Repaso 2do parcial

1. En una oficina existen 100 empleados que envían documentos para imprimir en 5 impresoras
   compartidas. Los pedidos de impresión son procesados por orden de llegada y se asignan a la primera
   impresora que se encuentre libre:
   a) Implemente un programa que permita resolver el problema anterior usando PMA.
   b) Resuelva el mismo problema anterior pero ahora usando PMS.

a)

```
chan pedidos(doc)
chan avisarImpresora[5](doc)
chan impresorasLibres(id)


process empleado[id:0..99]{
    text pedido;
    pedido = //generarDoc()
    send pedidos(pedido)
}

process admin{
    int id
    text doc
    while(true){
        receive impresorasLibres(id)
        receive pedidos(doc)
        send avisarImpresora[id](doc)
    }
```

```
}


process impresora[id:0..4]{
    text doc
    while (true){
        send impresorasLibres(id)
        receive avisarImpresora[id](doc)
        //imprimir(doc)
    }
}
```

b)

```
process empleado[id:0..99]{
    text pedido;
    pedido = //generarDoc()
    admin!darDoc(pedido)
}

process admin{

    cola buffer;
    text pedido;
    int id;

    do  empleado[*]?darDoc(pedido) -> buffer.push(pedido)

    []  not empty(buffer); impresora[*]?esperarDoc(id) ->
```

```
    od


}


process impresora[id:0..4]{
    text doc
    while (true){
        admin!esperarDoc(id)
        admin?recibirDoc(doc)
        //imprimir(doc)
    }
}
```

1- Resolver el siguiente problema. La página web del Banco Central exhibe las diferentes cotizaciones del dólar oficial de 20 bancos del país, tanto para la compra como para la venta. Existe una tarea programada que se ocupa de actualizar la página en forma periódica y para ello consulta la cotización de cada uno de los 20 bancos. Cada banco dispone de una API, cuya única función es procesar las solicitudes de aplicaciones externas. La tarea programada consulta de a una API por vez, esperando a lo sumo 5 segundos por su respuesta. Si pasado ese tiempo no respondió, entonces se mostrará vacía la información de ese banco.

```
procedure ej1 is

    Task type ApiBanco
        entry solicitarValores(venta: out real, compra: out rea

    apis = array (1..20) of ApiBanco
```

```
    Task body ApiBanco is
        real compra,venta;
    begin
        loop
            compraDolar := //obtenerValorCompra()
            ventaDolar := //obtenerValorVenta()
            accept solicitarValores(venta: out real, compra: out
                venta := ventaDolar
                compra := compraDolar
            end solicitarValores
        end loop
    end

    Task consulta is

    Task body consulta is
        valoresCompra array (1..20) of real;
        valoresVenta array (1..20) of real;
        integer periocidad;
    begin
        periocidad := //obtenerPeriocidad()
        loop
            for i in 1..20 loop
                SELECT
                    apis[i].solicitarValores(valoresVenta[i],val
                OR DELAY 5
                    valoresVenta[i] := 'blanco'
                    valoresCompra[i] := 'blanco'
            //Mostrar(valoresCompra,valoresVenta)
            DELAY periocidad
        end loop
    end consulta

begin
```

```
    null
end
```

2- Resolver el siguiente problema con PMS. En la estación de trenes hay una terminal de SUBE que
debe ser usada por P personas de acuerdo con el orden de llegada. Cuando la persona accede a la
terminal, la usa y luego se retira para dejar al siguiente. Nota: cada Persona usa sólo una vez la
terminal.

```
process persona[id:0..P-1]{
    terminalSube!pedirUso(id)
    terminalSube?recibirUso()
    //usarMaquina()
    terminalSube!dejarTerminal()


}

process terminalSube{
    cola buffer;
    boolean libre = true;
    int id;


    do persona[*]?pedirUso(id) ->
                                                    if (libr
                                                        libr
                                                        pers
                                                    }
                                                    else
                                                        buf
```

```
    [] persona[*]?dejarTerminal()->
                                                    if (buf
                                                        pers
                                                    else
                                                        libr
    od
}
```

2- Resolver el siguiente problema. En un negocio de cobros digitales hay P personas que deben pasar por la única caja de cobros para realizar el pago de sus boletas. Las personas son atendidas de acuerdo con el orden de llegada, teniendo prioridad aquellos que deben pagar menos de 5 boletas de los que pagan más. Adicionalmente, las personas ancianas tienen prioridad sobre los dos casos anteriores. Las personas entregan sus boletas al cajero y el dinero de pago; el cajero les devuelve el vuelto y los recibos de pago.

```
procedure 2ada is

Task type persona;

personas = array (1..P) of persona

Task body persona is
    integer cantidadBoletas;
    boleta boletas;
    real dinero;
    integer edad;
    real vuelto;
    string component
    if (edad > 60)
        caja.pagarAnciano(vuelto,comprobante,dinero,boletas)
    else
```

```
        if (cantidadBoletas < 5)
            caja.pagarMenosBoletas(vuelto,comprobante,dinero,bol
        else
            caja.pagar(vuelto,comprobante,dinero,boletas)
end persona

Task caja is
    entry pagarAnciano(vuelto: out real, comprobante: out real,
    entry pagarMenosBoletas(vuelto: out real, comprobante: out r
    entry pagar(vuelto: out real, comprobante: out real, dinero
end caja

Task caja body is
begin
    for i in 1..P loop
        SELECT
            accept pagarAnciano(vuelto: out real, comprobante: o
                vuelto :=//cobrar(dinero,boletas)
                comprobante := //cobrar(dinero,boletas)
            end pagarAnciano
            OR
                WHEN (pagarAnciano'count = 0) ->




            OR
                WHEN ((pagarAnciano'count = 0) and (pagarMenosBo




        END SELECT
    end loop
end
begin
```

```
    null
end
```

3- Resolver el siguiente problema con PMA. En un negocio de cobros digitales hay P personas que
deben pasar por la única caja de cobros para realizar el pago de sus boletas. Las personas son
atendidas de acuerdo con el orden de llegada, teniendo prioridad aquellos que deben pagar menos
de 5 boletas de los que pagan más. Adicionalmente, las personas embarazadas tienen prioridad sobre los dos casos anteriores. Las personas entregan sus boletas al cajero y el dinero de pago; el cajero les devuelve el vuelto y los recibos de pago.

```
endchan comprobantes[P](string,real)
chan hayPago()
chan embarazada(

process persona[id:0..P-1]{
    boolean embarazada;
    int cantidadBoletas;
    boleta boletas[]
    double dinero;
    text comprobante;
    double vuelto;
    if (embarazada)
        send embarazada(dinero,boletas,id)
    else
        if (cantidadBoletas < 5)
            send menosBoletas(dinero,boletas,id)
        else
            send pagar(dinero,boletas,id)
    send hayPago()
```

```
    receive comprobantes[id](vuelto,comprobante)
}

process caja{
    int idPersona
    real dinero
    boleta boletas[]
    text comprobante
    double vuelto
    comprobante string
    for i := 1 to P{
        receive hayPago()
        if (!empty(embarazada))
             receive embarazada(dinero,boletas,idPersona)
        else
            if (!empty (menosBoletas))
                receive menosBoletas(dinero,boletas,idPersona)
            else
                receive pagar(dinero,boletas,idPersona)
        comprobante := //cobrar(dinero,boletas)
        vuelto := //generarComprobante(cobrar)
        send comprobantes[idPersona](comprobante,vuelto)
    }
```

3- Resolver el siguiente problema. La oficina central de una empresa de venta de indumentaria debe
calcular cuántas veces fue vendido cada uno de los artículos de su catálogo. La

empresa se compone
de 100 sucursales y cada una de ellas maneja su propia base de datos de ventas. La oficina central
cuenta con una herramienta que funciona de la siguiente manera: ante la consulta realizada para un
artículo determinado, la herramienta envía el identificador del artículo a las sucursales, para que cada
una calcule cuántas veces fue vendido en ella. Al final del procesamiento, la herramienta debe
conocer cuántas veces fue vendido en total, considerando todas las sucursales. Cuando ha terminado
de procesar un artículo comienza con el siguiente (suponga que la herramienta tiene una función
generarArtículo() que retorna el siguiente ID a consultar). Nota: maximizar la concurrencia. Existe
una función ObtenerVentas(ID) que retorna la cantidad de veces que fue vendido el artículo con
identificador ID en la base de la sucursal que la llama.

```
procedure ej3 is

    Task type sucursal;

    sucursales = array (1..100) of sucursal

    Task body sucursal is
        integer idArticulo;
        integer cant;
    begin
        loop
            oficina.recibirId(id)
            cant := ObtenerVentas(id)
            oficina.devolverCantidad(cant)
        end loop
```

```
    end sucursal

    Task oficina is
        entry recibirId(idArticulo : out integer);
        entry devolverCantidad(cant : in integer);
    end oficina

    Task body oficina is
        integer id;
        integer cantTotal;
    begin
        loop
            id := generarArticulo()
            cantTotal := 0
            for i in 1..100 loop
                accept recibirId(idArticulo: out integer)do
                    idArticulo := id
                end recibirId
            end loop
            for i in 1..100 loop
                accept devolverCantidad(cant: in integer)do
                    cantTotal += cant
            end loop
            //mostrar(id,cantTotal) muestra el articulo y la can
        end loop
    end
begin
    null
end
```