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*** el siguiente problema. Se debe simular una maratón con C corredores donde en la llegada hay 4 máquinas expendedoras de agua con capacidad para 20 botellas. Además existe un repositor encargado de reponer las botellas de todas las máquinas. Cuando los C corredores han llegado al inicio comienza la carrera. Cuando un corredor termina la carrera elige aleatoriamente una de las máquinas expendedoras y se dirige a ella, espera su turno (respetando el orden de llegada), saca una botella y se retira. Si encuentra la máquina sin botellas, le avisa al repositor para que cargue nuevamente la máquina con 20 botellas; espera a que se haga la recarga; saca una botella y se retira. ***Nota:*** maximizar la concurrencia; sólo se pueden usar los procesos corredor y repositor; no es necesario que el proceso repositor termine su ejecución.

```
sem mutexLlegada = 1;
sem correr = 0;
sem mutexMaquina[4] = ([4] 1);
sem espera[C] = ([C] 0);
sem mutexColaRep = 1;
sem reponer = 0;
sem esperandoReposicion[4] = ([4] 0);

int cantCorredores = 0;
bool libre[4] = ([4] true);
queue colaMaq[4], colaRep;
int cantBotellas[4] = ([4] 20);
```

```
Process Repositor{
  int numM;

  while (true) {
      P(reponer);
      P(mutexColaRep);
      pop(colaRep, numM);
      V(mutexColaRep);
      //reponer la máquina numM
      cantBotellas[numM] = 20;
      V(esperandoReposicion[numM]);
  };
};
```

```
Process Corredor[id: 0..C-1] {
  int i, numM, auxC;

  P(mutexLLegada);
  cantCorredores++;
  if (cantCorredores = C)
      for (i =0; i<C; i++) V(correr);
  V(mutexLlegada);
  P(correr);
  //corre carrera
  numM = random(0..3); //elige máquina
  P(mutexMaquina[numM]);
  if (not libre[numM]) {
      push(colaMaq[numM], id);
      V(mutexMaquina[numM]);
      P(espera[id]);
  }
  else {
      libre[numM] = false;
      V(mutexMaquina[numM]);
  };
  if (cantBotellas[numM] == 0) {
      P(mutexColaRep);
      push(colaRep, numM);
      V(mutexColaRep);
      V(reponer);
      P(esperandoReposicion[numM]);
  };
  //saca una botella
  cantBotella[numM]--;
  P(mutexMaquina[numM]);
  if (not empty(colaMaq[numM])) {
      pop(colaMaq[numM], auxC);
      V(espera[auxC]);
  }
  else libre[numM] = true;
};
```

Resolver con ***MONITORES*** el siguiente problema. Una empresa de turismo posee 4 combis con capacidad para 25 personas cada uno. Hay un único punto de venta donde C clientes comprar un pasaje para una combi en particular (el cliente conoce a que combi quiere ir); si aún hay lugar en la combi seleccionada se le da el pasaje y se dirige hacia la combi; en caso contrario se retira. Cada combi espera a que suban los 25 pasajeros, luego realiza el viaje, y cuando llega al destino baja a todos los pasajeros. ***Nota:*** maximizar la concurrencia; suponga que para cada combi al menos 25 clientes intentarán comprar pasaje.

```
Process Cliente [id: 0..N-1] {
   int numC = //combi seleccionada;
   bool pasaje = false;

   PuntoDeVenta.Comprar(numC, pasaje);
   if (pasaje)
        AdminCombi[numC].Subir;
};
```

```
Monitor PuntoDeVenta {
   int cantVendidos[4] = ([4] 0);

   Procedure Comprar(int numC, bool *pasaje) {
        if (cantVendidos[numC] < 25) {
              cantVendidos[numC]++;
              pasaje = true;
        else pasaje = false;
   }
}
```

```
Process Combi [id: 0..3] {

   AdminCombi[id].ComenzarViaje;
   //realiza el viaje
   AdminCombi[id].TerminoViaje;
};
```

```
Monitor AdminCombi [id: 0..3] {
   int cantArriba = 0;
   cond esperaCombi, esperaBajar;

   Procedure Subir {
        cantArriba++;
        if (cantArriba == 25)  signal (esperaCombi);
        wait (esperaBajar);
   }

   Procedure ComenzarViaje {
        if (cantArriba < 25)  wait (esperaCombi);
   }

   Procedure TerminoViaje {
        signal_all (esperaBajar);
   }
}
```