Posibles soluciones a los ejercicios del parcial práctico del 15-12-25

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 en la boletería del estadio presentando su DNI (cada persona retira una única entrada). Los asistentes son atendidos por orden de llegada por un empleado del club que atiende en la boletería. Implemente una solución al problema usando **PMA**, considerando que el empleado realiza tareas administrativas por 15 minutos cuando no hay asistentes para atender. **Notas:** cada persona conoce su DNI mediante la función *obtenerDNI()*; 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 realiza por 15 minutos.

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

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

Process Empleado {
      int idA, dni;
      Entrada ent;
      while (true) do
            if not(empty(pedirEntrada)) then
                  receive pedirEntrada (idA,dni);
                  ent = obtenerEntrada(dni);
                  send recibirEntrada[idA] (ent);
            else
                  realizarTareasAdmin();
      end
  }
```

2. En un estadio de fútbol se dará un recital al que asistirán P personas. Para ello, cada asistente debe retirar su entrada en 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 usando **PMS**. **Notas:** maximizar la concurrencia. Cada persona conoce su DNI mediante la función *obtenerDNI()*; la función *obtenerEntrada(dni)* le retorna al empleado la entrada para el DNI provisto por argumento.

```
Process Asistente [id=0..P-1] {
      int idE, dni = obtenerDNI();
      Entrada ent;
      //Solicita empleado para su atención
      Buffer!pedirEmpleado(id);
      Buffer?asignarEmpleado(idE);
      // enviar DNI y espera entrada
      Empleado[idE]!entregarDNI(dni);
      Empleado[idE]?entregarEntrada(ent);
}

Process Empleado[id=0..4] {
      int idA, dni;
      while (true) do
            Buffer!pedirAsistente(id);
            Buffer?asignarAsistente(idA);
            Asistente[idA]?entregarDNI(dni);
            ent = obtenerEntrada(dni);
            Asistente[idA]!entregarEntrada(ent);
      end
}

Process Buffer {
      Queue asistentes;
      int idA, idE;
      for i in 1..2*P {
            if (Asistente[*] ? pedirEmpleado (idA)) ->
                  push(asistentes,idA);
            [] (not(empty(asistentes)); Empleado[*] ? pedirAsistente(idE)) ->
                  idA = pop(asistentes);
                  Empleado[idE]!asignarAsistente (idA);
            end
      }
}
```

3. Se requiere modelar el acceso a un servidor de procesamiento de alto rendimiento que cuenta con 128 núcleos computacionales. Las tareas se dividen en livianas, medias y pesadas, requiriendo 1, 2, y 3 núcleos computacionales, respectivamente. Suponga que hay una cantidad conocida de tareas (T1 livianas, T2 medias y T3 pesadas) que se ejecutan una única vez. El servidor sólo puede ejecutar una tarea si tiene núcleos computacionales disponibles. Implemente una solución en **ADA** considerando que las tareas medias tienen prioridad sobre las pesadas, y las livianas sobre todas las demás.

```
Task Type TareaLiviana;
Task Type TareaMedia;
Task Type TareaPesada;

Task AdministradorDeServidor IS
      entry entrada_liviana();
      entry salida_liviana();
      entry entrada_media();
      entry salida_media();
      entry entrada_pesada();
      entry salida_pesada();
end AdministradorDeServidor;

admin: AdministradorDeServidor;
livianas: array (1..T1) of TareaLiviana;
medias: array (1..T2) of TareaMedia;
pesadas: array (1..T3) of TareaPesada;

TASK Body TareaLiviana IS
Begin
      admin.entrada_liviana ();
      -- Ejecutar
      admin.salida_liviana();
End

TASK Body TareaMedia IS
Begin
      admin.entrada_media ();
      -- Ejecutar
      admin.salida_media();
End

TASK Body TareaPesada IS
Begin
      admin.entrada_pesada ();
      -- Ejecutar
      admin.salida_pesada();
End
```

```
TASK Body AdministradorDeServidor IS
      nucleos_ocupados: int = 0;
Begin
      loop
            SELECT
                  Accept salida_liviana ();
                  nucleos_ocupados--;
            OR
                  Accept salida_media();
                  nucleos_ocupados-=2;
            OR
                  Accept salida_pesada();
                  nucleos_ocupados-=3;
            OR
                  WHEN ((nucleos_ocupados+3 <= 128) AND (entrada_liviana'count ==0)
AND (entrada_media'count ==0))
                        => Accept entrada_pesada ();
                        nucleos_ocupados+=3;
            OR
                  WHEN ((nucleos_ocupados+2 <= 128) AND (entrada_liviana'count ==0))
                        => Accept entrada_media ();
                  nucleos_ocupados+=2;
            OR
                  WHEN (nucleos_ocupados+1 <= 128)
                        => Accept entrada_liviana ();
                  nucleos_ocupados++;
            End SELECT;
      End loop;
End AdministradorDeServidor;
```