063174

THE UNIVERSITY of York

Degree Examinations 2000

DEPARTMENT OF COMPUTER SCIENCE

Real-Time Systems and Programming Languages

Time allowed: **Three (3) hours**

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

1 (25 marks)

(i) [15 marks] Consider the following package specification which provides a procedure to search part of a large character array for a *unique* fixed-length string. The procedure returns the position of the start of the string if it is found.

```
package Search_Support is
  type Array_Bounds is range 1 .. 1_000_000_000;
  type Large_Array is array(Array_Bounds) of Character;
  type Pointer is access Large_Array;

  type Search_String is new String(1..10);

  procedure Search(Pt: Pointer;
                   Lower, Upper: Array_Bounds;
                   Looking_For : Search_String;
                   Found : out Boolean;
                   At_Location : out Array_Bounds);
end Search_Support;
```

Three tasks wish to perform a concurrent search of the array for the same string; they are derived from a common task type:

```
task type Searcher(Search_Array: Pointer;
                   Lower, Upper: Array_Bounds) is
  entry Find(Looking_For : Search_String);
  entry Get_Result(At_Location : out Array_Bounds);
end Searcher;
```

The string to be found is passed via an initial rendezvous with the tasks.

Sketch the body of the task type (and any other objects you might need) so that when one task finds the string, all other tasks are IMMEDIATELY informed of the string's location so that further fruitless search is avoided. You may assume that the `Search_String` will be found by one of the three tasks. Furthermore, all tasks must be prepared to pass back the result via the `Get_Result` entry.

(ii) [10 marks] The task body in your solution to Part (i) makes use of a particular type of Ada select statement. Explain carefully the full semantics of this type of select statement.

2 (25 marks)

(i) [15 marks] Explain the facilities provided by POSIX mutexes and condition variables. Use the bounded buffer as an example of how they can be used.

(ii) [10 marks] Show how POSIX mutexes and condition variables can be used to implement a resource controller where several identical resources can be allocated and freed. Clients can request and free one or more resource using the following interface:

```
typedef struct {
    ... /* you fill in */
  } resource;

void allocate(int size, resource *R);

void deallocate(int size, resource *R);

void initialize(resource *R);
```

You may ignore the issues associated with initialising mutexes and condition variables. You should also ignore the physical allocation and deallocation of the resource and focus on the synchronization aspects.

3 (25 marks)

(i) [10 marks] Discuss the various models by which a language can represent an interrupt handler; be sure you include in your discussion the Ada 95, occam2 and Modula-1 models.

(ii) [15 marks] Consider a 16 bit memory mapped computer which is embedded in a patient monitoring system. The system is arranged so that an interrupt is generated at the highest hardware priority, through vector location 100 (octal), every time the patient's heart beats. In addition, a mild electric shock can be administered via a device control register, the address of which is 177760 (octal). The register is set up so that every time an integer value 'x' is assigned to it the patient receives 'x' volts over a small period of time.

If no heart beat is recorded within a 5 second period then the patient's life is in danger. Two actions should be taken when the patient's heart fails: the first is

Continued.

that a ‘supervisor’ process should be notified so that it may sound the hospital alarm; the second is that a single electric shock of 5 volts should be administered. If the patient fails to respond then the voltage should be increased by 1 volt for every further 5 seconds.

Write an occam2 program which monitors the patient’s heart and initiates the actions described above. You may assume that communication with the supervisor process is via the following channel:

```
CHAN OF ANY SOUND.ALARM
```

4 (25 marks)

(i) [10 marks] Explain the differences and similarities between the following two Ada program fragments.

```
-- Fragment A
task Count is
  entry Increment;
  entry Decrement;
  entry Value(C : out Integer);
end Count;

task body Count is
  Current : Integer := 0;
begin
  loop
    select
      accept Increment do
        Current := Current + 1;
      end Increment;
    or
      accept Decrement do
        Current := Current - 1;
      end Decrement;
    or
      accept Value(C : out Integer) do
        C := Current;
      end Value;
    end select;
  end loop;
end Count;
```

```
-- Fragment B

protected Count is
  procedure Increment;
  procedure Decrement;
  function Value return Integer;
private
  Current : Integer := 0;
end Count;

protected body Count is
  procedure Increment is
  begin
    Current := Current + 1;
  end Increment;

  procedure Decrement is
  begin
    Current := Current - 1;
  end Decrement;

  function Value return  Integer is
  begin
    return Current;
  end Value;
end Count;
```

(ii) [15 marks] It has been suggested that York should put a limit on the number of motorists that can enter into the city at any one time. One proposal is to establish monitoring checkpoints at each of the city's Bars (entrances through the City's walls) and, when the city is full, to turn the traffic lights to red for incoming traffic. To indicate to the motorists that the city is full, the red light is set to flashing.

In order to achieve this goal, pressure sensors are placed in the road at the Bars' entry and exit points. Every time a car enters the city, a signal is set to a `Bar_Controller` task (as a task entry call); similarly when a car exits:

```
Maximum_Cars_In_City_For_Red_Light : constant Positive := N;
Minimum_Cars_In_City_For_Green_Light : constant Positive := N - 10;


type Bar is (Walmgate, Goodramgate, Micklegate,
            Bootham, Barbican);

task type Bar_Controller(G : Bar) is
  entry Car_Entered;
  entry Car_Exited;
end Bar_Controller;

Walmgate_Bar_Controller : Bar_Controller(Walmgate);
Goodramgate_Bar_Controller : Bar_Controller(Goodramgate);
Micklegate_Bar_Controller : Bar_Controller(Micklegate);
Bootham_Bar_Controller : Bar_Controller(Bootham);
Barbican_Bar_Controller : Bar_Controller(Barbican);
```

Show how the body of these tasks can be co-ordinated so that one of them calls the `City_Traffic_Light_Controller` (with the following task specification) to indicate whether more cars are to be allowed in or not.

```
task City_Traffic_Lights_Controller is
  entry City_Is_Full;
  entry City_Has_Space;
end City_Traffic_Lights_Controller;

task body Traffic_Lights_Controller is separate;
--body of no interest in this question
```

5 (25 marks)

(i) [5 marks] Distinguish between base and active priority, and give an example of when the base and active priority of a process might be different

(ii) [5 marks] What is the difference between a task dispatching mechanism and a scheduling policy? Does Ada define a scheduling policy?

(iii) [15 marks] Ada allows the base priority of a task to be set dynamically using the following package.

```
with Ada.Task_Identification;
with System;
package Ada.Dynamic_Priorities is

   procedure Set_Priority(Priority : System.Any_Priority;
                T : Ada.Task_Identification.Task_Id :=
                Ada.Task_Identification.Current_Task);
     -- raises Program_Error if T is the Null_Task_Id
     -- has no effect if the task has terminated

   function Get_Priority(T : Ada.Task_Identification.Task_Id :=
            Ada.Task_Identification.Current_Task)
            return System.Any_Priority;
     -- raises Tasking_Error if the task has terminated
     -- or Program_Error if T is the Null_Task_Id
private
  ... -- not required for this question
end Ada.Dynamic_Priorities;
```

Using this package, show how to implement a mode change protocol where a group of tasks must have their priorities changed as a single atomic operation.

6 (25 marks)

(i) [5 marks] Briefly compare and contrast the use of recovery blocks and exception handlers for programming fault tolerance.

(ii) [5 marks]

A set of independent reliable periodic processes, executing on a preemptive (fixed) priority-based kernel can have their deadline requirements checked by the use of *response-time analysis*. Define the equation used for response-time analysis (assuming all kernel overheads are negligible).

(iii) [5 marks]

Illustrate how the response time equation is used by calculating the worst-case response times (R) of the following process set (assuming priorities have been assigned in deadline monotonic order).

Process	Period T	Computation Time C	Deadline D	Response Time R
P_1	10	3	5	?
P_2	20	8	20	?
P_3	25	4	25	?

Table 1: Process Set

(iv) [10 marks] If a set of processes is subject to a single fault in one of the processes (that is tolerated by either the execution of a recovery block or an exception handler), consider how the response time equation can be modified to accommodate the extra computation. Would the above process set remain schedulable if there were a single fault and each of the processes had an exception handler? Assume the worst-case execution times for the handlers are 2, 3, and 1 respectively for the three process P_1, P_2 and P_3.