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();
	}
}
```

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. En un estadio de fútbol se dará un recital al que asistirán P personas. Para ello, cada asistente debe retirar su entrada por alguna de las 5 boleterías del estadio presentando su DNI (cada persona retira una única entrada). Los asistentes son atendidos por orden de llegada por alguno de los 5 empleados del club (cada uno atiende una boletería). Implemente una solución al problema usando **PMA**, considerando que los empleados realizan tareas administrativas por 15 minutos cuando no hay asistentes para atender. **Notas:** cada persona conoce su DNI; existe la función *obtenerEntrada(DNI)* que le retorna al empleado la entrada para el DNI provisto por argumento; existe la función *realizarTareasAdmin()* que representa las tareas administrativas que el empleado realizad por 15 minutos.

```
chan pedirEntrada(int,int);
chan pedirTrabajo(int);
chan recibirEntrada[P](Entrada);
chan asignarTrabajo[5](int,int);

Process Asistente [i:0..N-1] {
      Entrada ent;
      int dni = obtenerDNI();
      send pedirEntrada (i,dni);
      receive recibirEntrada [i] (ent);
}

Process Admin {
      int idA, idE, dni;
      while (true) {
            receive pedirTrabajo(idE);
            if (empty(pedirEntrada))
                  send asignarTrabajo[idE] (NULL, NULL);
            else {
                  receive pedirEntrada(idA, dni);
                  send asignarTrabajo[idE] (idA,dni);
            }
      }
}

Process Empleado [i:0..4] {
      int idA, dni;
      Entrada ent;
      while (true) {
            send pedirTrabajo(i);
            receive asignarTrabajo[i] (idA,dni);
            if (idA == NULL)
                  realizarTareasAdmin();
            else {
                  ent = obtenerEntrada(dni);
                  send recibirEntrada[idA] (ent);
            }
      }
}
```

2. Se desea modelar un sistema de recomendaciones para una plataforma de comercio electrónico. El sistema cuenta con 10 motores de recomendación, cada uno de ellos implementando un algoritmo distinto y operando sobre su propio conjunto de datos. Además, existe un Coordinador que genera solicitudes de recomendación y las procesa de manera indefinida. Cada vez que el coordinador genera una solicitud, la envía en paralelo a los 10 motores. Cada motor, al recibir la solicitud, ejecuta la función *Recomendar(solicitud, producto, puntaje)*, la cual retorna el producto recomendado para la solicitud ingresada y un puntaje numérico (de 0 a 100) que indica la relevancia de la recomendación generada. Cuando los 10 motores terminan, el Coordinador debe seleccionar el producto cuya recomendación tenga el mayor puntaje entre todas las recibidas. Luego de decidir la recomendación final, el Coordinador continúa inmediatamente procesando la siguiente solicitud. Implementar una solución de problema en **ADA**. **Notas:** maximizar concurrencia; la función *generarSolicitud()* genera la siguiente solicitud que se debe procesar; considere que la función Recomendar utilizada por los motores tarda un tiempo considerable.

```
Procedure ADA is

task type motor;

task coordinador is
      entry pedirSolicitud(soli: OUT Solictud);
      entry darRecomendacion (prod: IN Producto, puntaje: IN integer);
      entry siguienteRecomendacion;
end coordinador;

arrMotores: array (1..10) of motor;

task body motor is
      prod: Producto;
      soli: Solictud;
      puntaje: integer;
begin
      loop
            coordinador.pedirSolicitud(soli);
            Recomendar(soli, prod, puntaje);
            coordinador.darRecomendacion(prod, puntaje);
            coordinador.siguienteRecomendacion;
      end loop;
end motor;
```

```
task body coordinador is
      prod,prod_final: Producto;
      soli: Solicitud;
      puntaje, puntaje_max: integer;
begin
      loop
            soli := generarSolicitud();
            puntaje_max := -1;
            for i in 1..20 loop
                  select
                        accept pedirSolicitud(solicitud: OUT Solicitud) do
                              solicitud := soli;
                        end pedirSolicitud;
                  or
                        when (pedirSolicitud'count = 0) =>
                              accept darRecomendacion (prod: IN Producto, puntaje: IN
integer) do
                                    if (puntaje > puntaje_max) then
                                          puntaje_max := puntaje;
                                          prod_final := prod;
                                    end if;
                              end darRecomendacion;
                  end select;
            end loop;
            for i in 1..10 loop
                  accept siguienteRecomendacion;
            end loop;
      end loop;
end coordinador;

Begin
      null;
End ADA;
```