Posibles soluciones a los ejercicios del parcial práctico del 6-10-25

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

Tema 1

1. Implemente una solución para el siguiente problema usando **SEMÁFOROS**. Se debe simular el procesamiento de un sistema clasificador de 365 datasets meteorológicos. Todos los datasets se encuentran ya cargados en una estructura de datos. Para procesarla, el sistema dispone de 5 workers que trabajan colaborativamente procesando los datasets de a uno por vez. Cada procesamiento puede tomar un tiempo diferente y para realizarla los workers disponen de la función Procesar(d), la cual retorna como resultado un booleano que indica si los datos corresponden a clima tropical (true) o no (false). De acuerdo con el resultado, cada worker inserta el dataset en una cola de "tropicales" o de "no-tropicales". **Nota**: maximizar concurrencia.

```
sem sem_d = 1;
sem sem_trop = 1;
sem sem_notrop = 1;
Queue datasets = …; // Ya cargada
Queue tropicales, notropicales;

Process worker[i=0..4] {

    P(sem_d);
    // mientras haya datasets por procesar
    while (not empty(datasets)) do
       // saco uno con exlcusión mutua
        d = pop(datasets);
        V(sem_d);
       // lo proceso - no requiere exclusión mutua
        res = Procesar(d);
       // de acuerdo al resultado lo encolo. cada cola es independiente.
       if (res == true) then
             P(sem_trop);
             push(tropicales,d);
             V(sem_trop);
       else
             P(sem_notrop);
             push(notropicales,d);
             V(sem_notrop);
       end
       // vuelvo a chequear con exclusión mutua
        P(sem_d);
    end;
 // libero para que otro proceso lo pueda usar
    V(sem_d);
}
```

2. Implemente una solución para el siguiente problema usando **SEMÁFOROS**. En un taller de impresión 3D hay N clientes que solicitan piezas para ser impresas y esperan el resultado para poder retirarlas. El taller cuenta con una impresora 3D que fabrica las piezas en el orden de llegada de los pedidos. Una vez que le entregan la pieza, el cliente debe informar en pantalla si cumplió lo solicitado. **Nota**: la función ImprimirPieza(p) imprime la pieza en cuestión; la función ObtenerPedido(id) retorna el pedido de pieza para el cliente id; VerificarPieza(p) retorna true si el pedido p cumple lo solicitado o false en caso contrario.

```
sem sem_pedidos =1;
sem sem_impresora = 0;
sem_clientes[N] = [N] {0};
Queue pedidos;
Piezas piezas[N];

Process Cliente [i=0..N-1] {
	// obtiene pedido
	Pedido p = ObtenerPedido();
	// lo encola con exclusión mutua
	P(sem_pedidos);
	push(pedidos,(i,p));
	V(sem_pedidos);
	// avisa a la impresora
	V(sem_impresora);
	// espera por su pieza
	P(sem_clientes[i]);
	if (VerificarPieza(piezas[i]))
		print("Cumple lo solicitado");
	else
		print("No cumple lo solicitado");
	// se retira
}

Process Impresora {
	int id;
	Pedido p;
	for (int i=0; i < N; i++) {
		// espero a que llegue un cliente
		P(sem_impresora);
		// desencolo pedido con exclusión mutua
		P(sem_pedidos);
		(id,p) = pop(pedidos);
		V(sem_pedidos);
		// imprimo pieza
		piezas[id] = ImprimirPieza(p);
		// le aviso al cliente
		V(sem_clientes[id]);
	}
}
```

3. Implemente una solución para el siguiente problema usando **MONITORES**. En un evento hay 50 participantes que quieren tomarse fotos en una cabina fotográfica. La cabina puede ser usada por una persona a la vez, siguiendo el orden de llegada, pero dando prioridad a las personas de mayor edad (cuando la cabina está libre la debe usar la persona de mayor edad entre las que estén esperando para usarla). Cada participante se toma una única foto y luego se retira. **Nota**: solo se pueden utilizar procesos que representen a los participantes. La función obtenerEdad() retorna la edad de la persona; la función TomarFoto() simula el uso de la cabina.

```
Monitor Cabina {
	int esperando = 0;
	int usando = 0;
	cond colas[N];
	Queue(int,int) esperando;

	procedure pedir (int id, int edad) {
		if (usando > 0){
			esperando++;
			push(esperando, (edad, id)); // inserta ordenado por edad
			wait(colas[id]);
		}
		else
			usando++;
	}

	procedure liberar() {
		if (esperando > 0){
			int id = pop(esperando);
			signal(colas[id]);
			esperando--;
		} else
			usando--;
	}
}

Process Persona [i: 0..49]{
	Int edad = obtenerEdad();
	Cabina.pedir(i,edad);
	TomarFoto();
	Cabina.liberar();
}
```

TEMA 2

1. Implemente una solución para el siguiente problema usando **SEMÁFOROS**. Se debe simular el procesamiento de un sistema clasificador de 300 imágenes médicas. Todas las imágenes se encuentran ya cargadas en una estructura de datos. Para procesarla, el sistema dispone de 10 workers que trabajan colaborativamente procesando las imágenes de a una por vez. Cada procesamiento puede tomar un tiempo diferente y para realizarla los workers disponen de la función Procesar(img), la cual retorna como resultado un booleano que indica si los datos corresponden a un tejido anómalo (true) o no (false). De acuerdo con el resultado, cada worker inserta el dataset en una cola de "anomalas" o de "no-anomalas". **Nota**: maximizar concurrencia.

```
sem sem_i = 1;
sem sem_anom = 1;
sem sem_noanom = 1;
Queue imagenes = ...; // Ya cargada
Queue anomalas, noanomalas;

Process worker[i=0..9] {

    P(sem_i);
      // mientras haya imagenes por procesar
    while (not empty(imagenes)) do
            // saco uno con exclusión mutua
        img = pop(imagenes);
        V(sem_i);
            // la proceso - no requiere exclusión mutua
        res = Procesar(img);
            // de acuerdo al resultado lo encolo. cada cola es independiente.
            if (res == true) then
                  P(sem_anom);
                  push(anomalas,img);
                  V(sem_anom);
            else
                  P(sem_noanom);
                  push(noanomalas,img);
                  V(sem_noanom);
            end
            // vuelvo a chequear con exclusión mutua
        P(sem_i);
    end;
      // libero para que otro proceso lo pueda usar
    V(sem_i);
}
```

2. Implemente una solución para el siguiente problema usando **SEMÁFOROS**. En un servicio técnico de celulares hay N clientes que llevan sus dispositivos a reparar y esperan la devolución. El local cuenta con un técnico que atiende las reparaciones de acuerdo con el orden de llegada de los clientes. Una vez que le entregan el dispositivo, el cliente debe informar en pantalla si fue reparado. **Nota**: la función RepararDispositivo(d) repara el dispositivo d; la función ObtenerPedido(id) retorna el pedido de dispositivo para el cliente id; VerificarDispositivo(d) retorna true si el dispositivo d fue reparado o false en caso contrario.

```
sem sem_pedidos =1;
sem sem_tecnico = 0;
sem_clientes[N] = [N] {0};
Queue pedidos;
Dispositivos dispositivos[N];

Process Cliente [i=0..N-1] {
	// obtiene pedido
	Pedido p = ObtenerPedido();
	// lo encola con exclusión mutua
	P(sem_pedidos);
	push(pedidos,(i,p));
	V(sem_pedidos);
	// avisa al técnico
	V(sem_tecnico);
	// espera por su dispositivo
	P(sem_clientes[i]);
	if (RepararDispositivo(dispositivos[i]))
		print("Fue reparado");
	else
		print("No fue reparado");
	// se retira
}

Process Técnico {
	int id;
	Pedido p;
	for (int i=0; i < N; i++) {
		// espero a que llegue un cliente
		P(sem_tecnico);
		// desencolo pedido con exclusión mutua
		P(sem_pedidos);
		(id,p) = pop(pedidos);
		V(sem_pedidos);
		// reparo dispositivo
		dispositivos[id] = RepararDispositivo(p);
		// le aviso al cliente
		V(sem_clientes[id]);
	}
}
```

3. Implemente una solución para el siguiente problema usando **MONITORES**. En un parque natural hay 80 excursionistas que, en una parte del recorrido, deben cruzar un puente angosto que solo permite el paso de una persona a la vez. Los excursionistas cruzan siguiendo el orden de llegada pero dando prioridad a las personas de mayor edad (cuando el puente está libre, lo debe usar la persona de mayor edad entre las que estén esperando para usarlo). Cada excursionista cruza el puente una sola vez durante la excursión. **Nota**: solo se pueden utilizar procesos que representen a los excursionistas; la función obtenerEdad() retorna la edad de la persona; la función CruzarPuente() simula el cruce.

```
Monitor Puente {
	int esperando = 0;
	int usando = 0;
	cond colas[N];
	Queue(int,int) esperando;

	procedure pedirAcceso (int id, int edad) {
		if (usando > 0){
			esperando++;
			push(esperando, (edad, id)); // inserta ordenado por edad
			wait(colas[id]);
		}
		else
			usando++;
	}

	procedure liberarAcceso () {
		if (esperando > 0){
			int id = pop(esperando);
			signal(colas[id]);
			esperando--;
		} else
			usando--;
	}
}

Process Persona [i: 0..79]{
	Int edad = obtenerEdad();
	Puente.pedirAcceso(i,edad);
	CruzarPuente();
	Puente.liberarAcceso();
}
```