Practica 5

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 resolverlo.
a. Realice la solución suponiendo que todos los vehículos tienen la misma prioridad.

```
procedure Puente1a is

    Task Puente is
        entry pasaAuto();
        entry pasaCamioneta();
        entry pasaCamion();
        entry salir(peso: IN integer);
    END Puente;

    Task body Puente is
      pesoActual: integer = 0;
  begin
      loop
          select
                  when (pesoActual < 5) =>
                          accept entrarAuto() do
```

```
                    pesoActual:= pesoActual + 1;
                end entrarAuto;
            or
            when (pesoActual < 4) =>
                accept entrarCamioneta() do
                    pesoActual:= pesoActual + 2;
                end entrarCamioneta;
            or
            when (pesoActual < 3) =>
                accept entrarCamion() do
                    pesoActual:= pesoActual + 3;
                end entrarCamion;
            or
            accept salir(peso: IN integer) do
                pesoActual:= pesoActual - peso;
            end salir;
        end select;
    end loop;
  end Puente;


Task type Vehiculo
  arrAutos: array(1..A) of Vehiculo;
  arrCamioneta: array(1..B) of Vehiculo;
  arrCamion: array(1..C) of Camion;

  Task body Vehiculo is
      tipo: string;
  begin
      if (tipo = "Auto") then
          Puente.entrarAuto();
          Puente.salir(1);
      elsif (tipo = "Camioneta") then
          Puente.entrarCamioneta();
          Puente.salir(2);
      else
```

```
            Puente.entrarCamion();
            Puente.salir(3);
        end if;
    end Vehiculo;


begin
    null;
end Puente1A;
```

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

```
Procedure Puente1B is

    Task Puente is
        entry entrarAuto();
        entry entrarCamioneta();
        entry entrarCamion();
        entry salir(peso: IN integer);
    end Puente;

    Task body Puente is
        pesoActual: integer = 0;
    begin
```

```
        loop
            select
                when (pesoActual < 5) and (entrarCamion'count =
                    accept entrarAuto() do
                        pesoActual:= pesoActual + 1;
                    end entrarAuto;
                or
                when (pesoActual < 4) and (entrarCamion'count =
                    accept entrarCamioneta() do
                        pesoActual:= pesoActual + 2;
                    end entrarCamioneta;
                or
                when (pesoActual < 3) =>
                    accept entrarCamion() do
                        pesoActual:= pesoActual + 3;
                    end entrarCamion;
                or
                accept salir(peso: IN integer) do
                    pesoActual:= pesoActual - peso;
                end salir;
            end select;
        end loop;
    end Puente;

    Task type Vehiculo
    arrAutos: array(1..A) of Vehiculo;
    arrCamioneta: array(1..B) of Vehiculo;
    arrCamion: array(1..C) of Camion;

    Task body Vehiculo is
        tipo: string;
    begin
        if (tipo = "Auto") then
            Puente.entrarAuto();
            Puente.salir(1);
        elsif (tipo = "Camion") then
```

```
            Puente.entrarCamioneta();
            Puente.salir(2);
        else
            Puente.entrarCamion();
            Puente.salir(3);
        end if;
    end Vehiculo;

begin
    null;
end Puente1B;
```

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.
a) Implemente una solución donde los clientes llegan y se retiran sólo después de haber sido
atendidos.

```
Procedure Banco1 is

    Task empleado is
        Entry Pedido (D: IN texto; R: OUT texto);
    End empleado;

    Task type cliente;

    arrClientes: array (1..N) of Cliente;
```

```
    Task Body cliente is
        comprobante: texto;
    Begin
        Empleado.Pedido (“PAGO”, comprobante);
    End cliente;

    Task Body empleado is
    Begin
        loop
            accept Pedido (D: IN texto; R: OUT texto) do
                R := resolverPago(D);
            end Pedido;
        end loop;
    End empleado;

Begin
    null;
End Banco1;
```

b) Implemente una solución donde los clientes se retiran si esperan más de 10 minutos para
realizar el pago.

```
Procedure Banco2 is

    Task empleado is
        Entry Pedido (D: IN texto; R: OUT texto);
    End empleado;

    Task type cliente;
    arrClientes: array (1..N) of Cliente;
```

```
    Task Body cliente is
        Resultado: texto;
    Begin
        SELECT
            Empleado.Pedido (“datos”, Resultado);
        OR DELAY 600.0
            NULL;
        END SELECT;
    End cliente;

    Task Body empleado is
    Begin
        loop
            accept Pedido (D: IN texto; R: OUT texto) do
                R := resolverPedido(D);
            end Pedido;
        end loop;
    End empleado;

Begin
    null;
End Banco2;
```

c) Implemente una solución donde los clientes se retiran si no son atendidos inmediatamente.

```
Procedure Banco2 is

    Task empleado is
```

```
        Entry Pedido (D: IN texto; R: OUT texto);
    End empleado;

    Task type cliente;
    arrClientes: array (1..N) of Cliente;

    Task Body cliente is
        Resultado: texto;
    Begin
        SELECT
            Empleado.Pedido (“datos”, Resultado);
        ELSE
            NULL;
        END SELECT;
    End cliente;

    Task Body empleado is
    Begin
        loop
            accept Pedido (D: IN texto; R: OUT texto) do
                R := resolverPedido(D);
            end Pedido;
        end loop;
    End empleado;

Begin
    null;
End Banco2;
```

d) Implemente una solución donde los clientes esperan a lo sumo 10 minutos para ser
atendidos. Si pasado ese lapso no fueron atendidos, entonces solicitan atención

una vez más
y se retiran si no son atendidos inmediatamente.

```
Procedure Banco2 is

    Task empleado is
        Entry Pedido (D: IN texto; R: OUT texto);
    End empleado;

    Task type cliente;
    arrClientes: array (1..N) of Cliente;

    Task Body cliente is
        Resultado: texto;
    Begin
        SELECT
            Empleado.Pedido (“datos”, Resultado);
        OR DELAY 600.0
            SELECT
                Empleado.Pedido("datos",Resultado);
            ELSE
                null;
            END SELECT;
        END SELECT;
    End cliente;

    Task Body empleado is
    Begin
        loop
            accept Pedido (D: IN texto; R: OUT texto) do
                R := resolverPedido(D);
            end Pedido;
        end loop;
    End empleado;
```

```
Begin
    null;
End Banco2;
```

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 Sistema IS
    TASK Central IS
        ENTRY RecibirSenialPerifericoUno (senial: IN Senial);
        ENTRY RecibirSenialPerifericoDos (senial: IN Senial);
        ENTRY RecibirAviso;
    END Central;
```

```
TASK PerifericoUno
TASK BODY PerifericoUno IS
    senial: Senial;
BEGIN
    LOOP
        senial = GenerarSenial();
        SELECT
            Central.RecibirSenialPerifericoUno (senial);
        OR DELAY 120
            null;
        END SELECT;
    END LOOP;
END PerifericoUno;

TASK PerifericoDos
TASK BODY PerifericoDos IS
    senial: Senial;
BEGIN
    senial = GenerarSenial();
    LOOP
        SELECT
            Central.RecibirSenialPerifericoDos (senial);
            senial = GenerarSenial();
        ELSE
            DELAY 60;
        END SELECT;
    END LOOP;
END PerifericoDos;

TASK Timer IS
    ENTRY InicioTimer;
TASK BODY Timer IS
BEGIN
    ACCEPT InicioTimer;
    DELAY(180);
```

```
        Central.RecibirAviso;
    END Timer;

    TASK BODY Central IS
        continuar: Boolean;
    BEGIN
            continuar := False
        ACCEPT RecibirSenialPerifericoUno (senial: IN Senial); /
        LOOP
            SELECT
                WHEN(continuar = false) ->  //el segundo puede s
                                        ACCEPT RecibirSenialPeri
            OR
                WHEN(contiunar = false) -> //el segundo si es de
                ACCEPT RecibirSenialPerifericoDos (senial: IN Se
                continuar = True;
                Timer.InicioTimer;
              OR
                  WHEN (RecibirAviso'COUNT = 0) && (continuar) -
                                          ACCEPT RecibirSenialPe
                  OR
                ACCEPT RecibirAviso; //recibe aviso del timer qu
                    continuar = False;
            END SELECT;
         END LOOP;
    END Central;

BEGIN
    null;
END Sistema;
```

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 eje4 is

    Task medico is
        entry recibeAtencionPaciente(solicitud: IN text, resoluc
        entry recibeAtencionEnfermera()

    Task administrador is
        entry recibeNota(nota: IN text)
        entry recibePedidoMedico(nota: OUT text)

    Task type paciente is

    Task body paciente is
        integer intentos = 0;
        text solicitud,resolucion;
```

```
        boolean atendido = false
        SELECT
            medico.recibeAtencionPaciente(solicitud, resolucion)
        OR DELAY 300
            DELAY 600
            while (intentos < 3) and (not atendido) loop
                SELECT
                    medico.recibeAtencionPaciente(solicitud,reso
                    atendido = true
                OR DELAY 600
                    intentos++
            end loop
        END SELECT
        //se retira
    end paciente



    Task type enfermera is

    Task body enfermera is
        nota text;
        loop
            SELECT
                medico.recibeAtencionEnfermera
            ELSE
                nota = generarNota()
                administrador.recibeNota(nota)
            END SELECT
        end loop
    end enfermera


    Task body administrador is
        cola notas
        text nota
```

```
        loop
            SELECT
                accept recibeNota(nota: in text)
                    notas.push(nota)
            OR
                WHEN (!notas.vacio()) -> accept recibePedidoMedi



            END SELECT
        end loop
    end administrador


    Task body medico is
        nota text;
        loop
            SELECT
                accept recibeAtencionPaciente(solicitud: in text
                    resolucion = resolverSolicitud(solicitud)
                OR
                    WHEN (recibeAtencionPaciente'count = 0) -> a
            END SELECT
            administrador.recibePedidoMedico(nota)
        end loop
```

4bis

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 4bis is

    Task servidor is
        entry recibeDoc(doc: in text; resultado: out boolean)

    Task type cliente is

    Task body servidor is
    begin
        loop
            accept recibeDoc(doc: in text; resultado: out boolea
                resultado = analizarDoc(doc)
        end loop
    end servidor

    Task body cliente is
        sigue = true
        text doc
    begin
        while (sigue) loop
            doc =//trabajarDoc()
            SELECT
                servidor.recibeDoc(doc,sigue)
            OR DELAY 120
```

```
                DELAY 60
            END SELECT
        end loop
    end

begin
    null
end
```

5- 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 playa is

    Task type Equipo is
        entry recibirMonedas(monedas: in integer, id: in integer
        entry barrera(idPersona: in integer)
        entry asignarIdEquipo(id: in integer)
        entry recibirGanador(id: in integer)
        entry darGanador(id: out integer)
    end Equipo

    Task type persona is
        entry iniciar()
        entry asignarIdPersona(id: in integer)
    end persona

    Task admin is
        entry recibirTotales(cant: in integer; id: in integer)
    end admin

    ArrE: array (1..5) of Equipo;
    ArrP array (1..20) of Persona;



    Task body persona is
        integer idEquipo, cantMonedas, idPersona, idGanador;

        accept asignarIdPersona(id: in integer)
            idPersona = id
        equipo[idEquipo].barrera(idPersona)
        accept iniciar()
```

```
        for i := 1 to 15 loop
            cantMonedas += Moneda()
        end loop
        equipo[idEquipo].recibirMonedas(cantMonedas)
        equipo[idEquipo].darGanador(idGanador)
    end persona


    Task body equipo is
        idEquipo, idGanador integer;
        integrantes cola;
        monedasTotales integer = 0;

        accept asignarId(id: in integer)
            idEquipo = id;
        for i in 1..4 loop
            accept barrera(idPersona: in integer)
                cola.push(id)
        end loop
        for i in 1..4 loop
            persona[cola.pop()].iniciar()
        end loop

        for i in 1..4 loop
            accept recibirMonedas(monedas: in integer)
                monedasTotales+= monedas
        end loop

        admin.enviarTotal(monedasTotales,idEquipo)

        accept recibirGanador(id)
            idGanador = id

        for i in 1..4 loop
            accept darGanador(id: out integer)
                id = idGanador
```

```
        end loop
    end equipo

    Task body admin is
        cola equipos;
        idGanador integer;

        for i in 1..5 loop
            accept enviarTotal(monedas: in integer; id: in integ
                equipos.push(monedas,id)
        end loop

        idGanador = calcularMaximo(equipos)//calcula el maximo y

        for i in 1..5 loop
            equipo[i].recibirGanador(idGanador)
        end loop
    end admin

BEGIN
    for i in 1..5 loop
        ArrE[i].asignarIdEquipo(i)
    end loop
    for i in 1..20 loop
        ArrP[i].asignarIdPersona(i)
    end loop
END
```

6- Se debe calcular el valor promedio de un vector de 1 millón de números enteros que se
encuentra distribuido entre 10 procesos Worker (es decir, cada Worker tiene un vector de

100 mil números). Para ello, existe un Coordinador que determina el momento en que se
debe realizar el cálculo de este promedio y que, además, se queda con el resultado. Nota:
maximizar la concurrencia; este cálculo se hace una sola vez.

```
procedure eje6 is

    Task type worker

    Task admin is
        entry meToca()
        entry Resultado(nro: in integer)
    end admin

    arrW = array (1..10) of worker

    Task body worker is
        vec: array (1..100000) of integer := //cargarVector();
        promedio real = 0;
        admin.meToca()
        for i in 1.100000 loop
            promedio += vec[i]
        end loop
        promedio := promedio/100000
        admin.Resultado(promedio)
    end worker

    Task body admin
        promedio: real
    begin
        promedio = 0
        for i in 1..20 loop
            SELECT
```

```
                accept meToca()
            OR
                accept Resultado(numero: in integer)do
                    promedio += numero
            end select
        end loop
        promedio := promedio/10
    end admin
Begin
    null
end
```

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 ej7 is

    Task type servidor

    arrS = array (1..8] of servidor


    Task especialista is
        entry tomarHuella(huella: out TEST);
        entry recibirValor(valor: in real; codigo: in real)
    end especialista


    Task body servidor is
    begin
        huella test;
        real codigo,valor;
        loop
            especialista.tomarHuella(huella);
            Buscar(huella,codigo,valor)
            especialista.recibirValor(valor,codigo)
        end loop
    end servidor


    Task body especialista is
    begin
        huella test;
        cola valores;
        codigoMax real;
        loop
```

```
            huella := //tomarImagen()
            for i in 1..8 loop
                accept tomarHuella(huellaSV: OUT Test)do
                    huellaSV := huella
            end loop
            for i in 1..8 loop
                accept recibirValor(valor: in real, codigo in re
                    valores.push(valor,codigo)
            end loop
            codigoMax := calcularMax(valores)//Retorna el codigo
        end loop
    end especialista

    begin
        null
    end
```

8- Una empresa de limpieza se encarga de recolectar residuos en una ciudad por medio de 3
camiones. Hay P personas que hacen reclamos continuamente hasta que uno de los camiones pase por su casa. Cada persona hace un reclamo y espera a lo sumo 15 minutos a
que llegue un camión; si no pasa, 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. Sólo cuando un camión llega, es cuando deja de hacer reclamos y se retira.

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 ej8 is

    Task type camion is

    camiones: array(1..3) of camion

    Task type persona is
        entry asignarmeId(id: in integer)
        entry atender()
    end persona

    Task admin is
        entry pedirCamion(id: out integer)
        entry hacerReclamo(id : in integer)
    end admin


    Task body camion is
        idPersona integer;
    begin
            loop
        admin.pedirCamion(idPersona)
        personas[idPersona].atender()
    end

    Task body persona is
        id integer;
        boolean atendido;
    begin
        atendido := false
```

```
        accept asignarmeId(idPersona: in integer)do
            id := idPersona
        end asignameId
        while not(atendido) loop
            admin.hacerReclamo(id)
            SELECT
                accept atender()
                atendido := true
            OR DELAY 900
                null
            END SELECT
        end loop
    end persona


    Task body admin is
        contador: array (1..P) of integer;
        hayPersona boolean;
        idPersona intger;
    begin
        hayPersona := false
        contador = ([P] 0)
        loop
            select
                accept hacerReclamo(id: in integer)do
                    idPersona := id
                end hacerReclamo
                contador[idPersona]++
                hayPersona := true
            or
                when(hayPersona) ->
                    accept pedirCamion(id: out integer)do
                        id := maximo(contador) // retorna el ind
                        contador[id] := 0
                    end pedirCamion
            end select
```

```
        end loop
    end

    begin
        for i:= 1 to P
            personas[i].asignarmeId(i)
    end
```