HOJA: 1

FECHA

Concurrente Práctica 5

1)a)
```
PROCEDURE Puente IS
   TASK Paso IS
      ENTRY AccesoA;
      ENTRY AccesoB;
      ENTRY AccesoC;
      ENTRY Salir( p : IN int);
   END Paso;

   TASK TYPE Auto;
   TASK TYPE Camioneta;
   TASK TYPE Camion;

   arrAutos: array (1..A) of Auto;
   arrCamionetas: array (1..B) of Camioneta;
   arrCamiones: array (1..C) of Camion;

   TASK BODY Paso IS
      peso: int;
   BEGIN
      peso = 5;
      LOOP
         SELECT
            when (peso ≥ 1) → ACCEPT AccesoA DO
                  peso = peso - 1;
                              END AccesoA;
         OR
            when (peso ≥ 2) → ACCEPT AccesoB DO
                  peso = peso - 2;
                              END AccesoB;
         :
         :
```

```
  :
  OR
      When (peso >= 3) → ACCEPT AccesoC DO
          peso = peso - 3;
                          END AccesoC;
    OR
        ACCEPT Salir ( p : IN int) DO
              peso = peso + p;
        END Salir;
    END SELECT;
  END LOOP;
END Paso;

TASK BODY Auto IS
    p: int;
BEGIN
    p=1;
    Paso.AccesoA;
    Paso.Salir(p);
END Auto;

TASK BODY Camioneta IS
    p: int;
BEGIN
    p=2;
    paso.AccesoB;
    paso.Salir (p);
END camioneta;

TASK Body Camion is
    P: int;
BEGIN
    P=3;
    Paso.AccesoC;
    Paso. Salir(P);
END Camion;
```

```
BEGIN
    null;
END Puente;
```

Concurrente | Práctica 5 | HOJA: 2 | FECHA

1) b) Idem inciso A menos el TASK BODY Paso:

```
  ⋮
TASK BODY Paso is
    peso = int;
BEGIN
    peso = 5;
    LOOP
      SELECT
        when ((Acceso C' COUNT = 0) or (peso < 3)) AND
             (peso ≥ 1)) → ACCEPT AccesoA DO
                                 peso = peso - 1;
                              END AccesoA;
      OR
        when ((Acceso C' COUNT = 0) or (peso < 3)) AND
             (peso ≥ 2)) → ACCEPT Acceso B DO
                                 peso = peso - 2;
                              END AccesoB;
      OR
        when (peso ≥ 3) → ACCEPT Acceso C DO
                                 peso = peso - 3;
                              END Acceso C;

      OR

        ACCEPT Salir (p: IN int) DO
              peso = peso + p;
        END Salir;
      END SELECT;
    END LOOP;
END Paso;
```

NOTA

```
2)a) PROCEDURE Banco IS
       TASK Empleado IS
           ENTRY Acceso (pago: IN txt, comp: OUT txt);
       END Empleado;

       TASK TYPE Cliente;

     arrCliente: array (1..c) of Cliente;

     TASK BODY Cliente IS
         comp, pago: txt;
     BEGIN
         pago = ...;
         Empleado.Acceso (pago, comp);
     END Cliente;

     TASK BODY Empleado IS
     BEGIN
       LOOP
         ACCEPT Acceso (pago: IN txt, comp: OUT txt) DO
                comp = Cobrar (pago);
         END Acceso;
       END LOOP;
     END Empleado;

   BEGIN
      null;
   END Banco;
```

NOTA

HOJA 3
FECHA

Concurrente Práctica 5

b) Idem inciso a menos el TASK BODY Cliente.

```
:

TASK BODY Cliente IS
 comp, pago: txt;
BEGIN
    pago = ...;
    SELECT
        Empleado.Acceso (pago, comp);
    OR DELAY 600;
        NULL;
    END SELECT;
END cliente;
 :
```

c) Idem inciso a menos el TASK BODY Cliente

```
 :

TASK BODY Cliente IS
   comp, pago: txt;
BEGIN
    pago = ...;
    SELECT
        Empleado. Acceso (pago, comp);
    ELSE
        NULL;
    END SELECT;
END cliente;
   :
```

NOTA

d) Idem inciso a menos TASK BODY Cliente

```
:
TASK BODY Cliente IS
   comp, pago: txt;
 BEGIN
    pago= ...;
    SELECT
       Empleado.Access (pago, comp);
    OR DELAY 600;
       SELECT
          Empleado.Access (pago, comp);
       ELSE
          NULL;
    END SELECT;
 END Cliente;
 :
 :
```

3)
```
PROCEDURE Sistema IS
   TASK Timer IS
      ENTRY Iniciar;
   END Timer;
   TASK Proceso1;
   TASK Proceso2;
   TASK Central IS
      ENTRY Recibir1 (signal: IN txt);
      ENTRY Recibir2 (signal: IN txt);
      ENTRY FinTimer;
   END Central;
```

HOJA 4

FECHA

Concurrente Práctica 5

```
TASK BODY Timer IS
 BEGIN
    ACCEPT Iniciar;
    DELAY (180);
     central.FinTimer;
 END Timer;

TASK BODY Proceso1 IS
   Signal: txt;
 BEGIN
    Loop
      signal = ...;
      SELECT
           central.Recibir1 (signal);
      or DELAY(120);
         NULL;
     END SELECT;
    END LOOP;
 END Proceso1;

TASK BODY Proceso2 IS
   Signal: txt;
 BEGIN
    signal = ...;
    Loop
      SELECT
          Central.Recibir2 (signal);
           signal = ...;
      ELSE
          DELAY (60);
     END SELECT;
   END LOOP;
 END Proceso2;
```

NOTA

```
TASK BODY Central IS
    ok: boolean;
BEGIN
    ACCEPT Recibir1(signal: IN txt);
    Loop
        SELECT
            ACCEPT Recibir1(signal: IN txt);
        OR
            ACCEPT Recibir2(signal: IN txt);
            ok := true;
            Timer.Iniciar;
            Loop (ok)
                SELECT
                    when (FinTimer'COUNT = 0) →
                        ACCEPT Recibir2(Signal: IN txt);
                OR
                    ACCEPT FinTimer;
                    ok := false;
                END SELECT;
            END LOOP;
        END SELECT;
    END LOOP;
END Central;

BEGIN
    null;
END Sistema;
```

HOJA: 5

FECHA

Concurrente Práctica 5

```
4) PROCEDURE Promedio IS
    TASK Coordinador IS
        ENTRY Empezar;
        ENTRY Recibir (cant: IN int);
    END Coordinador;

    TASK TYPE Worker;

    arrWorker: array (1..10) of Worker;

   TASK BODY Coordinador IS
      total: int;   promedio: double;
    BEGIN
       total = 0;
     For i IN  1..20 LOOP
        SELECT
            ACCEPT Empezar;
        OR
            ACCEPT Recibir (cant: IN int) DO
                total= total + cant;
            END Recibir;
        END SELECT;
     END LOOP;
   END Coordinador;
```

NOTA

```
TASK BODY Worker IS
        arrNumeros: array (1..100000) of int;        // viene cargado
        Suma: int;
    BEGIN
        Suma = 0;
        Coordinador. Empezar;
        FOR i IN 1..100000 LOOP
            Suma = Suma + arrNumeros(i);
        END LOOP;
        Coordinador. Recibir (suma);
    END Worker;

BEGIN
    null;
END Promedio;
```

NOTA

HOJA 6

FECHA

Concurrente Práctica 5

```
5) PROCEDURE Clinica IS
     TASK TYPE Persona;
     TASK TYPE Enfermera;
     TASK Consultorio IS
        ENTRY PedidoC (pedido: IN txt);
        ENTRY Nota (nota: OUT txt);
     END Consultorio;
     TASK Medico IS
        ENTRY PedidoP;
        ENTRY PedidoE (pedido: IN txt);
     END Medico;


     arrPersona: array (1..P) of Persona;
     arrEnfermera: array (1..E) of Enfermera;

     TASK BODY Persona IS
     BEGIN
       FOR i IN 1..3 LOOP
          SELECT
              Medico.PedidoP;
          OR DELAY (300);
              DELAY (600);
          END SELECT;
       END LOOP;
     END Persona;
```

NOTA

```
TASK BODY Enfermera IS
    pedido, nota: txt;
BEGIN
    LOOP
        SELECT
                Medico.PedidoE(pedido);
        ELSE
                nota = generaNota(pedido);
                Consultorio.PedidoC(nota);
        END SELECT;
    END LOOP;
END Enfermera;

TASK BODY Medico IS
    nota: txt;
BEGIN
    LOOP
        SELECT
            ACCEPT PedidoP;
        OR
            when(PedidoP'COUNT = 0)
                ACCEPT PedidoE(pedido : IN txt);
                        //procesarPedido;
                ELSE
                    SELECT Consultorio.Nota(nota);
                            // procesa Nota
                    ELSE
                        null;
                    END SELECT
        END SELECT;
    END LOOP;
END Medico;
```

NOTA

Concurrente Práctica 5

```
TASK BODY CONSULTORIO IS
 buffer: cola;
BEGIN
  LOOP
    SELECT
        ACCEPT PedidoC (nota: IN txt) DO
             push (buffer(nota));
        END PedidoC;
    OR
      ACCEPT Nota (nota: OUT txt) DO
          IF (NOT EMPTY (Buffer)) DO
              pop (buffer(nota));
          ELSE
              nota = " ";
          END IF
      END Nota;
    END SELECT;
  END LOOP;
END Consultorio;

BEGIN
  null;
END Clinica;
```

```
6) PROCEDURE Acreditacion IS
    TASK TYPE Usuario;
    TASK Servidor IS
        ENTRY Revision (doc: IN/OUT txt; error:OUT bool);
    END Servidor;

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

    TASK BODY Usuario IS
        doc: txt;
        ok: boolean;
    BEGIN
        OK = false;
        doc = generarDocumento();
        while (not OK) LOOP
            SELECT Servidor.Revision (doc, ok) DO
                    if (NOT OK) THEN
                        doc = arreglarDocumento();
                    END IF;
                END Revision;
            or  DELAY 120
                DELAY 60
            END SELECT;
        END LOOP;
    END Usuario;
```

Concurrente Práctica 5

HOJA: 8

FECHA

```
TASK BODY Servidor IS

BEGIN
  LOOP
      ACCEPT Revision (doc: IN/OUT txt, ok: OUT bool) DO
             ok = evaluar (doc);
      END REVISION;
  END LOOP;
 END Servidor;

BEGIN
 null;
END Acreditacion;
```

```
7) PROCEDURE JuegoPlaya IS
      TASK TYPE Participante;

      TASK TYPE Equipo IS
         ENTRY Llegada;
         ENTRY Empezar;
         ENTRY Junte(suma: IN int);
         ENTRY NroEquipo(equipo: IN int);
      END Equipo;

      TASK Admin IS
         ENTRY SumarPorEquipo(suma: IN int; team: IN int);
         ENTRY Ganador(team: OUT int);
      END Admin;

     arrParticipantes: array (1..P) of Participante;
     arrEquipos: array (1..5) of Equipo;

     TASK BODY Participante IS
        team, moneda, suma, ganador: int;
     BEGIN
        team = (.) // se tiene previamente
        suma = 0;
        Equipo(team).Llegada();
        Equipo(team).Empezar();

        for i in 1..15 LOOP
            moneda = moneda();
            suma = suma + moneda;
        END LOOP;

        Equipo(team).Junte(suma);
        Equipo(team).Ganador(ganador);
     END Participante;
```

Concurrente | Práctica 5 | HOJA 9 | FECHA

```
TASK BODY Equipo IS
  team, suma, cant, finaliza, sumaE : int;
BEGIN
   ACCEPT NroEquipo ( t : IN int) DO
         team= t;
   END NroEquipo;
   SumaE=0;
   Cant=0;
   finaliza=0;

   While (finaliza <> 4) LOOP
        SELECT
             ACCEPT Llegada () DO
                  cant= cant +1;
             END Llegada;
        OR
             (WHEN cant==4) => ACCEPT Empezar ()

        OR
             ACCEPT Junte (suma:IN int) DO
                    sumaE = sumaE + suma;
                    finaliza = finaliza +1;
             END Junte;
        END SELECT;
   END LOOP;

   Administrador. SumaporEquipo(SumaE, team);
END Equipo;
```

NOTA

```
TASK BODY Administrador IS
  Suma, ganador, equipo: int;
  resul Array (1..5) of Int;

 BEGIN
   FOR i IN 1..5 LOOP
         ACCEPT SumaPorEquipo (suma: IN int, team: IN int) DO
                resul [team) = suma;
         END SumaPorEquipo;
   END LOOP;

   ganador = ObtenerGanador(resul);

  FOR i IN 1..20 LOOP
         ACCEPT Ganador (g: OUT int) DO
                g = ganador;
         END Ganador;
  END LOOP;
END Administrador;

 BEGIN
    For i IN 1..5 LOOP
         Equipo (i).nroEquipo(i);
    END LOOP;
END JuegoMaya;
```

Concurrente Práctica 5

```
8) PROCEDURE Policia IS
     TASK Especialista IS
        ENTRY PedirHuella(huella: OUT img);
        ENTRY Resultado (codigo: IN int, valor: IN int);
    END Especialista;
    TASK TYPE Servidor IS { ENTRY ID(id: IN int)} END Servidor;
    TASK TYPE BaseDatos IS
          ENTRY Buscar (test: IN img, codigo: OUT int, valor: OUT int);
    END BaseDatos;
    arrServidores: array (1..8) of Servidor;
    arrBaseDatos: array (1..8) of BaseDatos;
    TASK BODY Especialista IS
      huella: img;
      valorMax, codigoMax: int;
     BEGIN
      valorMax= -1;
      Loop
        huella = obtenerHuella();
        FOR i IN 1..16 Loop
          SELECT
              ACCEPT PedirHuella(h: OUT img) DO
                     h = huella;
              END PedirHuella;
          OR
              ACCEPT Resultado (valor: IN int, codigo: IN int) DO
                   IF (valor > valorMax) THEN
                       valorMax= valor;
                       codigoMax= codigo;
                   END IF.
              END Resultado;
          END SELECT;
        END Loop;
      END Loop.
    END Especialista;
```

```
TASK BODY Servidor IS
  huella: img;
  valor, codigo: int;          id: int;
BEGIN
   ACCEPT ID(id: IN int);
   LOOP
      Especialista. PedirHuella (huella);
      arrBaseDatos (id). Buscar (huella, codigo, valor);
      Especialista. resultado (valor, codigo);
   END LOOP;
END Servidor;

TASK BODY BaseDatos IS
BEGIN
   LOOP
     ACCEPT Buscar(huella: IN img, codigo: OUT int, valor: OUT int) DO
          realizarBusqueda(huella, codigo, valor);
     END Buscar;
   END LOOP;
END BaseDatos;

BEGIN
  FOR i IN 1..8 LOOP
     arrServidores (i). ID (i);
  END LOOP;
END Policia;
```