CS/063174

UNIVERSITY OF YORK

BSc, BEng and MEng Degree Examinations Example

COMPUTER SCIENCE
Finals

Real-Time Systems and their Programming Languages

Time allowed: 3 hours

Silent non-programmable calculators may be used.

Candidates should answer not more than **four** questions

1 (25 marks)

(i) [12 marks] The following formulae are used to analyse a fixed priority systems of tasks to determine if they will meet their completion time requirements:

$$R_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \quad (1)$$

$$R_i \leq D_i \quad (2)$$

(a) Define all the terms used in (1) and (2).

(b) How would the C_i term be determined?

(c) How would the B_i term be determined?

(d) How could equation (1) be solved?

(ii) [5 marks] What is release jitter, and how could equation (1) be modified to include this behaviour ?

continued

(iii) [8 marks] How can the following task set be optimally scheduled (using fixed priority scheduling)?

	T	C	B	D
Task_1	8	4	2	8
Task_2	10	2	2	5
Task_3	30	5	2	30

Is this task set schedulable? Explain your answer in detail.

2 (25 marks)

(i) [10 marks] Interfacing to 'time' is fundamental in real-time systems. Describe the features that an ideal programming language should support. How well does Ada compare with this ideal?

(ii) [7 marks] The following code implements a Modula-1 device module for a hardware clock that interrupts 20 times per second.

```
DEVICE MODULE hardwareclock[6];
  DEFINE tick;
  VAR tick : SIGNAL;
  PROCESS handler[120B];
    VAR count : INTEGER;
        StatusRegister[167444B] : BITS;
  BEGIN
    count := 0;
    StatusRegister[6] := TRUE;
    LOOP
      DOIO;
      count := count + 1;
      IF count = 10 THEN
        WHILE AWAITING(tick) DO
          SEND(tick)
        END;
        count := 0;
      END
    END
  END handler;
BEGIN
  handler
END hardwareclock;
```

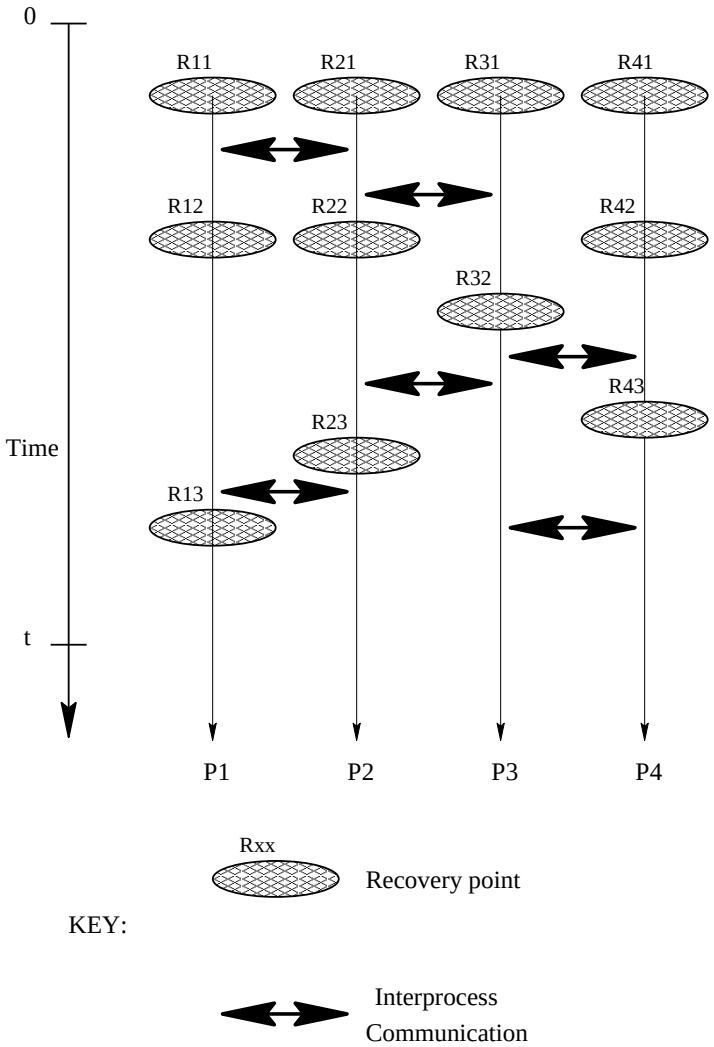
Describe the behaviour of this code.

(iii) [8 marks] Modify the device module just given so that it supports a DELAY procedure which can be called concurrently by many processes. This procedure takes an integer parameter (SEC). When it is called the invoking process is delayed for the defined number of seconds (SEC). The procedure returns when the delay has expired.

3 (25 marks)

(i) [8 marks] Explain the terms: **recovery point**, **checkpointing**, the **domino effect** and **recovery lines**.

(ii) [7 marks] The following diagram illustrates the concurrent execution of four communicating processes (P1, P2, P3 and P4) and their associated recovery points (for example, R11 is the first recovery point for process P1).



Explain what happens when an error is detected by

- Process P1 at time t.
- Process P2 at time t.

- Process P3 at time t.
- Process P4 at time t.

(iii) [10 marks] Consider the following program:

```
I : Integer := 1;
J : Integer := 0;
K : Integer := 3;

procedure Primary is
begin
    J := 20;
    I := K*J/2
end Primary;

procedure Secondary is
begin
    I := J;
    K := 4;
end Secondary;

procedure Tertiary is
begin
    J := 20;
    I := K*J/2
end Tertiary;
```

This code is to be executed in a recovery block environment AND an exception handling environment. The following is the recovery block environment:

```
ensure I = 20
by
    Primary;
else by
    Secondary;
else by
    Tertiary;
else
    I := 20;
end;
```

The following is the exception handling environment:

```
Failed : exception;
type Module is (P,S,T);

for Try in Module loop
    begin
        case Try is
            when P =>
                Primary;
                if I /= 20 then
                            raise Failed;
                end if;
                exit;
            when S =>
                Secondary;
                if I /= 20 then
                            raise Failed;
                end if;
                exit;
            when T =>
                Tertiary;
                if I /= 20 then
                            raise Failed;
                end if;
                exit;
            end case;
    exception
        when Failed =>
            if Try = T then
              I := 20;
            end if;
    end;
end loop;
```

Assuming the *Termination* model of exception handling, discuss the execution of BOTH the recovery block environment and the exception handling environment code fragments.

4 (25 marks)

(i) [10 marks] Compare and contrast protected objects in Ada 95 and Mutexes and Condition Variables in POSIX.

(ii) [5 marks] Explain the synchronisation imposed by the following Ada protected object.

```
protected type Barrier(Needed : Positive) is
    entry Wait;
private
    Releasing : Boolean := False;
end Barrier;

protected body Barrier is

    entry Wait when Wait'Count = Needed or Releasing is
    begin
        if Wait'Count = 0 then
            Releasing = False;
        else
            Releasing := True;
        end if;
    end Wait;
end Barrier;
```

(iii) [10 marks] The following package provides a simplified Ada binding to POSIX Mutexes and Condition Variables. All Mutexes and Condition Variables are initialised with default attributes.

```
package Pthreads is
    type Mutex_T is limited private;
    type Cond_T is limited private;

    procedure Mutex_Initialise (M: in out Mutex_T);
    procedure Mutex_Lock(M: in out Mutex_T);
    procedure Mutex_Trylock(M: in out Mutex_T);
    procedure Mutex_Unlock(M: in out Mutex_T);

    procedure Cond_Initialise(C: in out Cond_T);
    procedure Cond_Wait(C: in out Cond_T;
                        M : in out Mutex_T);

    procedure Cond_Signal(C: in out Cond_T);
    procedure Cond_Broadcast(C: in out Cond_T);
private
   ...
end Pthreads;
```

Show how Barriers, as defined in Part (ii), can be implemented using this package. Do NOT use any of Ada's communication and synchronisation facilities in your solution.

5 (25 marks)

(i) [9 marks] Explain the facilities provided by the following occam2 processes:

- ALT
- PRI ALT
- replicated ALT

(ii) [6 marks] Show how a bounded buffer can be implemented in occam2.

(iii) [10 marks] Consider the following occam2 program fragment.

```
INT Number.OF.Procs is 10:
-- other declarations, if needed

PROC Controller
   -- code for Controller procedure
:

PROC Stop.Go(VAL INT Id)
   -- code for procedure
:

PAR
  PAR i = 0 FOR Number.OF.Procs
    SEQ
      -- Part A code for user process
      Stop.Go(i);
      -- Part B code for user process
    Controller
```

This produces 10 user processes (which all execute their Part A code and then call the Stop.Go procedure, and then execute their Part B code) and a Controller process. The goal of the Stop.Go and Controller procedures is to ensure that AT LEAST 6 of the user processes execute their Part A before any execute their Part B.

Show how the Stop.Go and Controller procedures can be implemented. If necessary, introduce further declarations.

6 (25 marks)

(i) [4 marks] Explain the user needs for an asynchronous event handling mechanism.

(ii) [9 marks] Explain carefully the execution of Ada's asynchronous select statement.

(iii) [12 marks] Consider the following four fragments of code.

(a)

```
select
    T.Call; -- an entry call to task T
    Flag := A;
or
    delay 10.0;
    Flag := B;
    -- code taking 2 seconds to execute
end select;
```

(b)

```
select
    T.Call; -- an entry call to task T
    Flag := A;
else
    delay 10.0;
    Flag := B;
    -- code taking 2 seconds to execute
end select;
```

(c)

```
select
    T.Call; -- an entry call to task T
    Flag := A;
then abort
    delay 10.0;
    Flag := B;
    -- code taking 2 seconds to execute
end select;
```

(d)

```
select
    delay 10.0;
    Flag := A;
then abort
    T.Call; -- an entry call to task T
    Flag := B;
    -- code taking 2 seconds to execute
end select;
```

A rendezvous with `T.Call` takes 5 seconds to execute. What is the value of the `Flag` variable after the execution of each of the four fragments in each of the following cases. Explain your answers. You may assume a `Flag` assignment statement takes zero execution time.

(i) `T.Call` is available when the select is executed.

(ii) `T.Call` is NOT available when the select is executed and does not become available in the next 14 seconds.

(iii) `T.Call` is NOT available when the select is executed, but does become available after 2 seconds.

(iv) `T.Call` is NOT available when the select is executed but does become available after 8 seconds.