Posibles soluciones a los ejercicios del parcial práctico del 7-10-24

Importante: las soluciones que se muestran a continuación no son las únicas que se pueden considerar correctas para los ejercicios planteados.

1. Se debe simular el uso de un sistema virtual de venta de entradas para un evento musical. El sistema cuenta con C cajeros virtuales que atienden indefinidamente. Sin embargo, como la venta de entradas comienza a una hora determinada, sólo atienden a partir del aviso de un Timer. Una vez que reciben dicho aviso, los cajeros atienden de acuerdo con el orden de llegada de los compradores. La atención consiste en recibir la solicitud del comprador (datos para el pago) y responderle si pudo comprar (o no) junto al comprobante de la operación. Para este evento se cuenta con E entradas y N compradores, donde cada comprador puede solicitar a lo suma una entrada. Resuelva usando **SEMÁFOROS**.

```
sem sem_cant=1;
sem sem_cola=1;
sem sem_atencion=0;
sem sem_vendedor[C] = [C]{0};
sem sem_comprador[N] = [C]{0};

int cant_disponible = E;
Queue solicitudes;
Tuple respuestas[N];

Process Timer {
      // espera la hora indicada
      wait();
      // les avisa a los cajeros que pueden atender
      for i in 1..C
            V(sem_vendedor[i]);
}

Process Comprador [i=1..N] {

      Pago pago = generarPago();
      Comprobante comp;
      bool pudo_comprar;
      // ingresar solicitud
      P(sem_cola);
      push (id,pago);
      V(sem_cola);
      // solicitar atencion
      V(sem_atencion);
      // esperar respuesta
      P(sem_comprador[id]);
      // verificar si pudo comprar
      (pudo_comprar,comp) = respuestas[i];

}
```

```
Process Vendedor [i=1..C] {
      Pago pago;
      Comprobante comp;
      int id;
      bool pudo_comprar;
      // esperar aviso del Timer
      P(sem_vendedor[i]);
      // atender
      while (true) {
            // esperar solicitud
            P(sem_atencion);
            // desencolar solicitud
            P(sem_cola);
            (id, pago) = pop (solicitudes);
            V(sem_cola);
            pudo_comprar = false;
            // analizar disponibilidad
            P (sem_cant);
            if (cant_disponible > 0) {
                  cant_disponible --;
                  pudo_comprar = true;
            }
            V(sem_cant);
            // cobrar y responder
            comp = (pudo_comprar ? cobrar(pago) : null);
            respuestas[id] = (pudo_comprar, comp);
            // avisarle al comprador
            V(sem_comprador[id]);
      }
   }
```

2. Existen N personas que desean acceder a un mirador al borde del lago Nahuel Huapi en Bariloche. Como el mirador es angosto, sólo puede ser usado por una persona a la vez. Resuelva con **MONITORES** los dos casos siguientes:
   a. El acceso al mirador es por orden de llegada.

```
Monitor Mirador {
      int esperando = 0;
      int usando = 0;
      cond cola;

      procedure pedir (int id, int edad) {
            if (usando > 0){ // si lo están usando
                  // incrementar contador de espera y encolar
                  esperando++;
                  // dormirse en cola
                  wait(cola);
            }
            else // como está libre, marco que pasa a estar usado
                  usando++;
      }
```

```
      procedure liberar() {
            if (esperando > 0){ // si hay personas en espera
                  // despierto en cola
                  signal(cola);
                  // indico que hay uno menos en espera
                  esperando--;
            } else
                  // marco que vuelve a estar libre
                  usando--;
      }
}

Process Persona [i: 1..N]{
      // solicitar acceso
      Mirador.pedir();
      // usar mirador
      UsarMirador();
      // liberar
      Mirador.liberar();
}
```

b. El acceso al mirador es por orden de llegada, pero dando prioridad a los mayores de 60 años.

```
Alternativa 1: usando colas y un arreglo de variables condición


Monitor Mirador {
      int esperando = 0;
      int usando = 0;
      cond colas[N];
      Queue esperando_mayores;
      Queue esperando_menores;

      procedure pedir (int id, int edad) {
            if (usando > 0){ // si lo están usando
                  // incrementar contador de espera y encolar en cola correspondiente
según edad
                  esperando++;
                  if (edad >= 60)
                        push(esperando_mayores, id);
                  else
                        push(esperando_menores,id);
                  // dormirse en cola individual (no es correcto usar uno único)
                  wait(colas[id]);
            }
            else // como está libre, marco que pasa a estar usado
                  usando++;
      }
```

```
	procedure liberar() {
		if (esperando > 0){ // si hay personas en espera
			int id;
			// desencolo según prioridad
			if (not empty (esperando_mayores))
				id = pop(esperando_mayores);
			else
				id = pop(esperando_menores);
			// despierto en cola individual
			signal(colas[id]);
			// indico que hay uno menos en espera
			esperando--;
		} else
			// marco que vuelve a estar libre
			usando--;
	}
}

Process Persona [i: 1..N]{
	int edad = obtenerEdad();
	// solicitar acceso
	Mirador.pedir(i,edad);
	// usar mirador
	UsarMirador();
	// liberar
	Mirador.liberar();
}
```

Alternativa 2: sin usar colas y con dos variables condición en lugar de un arreglo

```
Monitor Mirador {
	int esperandoP = 0, esperandoReg = 0;
	int usando = 0;
	cond colaP, colaReg;

	procedure pedir (int edad) {
		if (usando > 0){ // si lo están usando
			if (edad >= 60) //lo duerme en la variable cond para prioritarios
				esperandoP++;
				wait (colaP)
			else  //lo duerme en la variable cond para no prioritarios
				esperandoReg++;
				wait (colaReg);
		}
		else // como está libre, marco que pasa a estar usado
			usando++;
	}

	procedure liberar() {
		int id;
```

```
            if (esperandoP > 0) { // si hay personas prioritaria en espera
                  // despierto al primero de ellos
                  signal (colaP);
                  esperandoP--;
            else  if (not esperandoReg > 0){   //si hay persona no prioritarias esperando
                        // despierto al primero de ellos
                        signal (colaReg);
                        esperandoReg--;

                  } else
                        // marco que vuelve a estar libre
                        usando--;
      }
}

Process Persona [i: 1..N]{
      int edad = obtenerEdad();
      // solicitar acceso
      Mirador.pedir(edad);
      // usar mirador
      UsarMirador();
      // liberar
      Mirador.liberar();
}
```