv) [7 marks] What is the expected output? Is this output deterministic? Which function is implemented by this program?

4. [10 marks] Scheduling

- (a) [3 marks] Which scheduling algorithm would you suggest in order to minimize the maximum turnaround time for a task set of unknown characteristics. Give precise reasons.

- (b) [3 marks] If you know the exact computation times of all tasks in a task set, would you change to a different scheduling algorithm in question (a) (while still minimizing the maximum turnaround time)? If so: to which other scheduling algorithm? Give precise reasons.

- (c) [4 marks] Feedback scheduling (as introduced in the lectures) has a potentially severe problem. Which problem is that? And how would you change/amend the basic feedback scheduling algorithm in order to overcome this problem? Give precise reasons for your suggested changes.

5. [8 marks] Safety and Liveness

(a) The code fragment below depicts a classical deadlock example.

```
var resource_1, resource_2, resource_3 : semaphore := 1;

process P1;                process P2;                process P3;
  statement X;              statement A;              statement K;
  wait (resource_1);        wait (resource_2);        wait (resource_3);
  wait (resource_2);        wait (resource_3);        wait (resource_1);
  statement Y;              statement B;              statement L;
  signal (resource_2);      signal (resource_3);      signal (resource_1);
  signal (resource_1);      signal (resource_2);      signal (resource_3);
  statement Z;              statement C;              statement M;
end P1;                     end P2;                   end P3;
```

(i) [2 marks] Will this example code always deadlock? Explain your answer or give a counter example if it does not always deadlock.

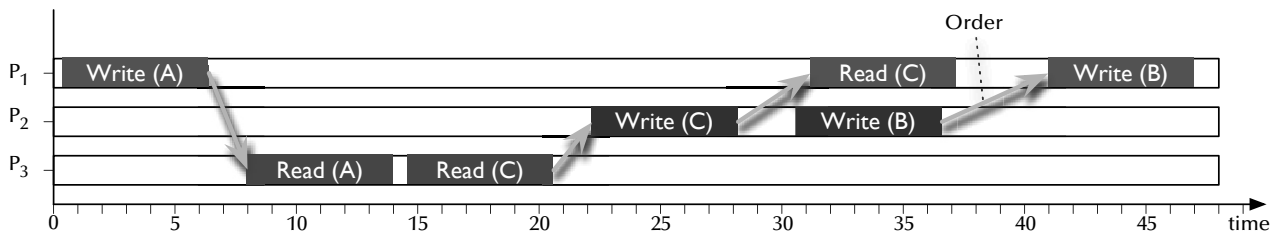
(ii) [6 marks] Suggest at least three, principally different ways to change/amend the code on the previous page so that all processes will gain access to the requested resources and deadlocks are prevented. You are allowed to exchange the semaphores with other means of synchronization.

6. [34 marks] Distributed Systems

(a) [6 marks] Three transactions:

- 1: Write (A) - Read (C) - Write (B)
- 2: Write (C) - Write (B)
- 3: Read (A) - Read (C)

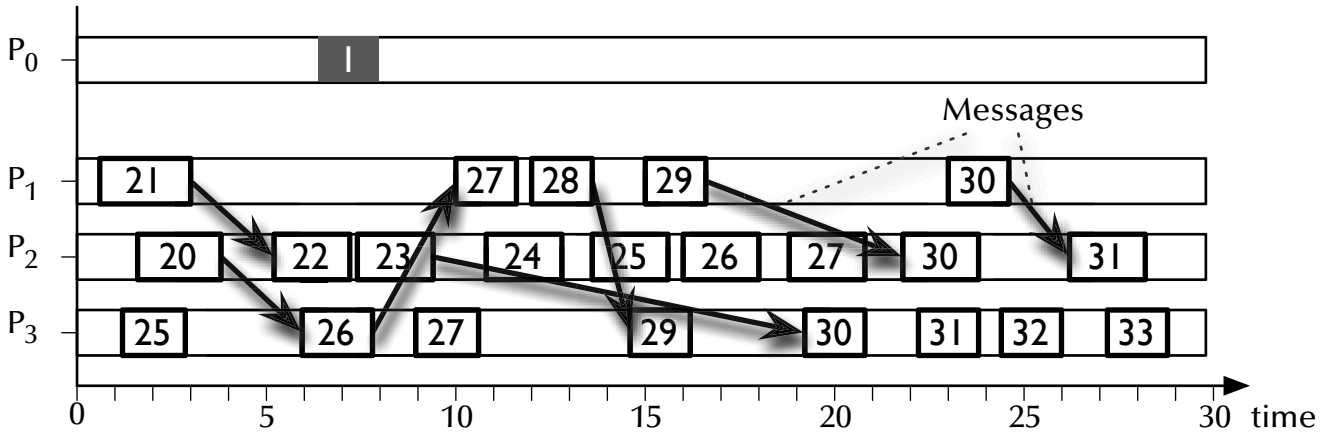
are executed by three different processes resulting in the following critical pairs:



Are any two of those transactions serializable? Is the whole set of transactions serializable? Explain your answers precisely.

(b) [10 marks] Process 0 in the diagram below has been tasked to take a snapshot of the processes set process 1 to process 3. Only message passing is available to perform this task. Events inside each tasks are carrying a logical time-stamp.

(i) [4 marks] Assume finite message speeds (meaning they cannot be received instantaneously) and draw into the diagram below how the snapshot will be assembled.



(ii) [2 marks] Which time-stamp out of each process will be the latest time stamp from the past with respect to the recorded snapshot?

(iii) [4 marks] Can a snapshot in a distributed system which has been assembled by means of a message passing system ever relate to a single, global time? Explain why this would be possible or not be possible. If you need to assume something for your answer then state your assumptions.

(c) [18 marks] Read the following Ada program carefully. The program is syntactically correct and will compile without warnings. See questions on the following pages.

```

with Ada.Text_IO; use Ada.Text_IO;

procedure Test_Selects is

  type Workers_Range is
    new Positive range 1 .. 2;
  type Clients_Range is
    new Positive range 1 .. 4;

  task type Worker is
    entry Ready;
    entry Service;
  end Worker;

  task body Worker is

begin
  loop
    select
      accept Ready;
    or
      accept Service do
        delay 2.0;
        Put (" W");
      end Service;
    or
      terminate;
    end select;
  end loop;
end Worker;

Workers : array (Workers_Range) of Worker;

task Server is
  entry Service;
end Server;

task body Server is

begin
  loop
    select
      accept Service do
        for i in Workers_Range loop
          select
            Workers (i).Ready;
            requeue
              Workers (i).Service;
          else
            null;
          end select;
        end loop;
      end Service;
    or
      terminate;
    end select;
  end loop;
end Server;

task type Client;

task body Client is

begin
  delay 1.0;
  for k in 1 .. 2 loop
    Server.Service;
  end loop;
  delay 3.0;
  Put (" T");
end Client;

Clients : array (Clients_Range) of Client;

begin
  null;
end Test_Selects;

```

(i) [2 marks] How many task queues are implemented in this program? Name them.

(ii) [4 marks] Which of the entries in this program are (potentially) blocking? Explain for each entry why they are blocking or non-blocking.

(iii) [4 marks] Name all sources of non-determinism in this program. Will this program never / sometimes / always terminate? Explain your answer.

(iv) [8 marks] Develop the sequence of outputs from this program on a time-line. If multiple outputs are possible then show them all. Divide your time-line(s) into stages and explain what happens in each stage.

A large empty rectangular box with a thin black border, intended for the student to draw a timeline showing the sequence of outputs from a program. The box is currently blank.

continuation of answer to question part

continuation of answer to question part

continuation of answer to question part

continuation of answer to question part

continuation of answer to question

part

continuation of answer to question

part

continuation of answer to question

part

continuation of answer to question

part

continuation of answer to question

part

continuation of answer to question

part