P5 – Rendezvous (ADA)

CONSIDERACIONES PARA RESOLVER LOS EJERCICIOS:

1. NO SE PUEDE USAR VARIABLES COMPARTIDAS
2. Declaración de tareas
   - Especificación de tareas sin ENTRY's (nadie le puede hacer llamados).

     TASK Nombre;

     TASK TYPE Nombre;

   - **Especificación de tareas con ENTRY's** (le puede hacer llamados). Los entry's funcionan de manera semejante los procedimientos: solo pueden recibir o enviar información por medio de los parámetros del entry. NO RETORNAN VALORES COMO LAS FUNCIONES

   TASK [TYPE] Nombre IS

     ENTRY e1;

     ENTRY e2 (p1: IN integer; p2: OUT char; p3: IN OUT float);

   END Nombre;

   - **Cuerpo de las tareas.**

   TASK BODY Nombre IS

     Codigo que realiza la Tarea;

   END Nombre;
3. Sincronización y comunicación entre tareas
   - **Entry call** para enviar información (o avisar algún evento).

     NombreTarea.NombreEntry (parametros);

   - **Accept** para atender un pedido de **entry call sin cuerpo** (sólo para recibir el aviso de un evento para sincronización). Lo usual es que no incluya parámetros, aunque podría tenerlos. En ese caso, son ignorados por no tener cuerpo presente.

     ACCEPT NombreEntry;

     ACCEPT NombreEntry (p1: IN integer; p3: IN OUT float);

   - **Accept** para atender un pedido de **entry call con cuerpo**.

     ACCEPT NombreEntry (p1: IN integer; p3: IN OUT float) do

       Cuerpo del accept donde se puede acceder a los parámetros p1 y p3.

       Fuera del entry estos parámetros no se pueden usar.

     END NombreEntry;

   - El accept se puede hacer en el cuerpo de la tarea que ha declarado el entry en su especificación. Los entry call se pueden hacer en cualquier tarea o en el programa principal.
   - Tanto **el entry call como el accept son bloqueantes, ambas tareas continúan trabajando cuando el cuerpo del accept ha terminado su ejecución.**
4. Select para ENTRY CALL.
   - **Select ...OR DELAY**: espera a lo sumo x tiempo a que la tarea correspondiente haga el accept del entry call realizado. Si pasó el tiempo entonces realiza el código opcional.

     SELECT

       NombreTarea.NombreEntry(Parametros);

       Sentencias;

     OR DELAY x

       Código opcional;

     END SELECT;

   - **Select ...ELSE**: si la tarea correspondiente no puede realizar el accept inmediatamente (en el momento que el procesador está ejecutando esa línea de código) entonces se ejecuta el código opcional.

     SELECT

       NombreTarea.NombreEntry(Parametros);

       Sentencias;

     ELSE

       Código opcional;

     END SELECT;

• En los select para entry call sólo puede ponerse un entry call y una única opción (OR DELAY o ELSE);

5. **Select para ACCEPT.**

• En los select para los accept puede haber más de una alternativa de accept, pero no puede haber alternativas de entry call (no se puede mezclar accept con entries). Cada alternativa de ACCEPT puede ser o no condicional (uso de cláusula WHEN).

```
SELECT
        ACCEPT e1 (parámetros);
        Sentencias1;
OR
        ACCEPT e2 (parámetros) IS cuerpo; END e2;
OR
        WHEN (condición) => ACCEPT e3 (parámetros) IS cuerpo; END e3;
        Sentencias3
END SELECT;
```

Funcionamiento: Se evalúa la condición booleana del WHEN de cada alternativa (si no lo tiene se considera TRUE). Si todas son FALSAS se sale del select. En otro caso, de las alternativas cuya condición es verdadera se elige en forma no determinística una que pueda ejecutarse inmediatamente (es decir que tiene un entry call pendiente). Si ninguna de ellas se puede ejecutar inmediatamente el select se bloquea hasta que haya un entry call para alguna alternativa cuya condición sea TRUE.

• Se puede poner una opción OR DELAY o ELSE (no las dos a la vez).

• Dentro de la condición booleana de una alternativa (en el **WHEN**) se puede preguntar por la **cantidad de entry call pendientes de cualquier entry de la tarea**.

```
NombreEntry'count
```

• Después de escribir una condición por medio de un WHEN siempre se debe escribir un accept.

1. Se requiere modelar un puente de un único sentido que soporta hasta 5 unidades de peso. El peso de los vehículos depende del tipo: cada auto pesa 1 unidad, cada camioneta pesa 2 unidades y cada camión 3 unidades. Suponga que hay una cantidad innumerable de vehículos (A autos, B camionetas y C camiones). Analice el problema y defina qué tareas, recursos y sincronizaciones serán necesarios/convenientes para resolver el problema.
a. Realice la solución suponiendo que todos los vehículos tienen la misma prioridad.

```
Procedure Puentecito is

TASK TYPE auto          //no tienen entries
TASK TYPE camioneta
TASK TYPE camion

TASK Puente IS
   ENTRY pasarAuto();
   ENTRY salirAuto();
   ENTRY pasarCamioneta();
   ENTRY salirCamioneta();
   ENTRY pasarCamion();
   ENTRY salirCamion();
end Puente;

autos = array (1..A) of auto;
camionetas = array (1..B) of camioneta;
camiones = array (1..C) of camion;

TASK BODY auto is
```

```
   Puente.pasarAuto();
   //pasar por el puente
   Puente.salirAuto();
end auto;

TASK BODY camioneta is
   Puente.pasarCamioneta();
   //pasar por el puente
   Puente.salirCamioneta();
end camioneta;

TASK BODY camion is
   Puente.pasarCamion();
   //pasar por el puente
   Puente.salirCamion();
end camion;

TASK BODY Puente IS
   int pesoActual=0;
   loop
      select                                                    //estos 3 se aceptan siempre
                  ACCEPT salirAuto do
                        pesoActual:= pesactual-1;
                  end salirauto;
      or
                  ACCEPT salirCamioneta do
                        pesoActual:=pesoactual-2;
                  end salircamioneta;
      or
                  ACCEPT salirCamion do
                        pesoActual-=pesoactual-3;
                  end salirCamion;
      or
                 when (pesoActual + 1 <= 5)  =>            //el Puente banca 5
                        ACCEPT pasarAuto do
                              pesoActual:=pesoActual+1;
                        end pasarAuto;
      or
                  when (pesoActual + 2 <= 5) =>
                        ACCEPT pasarCamioneta do
                              pesoActual:=pesoActual+2;
                        end pasarCamioneta;
      or
                   when (pesoActual + 3 <= 5) =>
                        ACCEPT pasarCamion do
                              pesoActual:=pesoactual + 3;
                        end pasarCamion;
       END SELECT;
   end loop;
end Puente;

Begin

End Puentecito;
```

b. Modifique la solución para que tengan mayor prioridad los camiones que el resto de los vehículos.

```
Procedure Puentecito is

TASK TYPE auto          //no tienen entries
TASK TYPE camioneta
TASK TYPE camion

TASK Puente IS
   ENTRY pasarAuto();
   ENTRY salirAuto();
   ENTRY pasarCamioneta();
   ENTRY salirCamioneta();
   ENTRY pasarCamion();
   ENTRY salirCamion();
end Puente;

autos = array (1..A) of auto;
camionetas = array (1..B) of camioneta;
camiones = array (1..C) of camion;

TASK BODY auto is
   Puente.pasarAuto();
   //pasar por el puente
   Puente.salirAuto();
end auto;

TASK BODY camioneta is
   Puente.pasarCamioneta();
   //pasar por el puente
   Puente.salirCamioneta();
end camioneta;

TASK BODY camion is
   Puente.pasarCamion();
   //pasar por el puente
   Puente.salirCamion();
end camion;

TASK BODY Puente IS
   int pesoActual=0;
   loop
      select                                          //estos 3 se aceptan siempre
                  ACCEPT salirAuto do
                        pesoActual:= pesactual-1;
                  end salirauto;
      or
                  ACCEPT salirCamioneta do
                        pesoActual:=pesoactual-2;
                  end salircamioneta;
      or
                  ACCEPT salirCamion do
                        pesoActual-=pesoactual-3;
                  end salirCamion;
      or
```

```
                when (pasarcamion'count=0) and (pesoActual + 1 <= 5)  =>          //el Puente banca 5
                        ACCEPT pasarAuto do
                                pesoActual:=pesoActual+1;
                        end pasarAuto;
    or
                when (pasarcamion'count=0) and (pesoActual + 2 <= 5) =>
                        ACCEPT pasarCamioneta do
                                pesoActual:=pesoActual+2;
                        end pasarCamioneta;
    or
                when (pesoActual + 3 <= 5) => //tiene prioridad. Los demás solo pasan si no hay camiones
                        ACCEPT pasarCamion do
                                pesoActual:=pesoactual + 3;
                        end pasarCamion;
     END SELECT;
  end loop;
end Puente;

Begin

End Puentecito;
```

2. Se quiere modelar el funcionamiento de un banco, al cual llegan clientes que deben realizar un pago y retirar un comprobante. Existe un único empleado en el banco, el cual atiende de acuerdo con el orden de llegada. Los clientes llegan y si esperan más de 10 minutos se retiran sin realizar el pago.

```
Procedure Banco
  TASK Empleado IS
    Entry pedido (D: in string; R: out string)
  end Empleado;

  TASK TYPE Cliente;

  clientes: ARRAY(1..C) OF Cliente;

  TASK BODY Empleado IS
        loop
                ACCEPT pedido (d: in string; r: out string) do
                        r := ResolverPedido(d);
                end pedido
        end loop;
  end Empleado

  TASK BODY Cliente IS
        Resultado: string;
     begin
        SELECT
                Empleado.pedido(datos, resultado);
        OR DELAY (10 min)
                //se retira
        end SELECT;
  end Cliente;

begin
NULL
```

End banco;

3. Se dispone de un sistema compuesto por 1 central y 2 procesos periféricos, que se comunican continuamente. Se requiere modelar su funcionamiento considerando las siguientes condiciones:
- La central siempre comienza su ejecución tomando una señal del proceso 1; luego toma aleatoriamente señales de cualquiera de los dos indefinidamente. Al recibir una señal de proceso 2, recibe señales del mismo proceso durante 3 minutos.
- Los procesos periféricos envían señales continuamente a la central. La señal del proceso 1 será considerada vieja (se deshecha) si en 2 minutos no fue recibida. Si la señal del proceso 2 no puede ser recibida inmediatamente, entonces espera 1 minuto y vuelve a mandarla (no se deshecha).

```
procedure ej4 is

task Proceso1;                    //no tienen entries
task Proceso2;
task Timer;

task Central is
   entry recibir_p1(s : in Signal);
   entry recibir_p2(s : in Signal);
   entry timeout;
end Central;

task body Proceso1 is
   s : Signal;
begin
        loop
                s := generarSignal();
                select
                        Central.recibir_p1(s);
                or
                        delay 120              //120= 2 minutos de espera
                                null;
                end select;
        end loop;
end Proceso1;

task body Proceso2 is
   s : Signal;
begin
        s := generarSignal(); --genera la primera señal
        loop
                select
                        Central.recibir_p2(s);
                                s := generarSignal();
                else
                        delay(60.0);
                end select;
        end loop;
end Proceso2;

task body Central is
   ok : boolean;
   s_actual : Signal;
begin
```

```
    --primero recibe una señal de p1
    accept recibir_p1(s : in Signal) do
       s_actual := s;
    end recibir_p1;
    --hace algo con la señal
    loop
      select
          accept recibir_p1(s : in Signal) do
             s_actual := s;
          end recibir_p1;
          --hace algo con la señal
      or
        accept recibir_p2(s: in Signal) do
          s_actual := s;
        end recibir_p2;
        --hace algo con la señal
        ok := true;
        Timer.comenzar;
        while ok loop
          select
            when (timeout'count = 0) => accept recibir_p2(s: in Signal) do
               s_actual := s;
            end recibir_p2;
            -- hace algo con la señal
          or
            accept timeout;
            ok := false;
          end select;
        end loop;
      end select;
    end loop;
end Central;


task Timer is
    entry comenzar;
end Timer;

task body Timer is
begin
    loop
      accept comenzar;
      delay(180);
      Central.timeout;
    end loop;
end Timer;

begin
    null;
end ej4;
```

*-

versión 2

```
task body Central is
   recibirSoloDeP2 : boolean;
   s_actual : Signal;
begin
   recibirSoloDeP2 := false;
   --primero recibe una señal de p1
   accept recibir_p1(s : in Signal) do
      s_actual := s;
   end recibir_p1;
   --hace algo con la señal
   loop
     select
        WHEN recibirSoloDeP2=false => accept recibir_p1(s : in Signal) do
           s_actual := s;
         end recibir_p1;
         --hace algo con la señal
     or
       WHEN timeout'count = 0 => accept recibir_p2(s: in Signal) do
         s_actual := s;
       end recibir_p2;
      if (recibirSoloDeP2=false) then recibirSoloDeP2:=true;
           Timer.comenzar; END IF;
       --hace algo con la señal
     OR ACCEPT timeout;
      recibirSoloDeP2 := false;
     end select;
   end loop;
end Central;
```

4. En una clínica existe un médico de guardia que recibe continuamente peticiones de atención de las E enfermeras que trabajan en su piso y de las P personas que llegan a la clínica ser atendidos. Cuando una persona necesita que la atiendan espera a lo sumo 5 minutos a que el médico lo haga, si pasado ese tiempo no lo hace, espera 10 minutos y vuelve a requerir la atención del médico. Si no es atendida tres veces, se enoja y se retira de la clínica.
Cuando una enfermera requiere la atención del médico, si este no lo atiende inmediatamente le hace una nota y se la deja en el consultorio para que esta resuelva su pedido en el momento que pueda (el pedido puede ser que el médico le firme algún papel). Cuando la petición ha sido recibida por el médico o la nota ha sido dejada en el escritorio, continúa trabajando y haciendo más peticiones. El médico atiende los pedidos dándole prioridad a los enfermos que llegan para ser atendidos. Cuando atiende un pedido, recibe la solicitud y la procesa durante un cierto tiempo. Cuando está libre aprovecha a procesar las notas dejadas por las enfermeras.

```
procedure Hospital is
   Task type Enfermera;                    //no tienen entries
   Task type Paciente;

   Task Medico is
      Entry NecesitoAtencion (sintomas: IN string, diagnostico: OUT string);
      Entry PedidoEnfermera (pedido: IN string);
  End medico;


   Task escritorio is
```

```
    Entry DejarNota (nota: IN STRING);
    Entry MandameUnaNota(nota: OUT STRING)
End escritorio;

enfermeras: array(1..N) of Enfermera;
pacientes: array(1..N) of Paciente;

Task Body Enfermera is
   pedido, nota: texto
Begin
        loop
                pedido = generarPedido();
                SELECT
                        Medico.PedidoEnfermera(pedido);
                ELSE
                        nota = generarNota();
                        escritorio.DejarNota(nota);      //para no esperar que el medico la pueda agarrar
        end loop;
End Enfermera;


Task Body Paciente is
   Problema, diagnostico: string;
   esperando: boolean;
   cant_intentos: integer;
Begin
        intentos = 0;     esperando = true,         sintomas = //anotarSintomas ()
        while (esperando) do
                SELECT
                        Medico.NecesitoAtencion(sintomas, diagnostico);
                        esperando = false;
                OR delay 300              //5 minutos
                        Cant_Intentos:=cant_intentos+1;
                        if (cant_intentos < 3) then
                                delay(600);                //espera 10 minutos
                        else
                                esperando = false;
                End Select;
        end;
        //se retira de la clinica
End Paciente;

Task Body Medico is
   sintomas, diagnostico, pedido, nota: string;
Begin
   loop
        SELECT
                Accept NecesitoAtencion (sintomas: IN string, diagnostico: OUT string) do
                        diagnistico = diagnosticar(sintomas);
                end NecesitoAtencion;
        OR
                when (NecesitoAtencion'count = 0) =>            //si no hay pacientes esperando
                        Accept PedidoEnfermera(pedido: in string) do
                                //Procesar pedido
```

```
                    End pedidoenfermera;
        ELSE
              escritorio.MandameUnaNota(nota);     //aca no puede preg (nota'count = 0)
                    if (nota <> " ") then                //solo esa tarea puede preg eso
                          //leernota
              end Select
        end loop;
  End Medico;

TASK BODY escritorio IS
Cola colanotas;
begin
  loop
    SELECT
        ACCEPT DejarNota(nota: in string) do              //recibe de la enfermera
           encolar(colanotas, nota);
        END DejarNota;
    OR ACCEPT MandameUnaNota (nota1: out string) do          //manda al medico
        if (nota'count = 0) then
              nota1:= " "
        else
              desencolar(colanotas, nota1);
    END MandameUnaNota;
  end loop;
end escritorio;

Begin
  null;
End hospital.
```

*_

```
Procedure hospital is
  task medico is
    Entry atenderPersona (solicitud: in string);
    Entry atenderEnfermera (pedido: in string, pedidolisto: out string);
    Entry identificacion (id: out integer );
  end medico;

  task type enfermera is
    Entry recepcionDelMedico(pedidoFirmado: in text)
    Entry identificación (ident: in integer)
  end enfermera;

  task type persona;

  task escritorio is
    Entry procesarNota (notaEnfermera:out text, auxIdEnfermera: out integer)
    Entry atenderEnfermera (pedido: in text, idE:in integer);
  end escritorio;

arrEnfermeras: array (1..E) of enfermera;
arrPersonas: array (1..P) of persona;
```

```
task body medico is
        idE: integer= 0;
        notaEnfermera: string;
        auxIdEnfermera: int;
        pedidolisto: string;
        begin
        loop
            SELECT
              accept atendemePersona(solicitud: in string) do
                Delay (600) #Lo atiendo
              end atendemePersona;

            OR
               when (atendemePersona'count == 0) =>          //si no hay personas esperando
                        accept atenderEnfermera(pedido: in text, idE:in integer, pedidolisto: out string) do
                                pedidolisto:=atenderelpedido(pedido)
            ELSE        ESTO NO SE PUEDE HACER
              select
               escritorio.procesarNota(notaEnfermera,idE);
                        pedidoFirmado = firmar(notaenfermera);      //si escritorio devolvió la nota
                        arrEnfermeras[idE].recepcionDelMedico(pedidoFirmado);  //se hacen estas
                        //se lo manda a la enfermera NO PODRIA GENERAR DEMORA?
              else
               null; //no hay notas o no atendio rapido NO PODRIA IRSE DE ACA SIEMPRE SI HAY
        DEMORA?
        end select;
            end select;
          end loop;
  end medico;

  task body escritorio is
        cola c;
        string pedido_aux;
        int idE_aux;
  begin
    loop
    SELECT
      when not cola.empty() =>          //le atiende un pedido de nota a un medico
       accept procesarNota(notaEnfermera:out text, IdE: out Integer) do
                desencolar (c, (notaEnfermera, idE));
       end procesarNota;
      or
        accept atenderEnfermera(pedido: in string ,idE: in int) do
                pedido_aux := pedido;
                idE_aux:=
        end atenderEnfermera;
        cola.push(pedido_aux,idE_aux);     //hacer esto fuera de accept para desocupar rápido al otro proc
      end select;
    end loop;
  end escritorio;

  task body type enfermera is
         int  idE;
```

```
        string  pedido;
        cola  archivo;
        string pedidolisto;
    begin
        accept identificación (ident: in integer) //recibe id del main
                idE:=ident;
        loop
                select   //si hay algun pedido que espera ser recibido…
                        accept recepcionDelMedico(pedidolisto: in string) do
                                archivo.push(pedidolisto);
                        end recepcionDelMedico;
                else
                        pedido:=generarpedido();
                        select
                                medico.atenderEnfermera(pedido, pedidolisto);
                                encolar(archivo, pedidolisto);
                        else
                                escritorio.atenderEnfermera(pedido,idE);
                        end select;
                end select;
        end loop;
  end enfermera;

  task body type persona is
        sintomas, diagnostico: string;
        begin
          for i: 1..3
                loop
                    select
                            medico.atenderPersona(síntomas, diagnostico);
                            exit;          //break for en ada
                    or delay 300           //espera 5 minutos
                            if i<3 then    //si i==3 se va
                                    delay (600);      //espera 10 minutos
                            end if;
                    end select;
                end loop;
        //se retira de la clinica
  end persona;

begin
        for i:=1 to E loop
                Enfermera(i).identificacion(i);
        end loop;
End clinica;
```

5. En un sistema para acreditar carreras universitarias, hay UN Servidor que atiende pedidos de U Usuarios de a uno a la vez y de acuerdo con el orden en que se hacen los pedidos. Cada usuario trabaja en el documento a presentar, y luego lo envía al servidor; espera la respuesta de este que le indica si está todo bien o hay algún error. Mientras haya algún error, vuelve a trabajar con el documento y a enviarlo al servidor. Cuando el servidor le responde que está todo bien, el usuario se retira. Cuando un usuario envía un pedido espera a lo sumo 2 minutos a que sea recibido por el servidor, pasado ese tiempo espera un minuto y vuelve a intentarlo (usando el mismo documento).

```
procedure sistemaU is
task Servidor is
   entry enviodocumento(doc: in string, correccion: out boolean);
end Servidor;

task type usuario;

usuarios : Array(1..U) of Usuario;

task body Usuario is
   doc: string;
   error: boolean;
begin
            error:=true;            //asume que hay error
            doc:= generardocumento();
            while (error=true) loop
                        select
                                    Servidor.enviodocumento(doc, correccion);
                                    if (correccion=true) then  //si hay correcciones
                                                corregir_documento(docu);
                                    else                              //si no hay correcciones
                                                error:=false;     //para el while
                                    end if;
                        or delay (120)        //espera ser atendido dos minutos
                                    delay(60);//espera un minuto e intenta de nuevo
                        end select;
             end loop;
             //se retira
end Usuario;

task body Servidor is
begin
            loop
                  accept enviodocumento(doc: in string, correccion: out boolean) do
                              corregir(doc, correccion); //si hay correcciones, corrección=true
                  end enviodocumento;
            end loop;
end;

begin
   null;
end sistemaU;
```

6. En una playa hay 5 equipos de 4 personas cada uno (en total son 20 personas donde cada una conoce previamente a que equipo pertenece). Cuando las personas van llegando esperan con los de su equipo hasta que el mismo esté completo (hayan llegado los 4 integrantes), a partir de ese momento el equipo comienza a jugar. El juego consiste en que cada integrante del grupo junta 15 monedas de a una en una playa (las monedas pueden ser de 1, 2 o 5 pesos) y se suman los montos de las 60 monedas conseguidas en el grupo. Al finalizar cada persona debe conocer el grupo que más dinero junto. Nota: maximizar la concurrencia. Suponga que para simular la búsqueda de una moneda por parte de una persona existe una función Moneda() que retorna el valor de la moneda encontrada.

```
Procedure ejercicio 6 is
Task type persona is
```

```
        Entry identificación (ident: in integer)
        Entry seguir()
        Entry resultado (grupo: in, integer)
End persona

Task type equipo is
        Entry identificación (ident: in integer)
        Entry esperar ()
        Entry resultado2 (resul: in, integer);
        Entry enviosuma (monto: in integer);
End equipo

Task type admin is
        Entry final (total: int in, grupoid: int in);
End admin;

Task body persona is
        idP:integer;
        monto_mio: integer;
        ganador: integer;
begin
        monto_mio:=0;
        Accept identificación (ident: in integer) do        //recibe su id
                idP:=ident;
        end identificación;
                case idP is                    //segun tu id, es tu equipo
                        when 1..4 => mi_equipo:=1;
                        when 5..8 => mi_equipo:=2;
                        when 9..12 => mi_equipo:=3;
                        when 13..16 => mi_equipo:=4;
                        when 17..20 => mi_equipo:=5;
        equipo[mi_equipo].esperar ()
        accept seguir (); //espera que esten todos
        for i:=1 to 15 do
                //monedita:=encontrarmoneda()
                monto_mio:=monto_mio+moneda(monedita)
        end;
        equipo[mi_equipo].enviomisuma(monto_mio)
        accept resultado (grupo: in, integer) do
                ganador:=grupo;
        end resultado
        //print (“El grupo ganador es”, ganador)
End persona;

Task body equipo
        Total, base, i: integer;
Begin
        Total:=0;
        Accept identificación (ident: in integer) do        //recibe su id
                idG:=ident
        end identificación;
        For i:=1 to 4 loop          //los 4 le avisan que llegaron
                Accept esperar() do
                End esperar
```

```
        End loop
        base:= idG*4-3 //base para un for. Ej: id4= el primero de tu grupo (tu base) = 4*4-3=13
        For i:=base to (base+4) loop
                Persona(i).seguir();        //le avisas a todos que sigan
        End loop;
        For i:=1 to 4 loop          //los 4 le envian sus montos
                Accept enviosuma (monto: in integer) do
                                Total:=total+monto;
                End enviosuma
        End loop
        Admin.final (total, idG)            //manda puntaje y su id
        Accept resultado2 (resul: in, integer) do
                Grupowin:=resul;
        End resultado
        For i:=base to (base+4) loop
                Persona(i).resultado(grupowin);//le avisas a todos que sigan
        End loop;
End equipo;

Task body admin is
        Max: integer
        Grupoganador:integer;
Begin
        Max:=-1;
        For i:=1 to 4
                Accept final (total: int in, grupoid: int in) do
                        If (total>max) then
                                Max:=total;
                                Grupoganador:=grupoid;
                End final;
        End loop;
        For i:=1 to 5 loop
                Equipo(i).resultado2(grupoganador);
        End loop;
End admin;

begin
        for i:=1 to 20 loop
                persona(i).identificacion(i);
        end loop;
        for i:=1 to 5 loop
                equipo(i).identificacion(i);
        end loop;

end ejercicio6;
```

*-

Otra opción con un solo proceso:

```
Task body persona is
        idP:integer;
        monto_mio: integer;
        ganador: integer;
begin
        monto_mio:=0;
```

```
Accept identificación (ident: in integer) do        //recibe su id
        idP:=ident;
end identificación;
        case idP is                 //segun tu id, es tu equipo
                when 1..4 => mi_equipo:=1;
                when 5..8 => mi_equipo:=2;
                when 9..12 => mi_equipo:=3;
                when 13..16 => mi_equipo:=4;
                when 17..20 => mi_equipo:=5;
if (idp=1 or idp=5 or idp=9 or idp=13 or idp=17) then     //si son los "capitanes"
        for i:=1 to 3 loop          //acepta los avisos de sus compas
                accept esperar()
                end esperar
        end loop
        for i:=idP+1 to idP+3 loop          //le manda a sus compas
                persona(i).seguir()
        end loop
else                                        //si no sos un capitan, le avisas a tu capitán que estas
        case idP is                 //segun tu id, es tu capitan
                when 2..4 => persona [1].esperar()
                when 4..8 => persona[5].esperar ()
                when 10..12 => persona[9].esperar ()
                when 14..16 => persona[13].esperar ()
                when 18..20 => persona[17].esperar ()
        accept seguir () //el capitan le avisa que estan todos
        end seguir;
for i:=1 to 15 do
        //monedita:=encontrarmoneda()
        monto_mio:=monto_mio+moneda(monedita)
end;
//PROCESA RESULTADOS
total:=monto_mio;
if (idp=1 or idp=5 or idp=9 or idp=13 or idp=17) then     //si son los "capitanes"
        for i:=1 to 3 loop          //acepta los avisos de sus compas
                Accept enviosuma (monto: in integer) do
                        Total:=total+monto;
                End enviosuma
        end loop
if (idp=5 or idp=9 or idp=13 or idp=17) then //si no es quien tiene que calcular, e manda a ese
        idG=(idp+3)/4
        persona(1).enviototal(total, idG);
else                        //si sos el proceso 1 que tiene que calcular
        Max:=-1;
        For i:=1 to 4
                Accept enviototal (total: int in, grupoid: int in) do
                        If (total>max) then
                                Max:=total;
                                Grupoganador:=grupoid;
                End final;
        End loop;
        for i:=2 to 20 loop         //le manda a sus compas
                persona(i).resultado(grupoganador)
        end loop
else                                        //si no sos un capitan, le avisas a tu capitán que estas
```

```
                case idP is                    //segun tu id, es tu capitan
                        when 1..4 => persona[1].enviomisuma(monto_mio)
                        when 5..8 => persona[5].enviomisuma(monto_mio)
                        when 9..12 => persona[9].enviomisuma(monto_mio)
                        when 13..16 => persona[13].enviomisuma(monto_mio)
                        when 17..20 => persona[17].enviomisuma(monto_mio)
        if (idP!= 1) then            //si no sos el que calculo el resultado, lo tenes que recibir
                accept resultado (grupo: in, integer) do //esperas que te den el ganador
                        ganador:=grupo;
                end resultado
        end;
        //print ("El grupo ganador es", ganador)
End persona;
```

7. Hay un sistema de reconocimiento de huellas dactilares de la policía que tiene 8 Servidores para realizar el reconocimiento, cada uno de ellos trabajando con una Base de Datos propia; a su vez hay un Especialista que utiliza indefinidamente. El sistema funciona de la siguiente manera: el Especialista toma una imagen de una huella (TEST) y se la envía a los servidores para que cada uno de ellos le devuelva el código y el valor de similitud de la huella que más se asemeja a TEST en su BD; al final del procesamiento, el especialista debe conocer el código de la huella con mayor valor de similitud entre las devueltas por los 8 servidores. Cuando ha terminado de procesar una huella comienza nuevamente todo el ciclo.
Nota: suponga que existe una función Buscar(test, código, valor) que utiliza cada Servidor donde recibe como parámetro de entrada la huella test, y devuelve como parámetros de salida el código y el valor de similitud de la huella más parecida a test en la BD correspondiente. Maximizar la concurrencia y no generar demora innecesaria.

```
Procedure sistema is
Task type servidor
        Entry envioimagen (test: in imagen)
End servidor;

Task especialista
        Entry resultado (código: in integer, valor: in integer)
End especialista;

Servidores: array (1..8) of servidor;

Task body servidor
        Aux_test:image;
        Código, valor: integer;
Begin
        loop
                Accept envioimagen (test: in imagen)
                        Aux_test:=test;
                End envioimagen;
                Buscar (test, codigo, valor);
                Especialista.resultado(codigo, valor);
        End loop;
End servidor;

Task body especialista
        Test: image;
        Aux_valor, Simil_max: integer;
        Aux_codigo, Código_max: integer;
```

```
Begin
        loop
                test=tomar_imagen_de_huella()
                For i:=1 to 8 loop
                        Servidor(i).envioimagen(test)
                End loop
                Simil_max:=-1;
                for i:=1 to 8 loop
                        accept resultado (código: in integer, valor: in integer) do
                                aux_valor=valor;
                                aux_codigo:=código;
                        End resultado;
                        If (aux_valor>simil_max) then
                                Simil_max:=aux_valor;
                                Código_max:=aux_codigo;
                        End;
                End loop;
                Print ("Código de la huella con mayor valor de similitud: ", código_max);
        End loop
End especialista;

Begin
        Null
End sistema;
```

8. Una empresa de limpieza se encarga de recolectar residuos en una ciudad por medio de 3 camiones. Hay P personas que hacen continuos reclamos hasta que uno de los camiones pase por su casa. Cada persona hace un reclamo, espera a lo sumo 15 minutos a que llegue un camión y si no vuelve a hacer el reclamo y a esperar a lo sumo 15 minutos a que llegue un camión y así sucesivamente hasta que el camión llegue y recolecte los residuos; en ese momento deja de hacer reclamos y se va. Cuando un camión está libre la empresa lo envía a la casa de la persona que más reclamos ha hecho sin ser atendido. Nota: maximizar la concurrencia.

```
Procedure ejercicio8 is
TASK TYPE camion        //no tienen entries
TASK TYPE persona
        Entry identificación (ident: in integer)
        Entry llegacaminion ()
End persona;
Task admin
        ENTRY reclamo (idP: in integer);
        ENTRY camión_desocupado (p_atender: out integer)
End admin;
camioncitos = array (1..3) of camion;
personas = array (1..P) of persona;

Task body camión
Begin
        loop
                admin.camion_desocupado(p_atender);//le avisa al admi que esta libre y este le da una
        persona
                persona(p_atender).llegacamion();
                //termino de atender a esa persona
        End loop
```

```
task body admin
        reclamos: array [1..P]
        camiones_disp: cola;

begin
        for i:=1 to P loop                    //inicializar vector en cero
                reclamos [i]:=0
        end loop
        loop
                select
                        acept reclamo (idP: in integer) do          //recibe reclamo de persona
                                reclamos [idP]:= reclamos [idP]+1;
                        end reclamo
                or
                        when (camión_desocupado'count >0) => //si hay camiones disponibles
                                prox_atender:= reclamos.max().pos();   //el que mas reclamos hizo
                                reclamos [prox_atender]:=0; //para que no le mande de nuevo
                                accept camión_desocupado (p_atender: out integer) do
                                        p_atender:=prox_atender;
                                end camión_desocupado
        end loop

Task body personas
        Atendido: boolean;
        idP: integer;
Begin
        Accept identificación (ident: in integer) do        //recibe su id
                idP:=ident;
        end identificación;
        atendido:=false;
        while (atendido=false) loop                //mientras nadie lo atiende
                admin.reclamo (idP)
                select
                        accept llegacaminion () do        //si el camion llega
                                //el camion recolecta residuos
                        end llegacamion
                        atendido:=true
                or delay (900)   //si no espera 15 minutos que llegue
                        null
                end select
        end loop
        //se retira

begin
        for i:=1 to P loop
                persona(i).identificacion(i);
        end loop;
end ejercicio8;
```