




# Posibles soluciones a los ejercicios del parcial

Estas son posibles soluciones, no significa que sean la única forma, traté de hacerlas simples y siguiendo las ideas que hemos visto en las teorías y/o explicaciones prácticas para no confundir.




Resolver con **SEMÁFOROS** el siguiente problema. Simular el uso de un conjunto de 10 fotocopadoras en una empresa. Hay un *empleado* que se encarga de atender a los *N usuarios* que llegan para usar una fotocopadora, de acuerdo al orden de llegada. Cada usuario al llegar espera hasta que el empleado le indica a que fotocopadora ir, la usan y se retira. *Nota:* maximizar la concurrencia; suponga que hay una función *Fotocopiar()* llamada por el usuario que simula el uso de la fotocopadora.

```
sem mutexColaU = 1;  
sem hayUsuario = 0;  
sem mutexColaL = 1;  
sem pasar[N] = ([N] 0);  
sem libres = 10;
```

```
queue colaU, colaL;  
int numF[N];
```

```
Process Usuario[id: 0..C-1] {  
    P(mutexColaU);  
    push(colaU, id);  
    V(mutexColaU);  
    V(hayUsuario);  
    P(pasar[id]);  
    //fotocopiat(numF[id])  
    P(mutexColaL);  
    push(colaL, numF[id]);  
    V(mutexColaL);  
    V(libres);  
};
```

```
Process Empleado {  
    int i, idCli;  
  
    for (i=1; i<11; i++) push(colaL, i);  
  
    while (true) {  
        P(libres);  
        P(hayUsuario);  
        P(mutexColaU);  
        pop(colaU, idCli);  
        V(mutexColaU);  
        P(mutexColaL);  
        pop(colaL, numF[idCli]);  
        V(mutexColaL);  
        V(pasar[idCli]);  
    };  
};
```



Resolver con **MONITORES** la siguiente situación. Simular la atención en un corralón de materiales donde hay **4 empleados** para atender a  **$N$  clientes** de acuerdo al orden de llegada. Cuando el cliente llega espera a que alguno de los empleados lo llame, se dirige hasta el escritorio del mismo y le entrega el listado de materiales que debe comprar, luego queda esperando a que terminen de atenderlo y el empleado le entregue la factura. **Nota:** maximizar la concurrencia; suponga que existe una función *HacerFactura()* llamada por el empleado que simula la atención.

```

Process Cliente[id: 0..N-1] {
    text factura, pedido;
    int idE;

    Corralon.Encolar(id, idE);
    Escritorio[idE].Atender(pedido, factura);
};

```

```

Process Empleado[id: 0..3] {
    text factura, pedido;

    while (true) {
        Corralon.Siguiente(id);
        Escritorio[id].VerPedido(pedido);
        factura = HacerFactura(pedido);
        Escritorio[id].DarFactura(factura);
    }
};

```

```

Monitor Corralon {
    queue colaC;
    int numEmpleado[N];
    cond esperaC, esperaE;

```

```

Procedure Encolar (int C, OUT int E) {
    push(colaC, C);
    signal(esperaE);
    wait (esperaC);
    E = numEmpleado[C];
};

```

```

Procedure Siguiente(int idE) {
    int idC;

    while (empty(colaC)) wait(esperaE);
    pop(colaC, idC);
    numEmpleado[idC] = idE;
};
}

```

```

Monitor Escritorio[id: 0..3] {
    text pedido, factura;
    bool listo = false;
    cond espera, esperaE;

```

```

Procedure Atender (text P, OUT text F) {
    pedido = P;
    listo = true;
    signal (esperaE);
    wait (esperaC);
    F = factura;
    signal (esperaE);
};

```

```

Procedure VerPedido (OUT text P) {
    if (not listo) wait (esperaE);
    listo = false;
    P = pedido;
};

```

```

Procedure DarFactura (text F) {
    factura = F;
    signal (esperaC);
    wait (esperaE);
};
}

```