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 **PMS (Pasaje de Mensajes SINCRÓNICOS)** el siguiente problema. En una carrera hay C corredores, 3 Coordinadores. Al llegar los corredores deben dirigirse a los coordinadores para que cualquiera de ellos le dé el número de “chaleco” con el que van a correr. Los coordinadores atienden a los corredores de acuerdo al orden de llegada (cuando un coordinador está libre atiende al primer corredor que está esperando). Cuando todos los corredores ya recibieron su número de “chaleco” comienza la carrera. Cuando un corredor termina de correr busca la posición en que termino la carrera y se retira. **Nota:** maximizar la concurrencia.

```
Process Corredor[id: 0..C-1] {
   int num, pos;

   AdminOrden ! quieroNumero(id);
   Coordinador[*] ? tomaNumero(num);
   AdminBarrera ! listo();
   AdminBarrera ! iniciarCarrera();
   CorrerCarrera();
   Meta ! quieroPosicion(id);
   Meta ? tomaPosicion(pos);
};
```

```
Process Coordinador[id: 0..2] {
   int idC, num;

   while (true) {
        AdminOrden ! siguiente(id);
        AdminOrden ? tomaCorredor (idC);
        num = generarNumero();
        Corredor[idC] ! tomaNumero(num);
   };
};
```

```
Process AdminOrden{
   queue cola;
   int idC, idE;

   do Corredor[*] ? quieroNumero(idA) → push (cola, idC);
   □ (not empty(cola)); Coordinador[*] ? siguiente(idE) →
                    pop(cola, idC);
                    Coordinador[idE] ! tomaCorredor(idC);
   od
};
```

```
Process AdminBarrera{
   int i;

   for (i=0; i< C; i++) Corredor[*] ? listo();
   for (i=0; i< C; i++) Corredor[*] ? iniciarCarrera();
};
```

```
Process Meta {
   int i, idC, pos = 1;

   for (i=0; i< P; i++) {
        Corredor[*] ? quieroPosicion(idC);
        Corredor[idC] ! tomaPosicion(pos);
        pos++;
   };
};
```

Resolver con **PMA (Pasaje de Mensajes ASINCRÓNICOS)** el siguiente problema. En una oficina hay 3 empleados y P personas que van para ser atendidas para iniciar un trámite, o para buscar su resultado. Cuando una persona llega espera hasta ser atendido por cualquiera de los empleados, le indica que necesita (iniciar trámite o buscar el resultado de un trámite) y espera hasta que terminan de atenderlo y le devuelven: un número de trámite en el primer caso, un dictamen en el segundo caso. Los empleados atienden las solicitudes en orden de llegada; si no hay personas esperando, durante 5 minutos resuelven trámites pendientes (simular el proceso de resolver trámites pendientes por medio de un *delay*). Cuando se han atendido a las P personas los empleados se retiran. **Nota:** no generar demora innecesaria; cada persona hace sólo un pedido y termina; los empleados deben terminar.

```
chan solicitud (int, text);
chan respuesta[P] (text);
chan siguiente (int);
chan respuestaEmpleado[3] (int, text);
```

```
Process Persona[id: 0..P-1] {
   text res, tramite;

   send solicitud(id, tramite);
   receive respuesta[id] (res);
};
```

```
Process Empleado[id: 0..2] {
   int idP;
   text tramite, res;

   send siguiente (id);
   receive respuestaEmpleado[id] (idP, tramite);

   while (idP > -1) {
      if (idP > 0) {
            respuesta = resolverTramite(tramite);
            send respuesta[idP] (res);
      }
      else delay (5 minutos);

      send siguiente (id);
      receive respuestaEmpleado[id] (idP, tramite);
   };
};
```

```
Process Admin {
   int i, idP, idE, cant =0;
   text tramite;

   while (cant < P) {
      receive siguiente(idE);
      if (not empty (solicitud)) {
            receive solicitud(idP, tramite);
            send respuestaEmpleado[idE] (idP, tramite);
            cant++;
      }
      else idP = 0;
   };
   for (i=0; i<3; i++) {
      receive siguiente(idE);
      send respuestaEmpleado[idE] (-1, tramite);
   };
};
```

Resolver con **ADA** el siguiente problema. Hay una empresa de análisis genético. Hay N clientes que sucesivamente envían secuencias de ADN a la empresa para que sean analizadas y esperan los resultados para poder envían otra secuencia a analizar. Para resolver estos análisis la empresa cuenta con 4 servidores que van rotando su uso para no exigirlos de más (en todo momento uno está trabajando y los otros descansando); cada 6 horas cambia en servidor con el que se trabaja siguiendo un orden circular (1-2-3-4-1-2…). El servidor que está trabajando, toma un pedido (de acuerdo al orden de llegada de los mismos), lo resuelve y devuelve el resultado al cliente correspondiente. **Nota:** suponga que existe una función *Resolver(texto)* que utiliza cada Servidor para resolver el análisis de una secuencia de tipo texto y devuelve el resultado que es un entero.

```
Procedure ParcialADA is
   task type cliente;
   task type servidor is
      entry Resolver (sec: IN text; res: OUT integer);
   end servidor;
   task reloj is
      entry Iniciar;
   end reloj;
   task empresa is
      entry Pedido(S: IN text; R: OUT integer);
      entry FinTiempo;
   end empresa;
   arrClientes: array (0..N-1) of cliente;
   arrServidores: array (0..3) of servidor;
```

```
task body reloj is
begin
   accept Iniciar;
   loop
      delay (6 hs);
      Empresa.FinTiempo;
   end loop;
end reloj;
```

```
task body empresa is
   numS: integer := 0;
begin
   Reloj.Iniciar;
   loop
      select
         when (FinTiempo'count = 0) =>
            accept Pedido(S: IN text; R: OUT integer) do
               arrServidores[numS].Resolver(S, R);
            end Pedido;
      or
         accept FinTiempo do
            numS := numS mod 4;
         end FinTiempo;
      end select;
   end loop;
end empresa;
```

```
task body cliente is
   Resultado: integer;
   Secuencia: text;
begin
   loop
      Secuencia := GenerarSecuencia();
      Empresa.Pedido(Secuencia, Resultado);
   end loop;
end cliente;
```

```
task body servidor is
begin
   loop
      accept  Resolver(sec: IN text; res: OUT integer) do
         res := Resolver(sec);
      end Resolver;
   end loop;
end servidor;
```

```
Begin
   null;
End ParcialADA;
```