Posibles soluciones a los ejercicios del parcial práctico del 1-12-25

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

1. Una empresa de software cuenta con un sistema de compilación distribuida para acelerar el procesamiento de proyectos grandes. El sistema dispone de N workers, donde cada worker procesa exactamente un módulo. La compilación de cada módulo es independiente de las demás y la puede realizar cada worker en forma local. Una vez compilado, el worker debe subir su archivo objeto a un servidor de almacenamiento compartido (esta acción tarda un tiempo considerable). Este servidor solo puede ser usado por un worker a la vez, respetando el orden de llegada. La última etapa del proceso (enlace final) solo puede ejecutarse cuando todos los workers ya subieron su archivo al servidor. Una vez que todos los archivos fueron subidos, uno de los workers se ocupa de enlazar el proyecto para generar el ejecutable final (puede ser cualquiera de ellos, pero sólo uno). Implemente una solución al problema con **SEMÁFOROS** utilizando únicamente los procesos worker. **Notas:** maximizar la concurrencia; existe la función *compilarModulo()* que compila el módulo del worker que la invoca retornando un archivo objeto como resultado; existe la función *subirArchivoObjeto(ao)* sube el archivo objeto recibido como argumento al servidor compartido; la función *enlazarProyecto()* retorna el ejecutable final que debe guardarse en una variable compartida llamada *eje*.

```
sem s_serv=1; sem s_esperar[N]={[N]0}; sem s_cant=1;
bool libre = true; Queue cola; int cant=0; bin eje;

Process Workers[id=0..N-1] {
      // compila modulo en formal local
      obj ao = compilarModulo();
      // solicitar acceso al servidor
      P(s_serv)
      if (libre==false) {
            push(cola,id);
            V(s_serv)
            P(s_esperar[id]);
      } else {
            libre=false;
            V(s_serv);
      }
      // sube el archivo objeto con EM
      subirArchivoObjeto(ao);
      // liberar servidor
      P(s_serv)
      if (not empty(cola))
            V(esperar[pop(cola)]);
      else
            libre=true;
      V(s_serv)

      //sincronización para esperar a que todos lleguen y enlazar
      P(s_cant);
      cant = cant + 1;
      if (cant == N)  // enlaza proyecto final
            eje = enlazarProyecto();
      V(s_cant);
   }
```

2. En un banco se utiliza un cajero automático para realizar operaciones bancarias. Existen N Clientes que deben usarlo y un Empleado de seguridad que administra el acceso de acuerdo con el orden dado por la edad (cuando el cajero está libre, deja pasar al de mayor edad de entre los que están esperando por entrar). El cajero automático solo puede ser usado por una persona a la vez. Implemente una solución al problema con **MONITORES** utilizando procesos para representar a los Clientes y al Empleado. **Notas:** existe la función *UsarCajero()* que representa el uso del cajero; cada cliente conoce su edad mediante la función *obtenerEdad()*.

```
Monitor AdminCajero {
	cond colas[N], autoridad, fin;
	Queue(int,int) esperando;

	procedure llegada (int id, int edad) {
		push(esperando, (edad, id)); // inserta ordenado por edad
		signal(autoridad);
		wait(colas[id]);
	}

	procedure salida () {
		signal(fin);
	}

	procedure siguiente () {
		if (empty(esperando))
			wait (autoridad);
		int id = pop(esperando);
		signal(colas[id]);
		wait (fin);
	}
}

Process Cliente [i: 0..N-1]{
	int edad = obtenerEdad();
	// solicitar acceso
	AdminCajero.llegada(i,edad);
	// usar máquina
	UsarCajero();
	// liberar
	AdminCajero.salida();
}

Process Empleado {
	While (true) {
		// dar acceso para votar a la siguiente persona
		AdminCajero.siguiente();
	}
}
```