PC | PRACTICA REPASO MD

PASAJE DE MENSAJES

EJERCICIO 1

A)

```
Chan documentos(text);
Chan pedido(int);
Chan siguiente[3](text);

Process Empleado[id:0..99] {
    text doc;
    while true {
        doc = getDocumento();
        Send documentos(doc);
    }
}

Process Impresora[id:0..4] {
    text doc;
    while true {
        Send pedido(id);
        Receive siguiente[id](doc);
        if (doc <> "NO") {
            Imprimir(doc);
        }
    }
}

Process Coordinador {
    int idI;
    text doc;
    while (true) {
        Receive pedido(idI);
```

```
        if (empty(documentos)) {
            doc = "NO";
        } else {
            Receive documentos(doc);
        }
        Send siguiente[idI](doc);
    }
}
```

B)

```
Process Empleado[id:0..99] {
    text doc;
    while (true) {
        doc = getDocumento();
        Admin!documentos(doc);
    }
}

Process Impresora[id:0..4] {
    text doc;
    while (true) {
        Admin!pedido(id);
        Admin?siguiente(doc);
        Imprimir(doc);
    }
}

Process Admin {
    int idI;
    text doc;
    Cola fila<text>;
    do Empleado[*]?documentos(doc) -> fila.push(doc);
            not empty(fila); Impresora[*]?pedido(idI) ->
                        doc = fila.pop();
                        Impresora[idI]!siguiente(doc);
```

```
    od
}
```

EJERCICIO 2

```
Process Persona[id:0..P-1] {
    Admin!usar(id);
    Admin?acceso();
    UsarTerminal();
    Admin!liberar();
}

Process Admin {
    int idP;
    boolean libre = true;
    Cola fila<int>;
    do Persona[*]?usar(idP) -> if (not libre) {

                                                                }


                                                                }
            Persona[*]?liberar() -> if (empty(usar)) {

                                                                }


                                                                }
    od
}
```

EJERCICIO 3

```
Chan llegueEmb(int);
Chan llegueMas5(int);
Chan llegue(int);
Chan pasar[P]();
Chan pagar(array of Boleta, int);
```

```
Chan resultado[P](int, Recibo);
Chan pedido();
Chan siguiente(int);

Process Persona[id:0..P-1] {
    int dinero, vuelto;
    array boletas of Boleta = getBoletasAPagar();
    boolean embarazada = getIsEmbarazada();
    Recibo recibo;
    if (embarazada) {
        Send llegueEmb(id);
    } elif (boletas.lenght() > 5) {
        Send llegueMas5(id);
    } else {
        Send llegue(id);
    }
    Receive pasar[id]();
    dinero = getDinero();
    Send pagar(boletas, dinero);
    Receive resultado[id](vuelto, recibo);
}

Process Admin {
    int idP;
    while (true) {
        Receive pedido();
        if (not empty(llegueEmb)) {
            Receive llegue(id);
        } elif (not empty(llegueMas5)) {
            Receive llegueMas5(id);
        } elif (not empty(llegue)) {
            Receive llegue(idP);
        } else {
            idP = -1;
        }
        Send siguiente(idP);
    }
}
```

```
Process Cajero {
    int idP, dinero, vuelto;
    array boletas of Boleta;
    Recibo recibo;
    while (true) {
        Send pedido();
        Receive siguiente(idP);
        if (idP <> -1) {
            Send pasar[idP]();
            Receive pagar(boletas, dinero);
            recibo = ProcesarPago(boletas, dinero);
            vuelto = getVuelto(boletas, dinero);
            Send resultado[idP](vuelto, recibo);
        }
    }
}
```

ADA

EJERCICIO 1

```
PROCEDURE BancoCentral is

    TASK TYPE Banco is
        Entry cotizacion(compra: OUT integer; venta: OUT inte
    End Banco;

    TASK BancoCentral;

    arrBancos: array(1..20) of Banco;

    TASK BODY Banco is
        c, v: integer;
    Begin
        Accept cotizacion(c: OUT integer; v: OUT integer) do
            c := getValorCompra();
```

```
            v := getValorVenta();
        end cotizacion;
    End Banco;

    TASK BODY BancoCentral is
        compra, venta: integer;
        valoresCompra: array(1..20) of integer;
        valoresVenta: array(1..20) of integer;
    Begin
        for i in 1..20 loop
            SELECT
                arrBancos[i].cotizacion(compra, venta);
                valoresCompra[i] := compra;
                valoresVenta[i] := venta;
            OR DELAY 5.0
                valoresCompra[i] := Null;
                valoresVenta[i] := Null;
            END SELECT;
    End BancoCentral;

BEGIN
    Null;
END BancoCentral;
```

EJERCICIO 2

```
PROCEDURE CobrosDigitales is

    TASK TYPE Persona;

    TASK Cajero is
        Entry llegueAnciano(id: IN integer; boletas: IN array
        Entry llegueMenos5(id: IN integer; boletas: IN array(
        Entry llegue(id: IN integer; boletas: IN array(1..N) (
    End Cajero;

    arrPersonas: array(1..P) of Persona;
```

```
TASK BODY Persona is
    edad, dinero, vuelto: integer;
    boletas: array(1..N) of Boleta;
    recibos: array (1..N) of Recibo;
Begin
    edad := getMiEdad();
    boletas := getMisBoletas();
    dinero := getDineroPago();
    if (soyAnciano(edad)) then
        Cajero.llegueAnciano(id, boletas, dinero, vuelto,
    else
        if (boletas.lenght() < 5) then
            Cajero.llegueMenos5(id, boletas, dinero, vuel
        else
            Cajero.llegue(id, boletas, dinero, vuelto, re
        end if;
    end if;
End Persona;

TASK BODY Cajero is
    v: integer;
    bols: array(1..N) of Boleta;
    recs: array(1..N) of Recibo;
Begin
    loop
        SELECT
            Accept llegueAnciano(boletas: IN array(1..N)
                bols := boletas;
                v := dinero;
                for i in 1..bols.lenght() loop
                    v := v - cobrarBoleta(bols[i]); //Dev
                    recs[i] := generarRecibo(bols[i]);
                end loop;
                vuelto := v; //Lo que sobro de todas las
                recibos := recs;
            end llegueAnciano;
        OR
```

```
                WHEN (llegueAnciano'count = 0) => Accept lleg
                    bols := boletas;
                    v := dinero;
                    for i in 1..bols.lenght() loop
                        v := v - cobrarBoleta(bols[i]); //Dev
                        recs[i] := generarRecibo(bols[i]);
                    end loop;
                    vuelto := v; //Lo que sobro de todas las
                    recibos := recs;
                end llegueMenos5;
            OR
                WHEN (llegueAnciano'count = 0) AND (llegueMen
                    bols := boletas;
                    v := dinero;
                    for i in 1..bols.lenght() loop
                        v := v - cobrarBoleta(bols[i]); //Dev
                        recs[i] := generarRecibo(bols[i]);
                    end loop;
                    vuelto := v; //Lo que sobro de todas las
                    recibos := recs;
                end llegue;
            END SELECT;
        end loop;
    End Cajero;

BEGIN
    Null;
END CobrosDigitales;
```

EJERCICIO 3

```
PROCEDURE VentaIndumentaria is

    TASK TYPE Sucursal;

    TASK OficinaCentral is
        Entry pedido(articulo: OUT integer);
```

```
        Entry resultado(veces: IN integer);
    End OficinaCentral;

    arrSucursales: array(1..100) of Sucursal;

    TASK BODY Sucursal is
        id, ventas: integer;
    Begin
        loop
            OficinaCentral.pedido(id);
            ventas := obtenerVentas(id);
            OficinaCentral.resultado(ventas);
        end loop;
    End Sucursal;

    TASK BODY OficinaCentral is
        art, total: integer;
    Begin
        loop
            total := 0;
            art := generarArticulo();
            for i in 1..200 loop
                SELECT
                    Accept pedido(articulo: OUT integer) do
                        articulo := art;
                    end pedido;
                OR
                    Accept resultado(veces: IN integer) do
                        total := total + veces;
                    end resultado;
                END SELECT;
            end loop;
        end loop;
    End OficinaCentral;

BEGIN
    Null;
END VentaIndumentaria;
```