




# 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 **ADA** el siguiente problema. En una playa hay 5 equipos de 4 personas cada uno (en total son 20 personas donde cada una conoce previamente a que equipo pertenece). Cuando las personas van llegando esperan con los de su equipo hasta que el mismo esté completo (hayan llegado los 4 integrantes), a partir de ese momento el equipo comienza a jugar. El juego consiste en que cada integrante del grupo junta 15 monedas de a una en una playa (las monedas pueden ser de 1, 2 o 5 pesos) y se suman los montos de las 60 monedas conseguidas en el grupo. Al finalizar cada persona debe conocer el grupo que más dinero junto. **Nota:** maximizar la concurrencia. Suponga que para simular la búsqueda de una moneda por parte de una persona existe una función *Moneda()* que retorna el valor de la moneda encontrada.

## *Procedure ParcialADA is*

**task type** *jugador*;

**task type** *equipo* is

entry numeroEquipo(num: IN integer);

entry llegada;

entry esperaInicio;

entry darMonto(monto: IN integer);

entry resultado(mejor: OUT integer);

**end** *equipo*;

**task** *admin* is

entry fin(idE, cant: IN integer);

entry mejorEquipo(numE: OUT integer);

**end** *admin*;

arrJugadores: array (1..20) of jugador;

arrEquipos: array (1..5) of equipo;

**task body** *persona* is

idM: integer;

total: integer := 0;

idE: integer := ...; --conoce el equipo

**begin**

arrEquipos(idE).llegada;

arrEquipos(idE).esperaInicio;

for i in 1..15 loop

total := total + Moneda();

end loop;

arrEquipos(idE).darMonto(total);

arrEquipos(idE).resultado(idM);

**end** *persona*;

**task body** *equipo* is

miId, idM, parcial: integer;

total: integer := 0;

**begin**

accept numeroEquipo(num: IN integer) do

miId := num;

end numeroEquipo;

for i in 1..4 loop

accept llegada;

end loop;

for i in 1..4 loop

accept esperaInicio;

end loop;

for i in 1..4 loop

accept darMonto(monto: IN integer) do

total := total + monto;

end darMonto;

end loop;

admin.fin(miId, total);

admin.mejorEquipo(idM);

for i in 1..4 loop

accept resultado(mejor: OUT integer) do

mejor := idM;

end resultado;

end loop;

**end** *equipo*;

**task body** *admin* is

mMax: integer:= -1;

idMax: integer;

**begin**

for i in 1..5 loop

accept fin (idE, cant: IN integer) do

if (cant > mMax) then

mMax := cant;

idMax := idE;

end if;

end fin;

end loop;

for i in 1..5 loop

accept mejorEquipo(numE: out integer) do

numE:= idMax;

end mejorEquipo;

end loop;

**end** *admin*;


**Begin**

for i in 1..5 loop

arrEquipos(i).numeroEquipo(i);

end loop;

**End ParcialADA;**



Resolver con **PMS (Pasaje de Mensajes SINCRÓNICOS)** el siguiente problema. En una exposición aeronáutica hay un simulador de vuelo (que debe ser usado con exclusión mutua) y un empleado encargado de administrar el uso del mismo. Hay  $P$  personas que esperan a que el empleado lo deje acceder al simulador, lo usa por un rato y se retira. El empleado deja usar el simulador a las personas respetando el orden de llegada. **Nota:** cada persona usa sólo una vez el simulador.

```
Process Persona[id: 0..P-1] {  
  Empleado ! solicitarPaso(id);  
  Empleado ? pasar();  
  UsaSimulador;  
  Empleado ! salir();  
};
```

```
Process Empleado{  
  queue cola;  
  int idP;  
  bool libre = true;  
  
  do Persona[*] ? solicitarPaso(idP) →  
    if (not libre) push (cola, idP)  
    else {  
      libre = false;  
      Persona[idP] ! pasar();  
    }  
  □ Persona[*] ? salir() →  
    if (empty(cola)) libre = true  
    else {  
      pop(cola, idP);  
      Persona[idP] ! pasar();  
    }  
  od  
};
```