




# 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. Un sitio web brinda un servicio de identificación de secuencias de ADN, para lo cual utiliza 5 servidores diferentes (cada uno utiliza algoritmos y bases de datos diferentes). Existen  $C$  clientes que hacen 20 pedidos cada uno a dicho sitio (debe esperar a que le respondan un pedido para hacer otro), en cada pedido envía un string (la secuencia que quiere identificar) y recibe como respuesta otro string que es la secuencia resultante, la cual imprime en pantalla. Cada vez que llega un pedido de un cliente, el sitio web le asigna la búsqueda a 2 de los 5 servidores (suponga que posee una función *Elegir* que devuelve los 2 servidores elegidos) y continúa atendiendo otro pedido (no espera los resultados). Cada *servidor* continuamente está resolviendo las búsquedas que le ha asignado el sitio web (de a una y en el orden de llegada) de la siguiente manera: procesa la búsqueda y al terminar, si es el primer servidor que identificó la secuencia para ese pedido le envía al cliente correspondiente el resultado, sino la descarta. En ambos casos continua con su siguiente búsqueda. **Nota:** maximizar la concurrencia.

***Process SitioWeb{***

```
int i, num, s1, s2;
string sec;

while (true) {
    P(hayPedido);
    P(mutexCola);
    pop(cola, (idCli, sec, num));
    V(mutexCola);
    Elegir(s1, s2);
    P(mutexColaServ[s1]);
    push(colaServ[s1], (idCli, sec, num));
    V(mutexColaServ[s1]);
    V(haySec[s1]);
    P(mutexColaServ[s2]);
    push(colaServ[s2], (idCli, sec, num));
    V(mutexColaServ[s2]);
    V(haySec[s2]);
};
```

```
sem mutexCola = 1;
sem hayPedido = 0;
sem mutexColaServ[5] = ([5] 1);
sem haySec[5] = ([5] 0);
sem esperaRespuesta[C] = ([C] 0);
sem mutexEstado[C,20] = ([C,20] 1);

queue cola, colaServ[5];
string estado[C, 20], respuesta[C];
```

***Process Cliente [id: 0..C-1] {***


```
string sec;
int i;

for (i=0; i<20; i++) {
    //generar secuencia SEC
    P(mutexCola);
    push(cola, (id, sec, i));
    V(mutexCola);
    estado[id, i] = "espera";
    V(hayPedido);
    P(esperaRespuesta[id]);
    print(respuesta[id]);
};
```


***Process Servidor[id: 0..4] {***

```
int idCli, num;
string sec, res;

while (true){
    P(haySec[id]);
    P(mutexColaServ[id]);
    pop(colaServ[id], (idCli, sec, num));
    V(mutexColaServ[id]);
    res = ResolverSecuencia(sec);
    P(mutexEstado[idCli, num]);
    if (estado[idCli, num] == "espera") {
        estado[idCli, num] == "listo"
        respuesta[idCli] = res;
        V(esperaRespuesta[idCli]);
    };
    V(mutexEstado[idCli, num]);
};
```



Resolver con **MONITORES** la siguiente situación. En una herrería hay 15 empleados que forman 5 grupos de 3 personas; los grupos se forman de acuerdo al orden de llegada (los 3 primeros pertenecen al grupo 1, los 3 siguientes al grupo 2, y así sucesivamente). Ni bien conoce el grupo al que pertenece el empleado comienza a trabajar (no debe esperar al resto de grupo para comenzar). Cada grupo debe hacer  $P$  productos; para cada producto se necesita que sus 3 empleados participen de la siguiente manera: cada empleado realiza una parte del producto independientemente del resto, y cuando los 3 terminaron trabajan juntos para armarlo. El grupo recién pueden comenzar a trabajar en un nuevo producto cuando terminó de armar el anterior. **Nota:** maximizar la concurrencia.



```
Process Empleado[id: 0..14] {  
  int i, numG;  
  
  Coordinador.DarGrupo(numG);  
  for (i=0; i<P; i++) {  
    // trabaja de forma individual  
    AdminGrupo[numG].armarProducto;  
  }  
};
```

```
Monitor Coordinador {  
  int cantidad = 0;  
  
  Procedure DarGrupo(int *idG) {  
    idG = cantidad div 3;  
    cantidad++;  
  }  
}
```

```
Monitor AdminGrupo[id: 0..4] {  
  int cantidad = 0;  
  cond espera;  
  
  Procedure armarProducto {  
    cantidad++;  
    if (cantidad == 3) {  
      // armar producto entre todos  
      signal_all (espera);  
      cantidad = 0;  
    }  
    else wait (espera);  
  }  
}
```