





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*** la siguiente situación. En una parcial hay **25 Ayudantes** que deben contar la cantidad total de parciales aprobados. Para esto cada Ayudante debe corregir 20 exámenes (suponga que ya los tiene en su poder) y determinar la nota (Aprobado o Desaprobado). Además hay un ***Profesor Titular*** que en base a la cantidad total de parciales aprobados debe calcular el porcentaje de aprobación. ***Nota:*** suponga que existe una función $nota = Corregir(examen)$ que es llamada por los ayudantes para corregir un *examen*, y retorna la *nota* del mismo (A o D). Maximizar la concurrencia.



```
sem mutex = 1;  
sem listo = 0;  
int totalAprobados = 0;
```

```
Process Usuario[id: 0..C-1] {  
    int i, cantAprobados;  
    text examenes[20];  
    char notas[20];  
    for (i=0; i<20; i++) {  
        notas[i] = corregir(exámenes[i]);  
        if (notas[i] == 'A') cantAprobados++;  
    };  
    P(mutex);  
    totalAprobados = totalAprobados + cantAprobados;  
    V(mutex);  
    V(listo);  
};
```

```
Process Titular{  
    int i;  
    float resultado;  
    for (i=0; i<25; i++) P(listo);  
    resultado = (totalAprobados * 100)/500;  
};
```



Resolver con **MONITORES** el siguiente problema. Simular la atención en un locutorio con 10 cabinas telefónicas, que tiene **un empleado** que se encarga de atender a los clientes. Hay **N clientes** que al llegar esperan hasta que el empleado les indica a que cabina ir, la usan y luego se dirigen al empleado para liberarla. El empleado atiende a los clientes en el orden en que hacen los pedidos, pero dando prioridad a los que quieren liberar una cabina. **Nota:** suponga que existe una función $UsarCabina(nroCabina)$ llamada por los clientes que representa el uso de la cabina $nroCabina$.

```

Process Cliente[id: 0..N-1] {
  int nroCabina;
  Admin.Encolar(nroCabina);
  UsarCabina(nroCabina);
  Admin.Liberar(nroCabina);
};

```

```

Process Empleado[id: 0..3] {
  int i, nro, cantLibres = 10;
  queue libres;
  text tipo;

  for (i=0; i<10; i++) push (libres, i);
  while (true) {
    Admin.Siguiente(cantLibres, tipo, nro);
    if (tipo = "Pedido") {
      pop (libres, nro);
      Admin.Asignar (nro);
      cantLibres--;
    }
    else {
      push (libres, nro);
      cantLibres++;
      Admin.Despedir();
    }
  };
};

```

Monitor Admin {

```

  int numCabina, cantP, cantL;
  cond esperaCP, esperaCL, esperaES, esperaEA;
  queue pedidoLiberar;

```

Procedure Encolar (OUT int num) {

```

  cantP++;
  signal(esperaE);
  wait (esperaCP);
  num = numCabina;
  signal (esperaEA);

```

```

};

```

Procedure Liberar(int num) {

```

  push (pedidoLiberar, num);
  cantL++;
  signal(esperaE);
  wait (esperaCL);

```

```

};

```

Procedure Siguiente (int c, OUT text t, OUT int n) {

```

  while ((cantL == 0) and (cantP == 0 or c == 0)) wait(esperaE);
  if (cantL == 0) {
    t = "Liberar";
    pop (pedidoLiberar, n);
  }
  else t = "Pedido";

```

```

};

```

Procedure Asignar (int num) {

```

  numCabina = num;
  signal (esperaCP);
  wait (esperaEA);

```

```

};

```

Procedure Despedir () {

```

  signal (esperaCL);

```

```

};

```

```


}

```

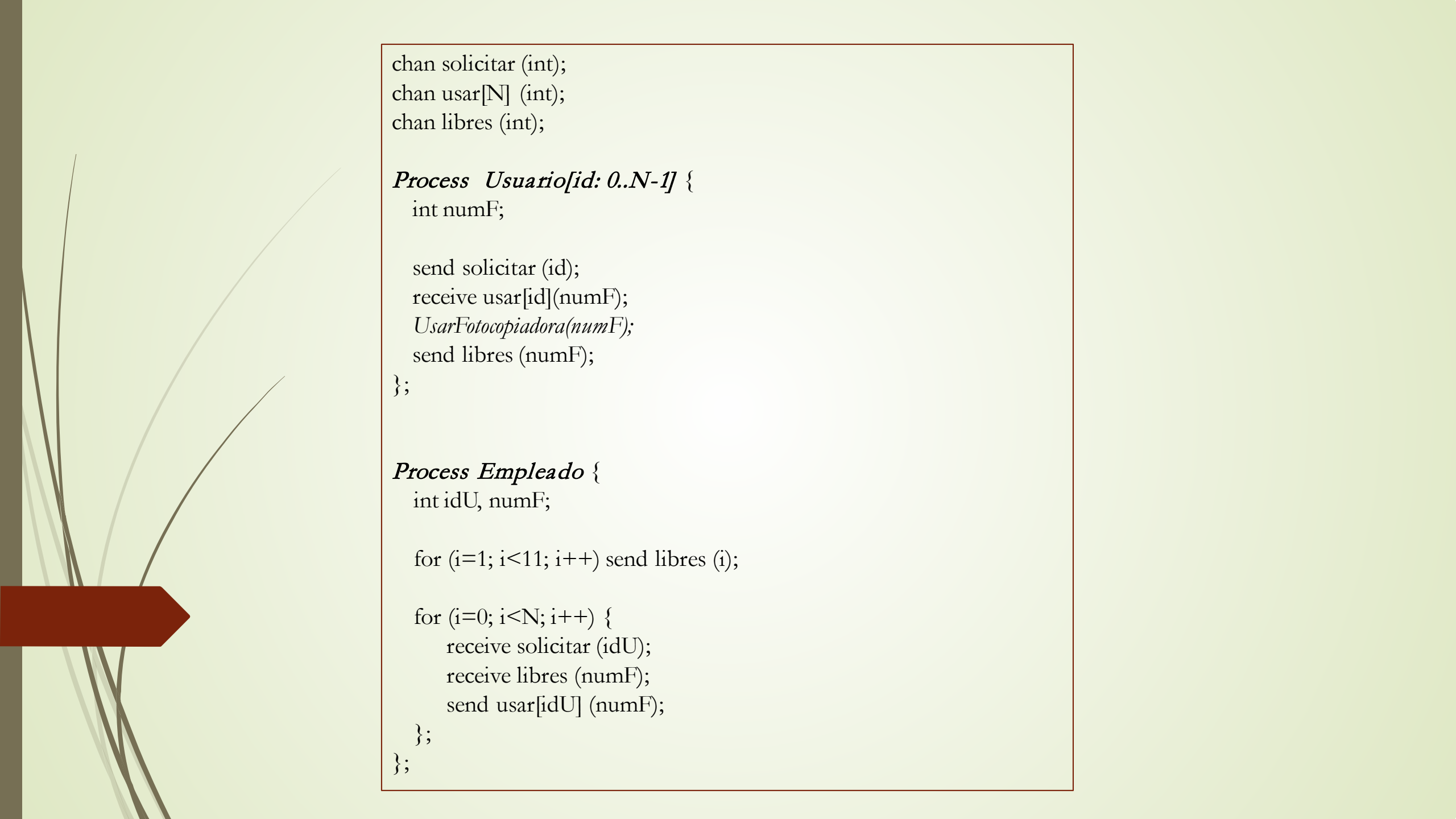


Posibles soluciones a los ejercicios del parcial

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 **PMA (Pasaje de Mensajes ASINCRÓNICOS)** 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 usa y se retira. ***Nota:*** suponga que hay una función *UsarFotocopiadora(*nroFot*)* llamada por el usuario que simula el uso de la fotocopadora *nroFot*. Sólo pueden existir los procesos Usuarios y el Empleado.



```
chan solicitar (int);  
chan usar[N] (int);  
chan libres (int);
```

```
Process Usuario[id: 0..N-1] {  
    int numF;  
  
    send solicitar (id);  
    receive usar[id](numF);  
    UsarFotocopiadora(numF);  
    send libres (numF);  
};
```

```
Process Empleado {  
    int idU, numF;  
  
    for (i=1; i<11; i++) send libres (i);  
  
    for (i=0; i<N; i++) {  
        receive solicitar (idU);  
        receive libres (numF);  
        send usar[idU] (numF);  
    };  
};
```

Resolver con **ADA** 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 cualquiera de los empleados lo atienda, y al terminar se retira. ***Nota:*** maximizar la concurrencia; suponga que existe una función *Atender()* llamada por el empleado que simula la atención.

Procedure ParcialADA is

task type *Cliente* **is**

entry Identificación (ident: IN integer);
entry Inicio (datos: OUT text);
entry Respuesta (res: IN text);

end *Cliente*;

task type *Empleado*;

task *Admin* **is**

entry Solicitar (id: IN integer);
entry Siguiente (id: OUT integer);

end *Admin*;

arrClientes: array (0..N-1) of *Cliente*;

arrEmpleados: array (0..3) of *Empleado*;

task body *Cliente* **is**

id: integer;
resultado: text;

begin

accept Identificación (ident: IN integer) do
 id := ident;
end Identificación;
Admin.Solicitar (id);
accept Inicio (datos: OUT text) do
 datos := *Listado de compras*
end Inicio;
accept Respuesta (res: IN text) do
 resultado := res;
end Respuesta;
end *Cliente*;

task body *Admin* **is**

idC: integer;

begin

loop
 accept Solicitar (id: IN integer) do
 idC := id;
 end Solicitar;
 accept Siguiente (id: OUT integer) do
 id := idC;
 end Siguiente;
end loop;
end *Admin*;

task body *Empleado* **is**

idC: integer;
resultado, listado: text;

begin

loop
 AdminSiguiente (idC);
 arrClientes(idC).Inicio (listado);
 resultado := Atender(listado);
 arrClientes(idC).Respuesta (resultado);
end loop;
end *Empleado*;

Begin

for i in 1..N loop
 arrClientes(i).Identificación(i);
end loop;
End ParcialADA;