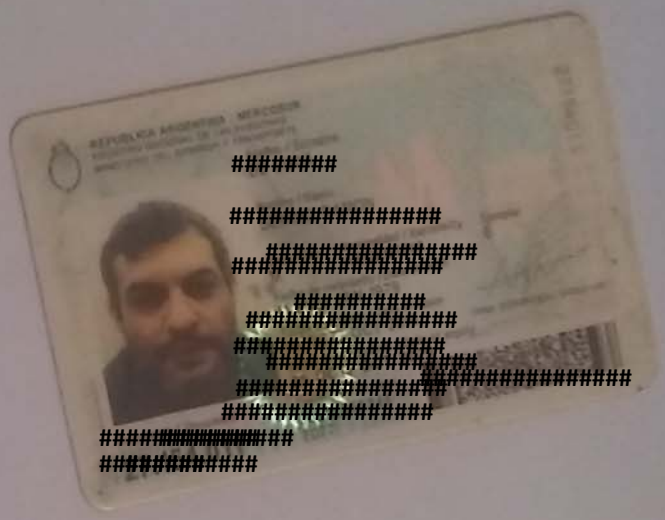
APROBADO

Sergio De Luca ejer 1 Hoja 1/2

```
int usuarios[N];
sem esperarFor[N](0); mutexUsr = 1; mutexF = 1; esperarUso[10] = (0,0,...,0);
queue usuarios;      sem enColaUsr = 0;  sem libro[10] = (1,1,...,1);
queue fotocopiadora; sem enColaFor = 0;  libroParaUso[10] = ([0] 0);
```

REGULAR

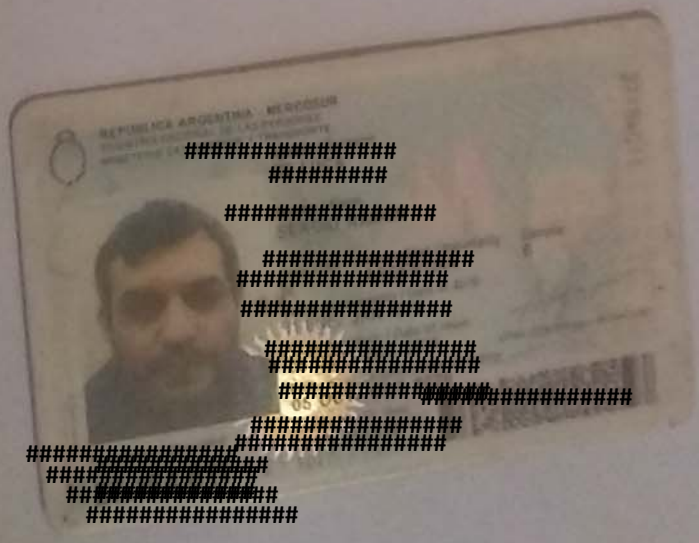
```
Process usuarios [i:1..N] {
int For;
    P(mutexUsr);
    usuarios.push(i);
    V(mutexUsr);
    V(enColaUsr);
    P(esperarFor[i]);
    For = usuarios[i];
    P(esperarUso[For]);
    V(libroParaUso[For]);
    fotocopiar();
    V(libro[For]);
}
```




```
Process fotocopiadoras [i:1..10] {
while true {
    P(mutexF);
    fotocopiadora.push(i);
    V(mutexF);
    V(enColaFor);
    P(libroParaUso[i]);
    // usando por usuario
    P(libro[i]);
}
}
```

Sergio De Luca ejer 1 hoja 2/2

```
Proceso empleado(){
int idUsr, idFot;
while true {
    P(enColaUsr);
    P(mtxUsr);
    idUsr = usuarios.pop();
    V(mtxUsr);
    usuarios[idUsr]
    P(enColaFot);
    P(mutxF);
    idFot = fotocopiad.pop();
    V(mtxF);
    usuarios[idUsr] = idFot;
    V(esperaFot[idUsr]);
    V(esperaUso[idFot]);
}
}
```

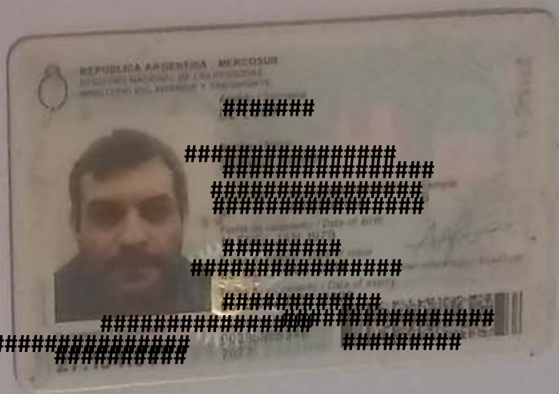


Se[redacted]ca [redacted] eje 2 hoja 1/2

BIEN-

```
Process clientes [i:1..N]{
int e; texto Ligrado, Factura;
corralon. esperar(e);
escritorio[e]. entregar(Ligrado);
escritorio[e]. recibirF(Factura);
}

Process empleados [i:1..4]{
texto Ligrado, Factura;
while (true)
      corralon. atender(i);
      escritorio[i]. recibir(Ligrado)
      Factura = HacerFactura();
      escritorio[i]. entregar(Factura);
  }
}
```




```
Monitor corralon {
    cola elibres;
    cond espera;
    int esperando = 0, libres = 0;

  Procedure esperar (int ide: out){
        if (Libres == 0){
              esperando ++;
              wait (espera();
        }else
              libres --;
        ide = elibres. pop
  Procedure proximo (int ide: in){
        push. elibres(ide);
        if (esperando > 0){
              esperando --;
              signal (espera(); }
        else libres ++;
```

Sergio De Luca 6773/1 eje 2 hoja 2/2

```
Monitor escritorio [i: 1..4] {
  text list, fact;
  cond aviso, recibe;
  bool okplani, okf;

  Procedure entregar (texto Lista: in) {
      list = Lista;
      okplani = true;
      signal (aviso);
  }

  Procedure recibir (texto Lista: out) {
      if (not okplani) wait (aviso);
      Lista = list;
  }

  procedure entregarF (texto f: in) {
      fact = f;
      okf = true;
      signal (recibe);
  }

  Procedure recibirF (texto f: out) {
      if (not okf) wait (recibe);
      f = fact;
  }
```

[redacted]