1. Resolver con Pasaje de Mensajes Sincrónicos (PMS) el siguiente problema. En un torneo de programación hay *1 organizador*, *N competidores* y *S supervisores*. El organizador comunica el desafío a resolver a cada competidor. Cuando un competidor cuenta con el desafío a resolver, lo hace y lo entrega para ser evaluado. A continuación, espera a que alguno de los supervisores lo corrija y le indique si está bien. En caso de tener errores, el competidor debe corregirlo y volver a entregar, repitiendo la misma metodología hasta que llegue a la solución esperada. Los supervisores corrigen las entregas respetando el orden en que los competidores van entregando. ***Nota:*** maximizar la concurrencia y no generar demora innecesaria.

```
Process Organizador {
    int idC;
    for i in 0..N-1 do
        string des = new Desafio();
        Competidor[*] ? pedido(idC);
        Competidor[idC] ! recibirDesafio(des);
    }
}

Process Competidor [id = 0..C-1] {
    string des, res;
    bool ok = false;
    Organizador ! pedio(id);
    Organizador ? recibirDesafio(des);
    while not(ok) {
        res = resolverDesafio(des);
        Coordinador ! entregarDesafio(id,des);
        Supervisor[*] ? recibirRespuesta (ok);
    }
}

Process Coordinador {
    Cola pedidos;
    string des;
    int idComp, idSup;
    while (true) {
        if (Competidor[*] ? entregarDesafio (idComp,des)) ->
            push(pedidos,(idComp,des));
        [] not(empty(pedidos)); Supervisor[*] ? pedirTrabajo(idSup) ->
            (idComp,des) = pop(pedidos);
            Supervisor[idSup] ! procesarTrabajo (idComp,des);
        end if;
    }
}

Process Supervidor [id = 0..S-1] {
    string des;
    bool ok = false;
    int idComp;
    while (true) {
        Coordinador ! pedirTrabajo (id);
        Coordinador ? procesarTrabajo (idComp,des);
        ok = corregirDesafio(des);
        Competidor[idComp] ! recibirRespuesta(ok);
    }
}
```

2. Resolver con ADA el siguiente problema. Una empresa de venta de calzado cuenta con *S sedes*. En *la oficina central* de la empresa se utiliza un sistema que permite controlar el stock de los diferentes modelos, ya que cada sede tiene una base de datos propia. El sistema de control de stock funciona de la siguiente manera: dado un modelo determinado, lo envía a las sedes para que cada una le devuelva la cantidad disponible en ellas; al final del procesamiento, el sistema informa el total de calzados disponibles de dicho modelo. Una vez que se completó el procesamiento de un modelo, se procede a realizar lo mismo con el siguiente modelo. **Nota:** suponga que existe una función *DevolverStock(modelo,cantidad)* que utiliza cada sede donde recibe como parámetro de entrada el modelo de calzado y retorna como parámetro de salida la cantidad de pares disponibles. Maximizar la concurrencia y no generar demora innecesaria.

```
Procedure ADA is

    task type Sede;

    task OficinaCentral is
        entry pedirModelo(m: OUT Modelo);
        entry darResultado (c: IN integer);
    end OficinaCentral;

    arrSedes: array (1..S) of Sede;

    task body Sede is
        mode: Modelo;
        cant: integer;
    begin
        loop
            OficinaCentral.pedirModelo(mode);
            DevolverStock(mode,cant);
            OficinaCentral.darResultado(cant);
        end loop;
    end Sede;

    task body OficinaCentral is
        mode: Modelo;
        cant, total: integer;
    begin
        loop
            mode := siguienteModelo();
            total := 0;
            -- Envía los pedidos
            for i in 1..S loop
                accept pedirModelo(m: OUT Modelo) do
                    m := mode;
                end pedirModelo;
            end loop;
            -- Recibe los resultados
            for i in 1..S loop
                accept darResultado(c: IN integer) do
                    total := total + c;
                end darResultado;
            end loop;
            -- Imprimir modelo y cantidad
        end loop;
    end OficinaCentral ;
Begin
    null;
End ADA;
```