Resumen 2do parcial

Tipos de ejercicios PMA

Prioridades

- Se debe crear un chan para cada prioridad y el proceso que toma los elementos debe tener un if que de prioridad a los canales de “alta prioridad”
- Acordarse de siempre preguntar del canal que se va a sacar si tiene elementos
- En caso de que sean mas de 1 proceso “servidor” se debe agregar un coordinador

Ejercicio 5 práctica: Resolver la administración de 3 impresoras de una oficina. Las impresoras son usadas por N administrativos, los cuales están continuamente trabajando y cada tanto envían documentos a imprimir. Cada impresora, cuando está libre, toma un documento y lo imprime, de acuerdo con el orden de llegada. Modifique la solución implementada para que considere la presencia de un director de oficina que también usa las impresas, el cual tiene prioridad sobre los administrativos.

Ejercicio 3 práctica repaso: En un negocio de cobros digitales hay P personas que deben pasar por la única caja de cobros para realizar el pago de sus boletas. Las personas son atendidas de acuerdo con el orden de llegada, teniendo prioridad aquellos que deben pagar menos de 5 boletas de los que pagan más. Adicionalmente, las personas embarazadas tienen prioridad sobre los dos casos anteriores. Las personas entregan sus boletas al cajero y el dinero de pago; el cajero les devuelve el vuelto y los recibos de pago.

Ejercicio 1 2020: Se debe simular la atención en un banco con 3 cajas para atender a N clientes que pueden ser especiales (son las embarazadas y los ancianos) o regulares. Cuando el cliente llega al banco se dirige a la caja con menos personas esperando y se queda ahí hasta que lo terminan de atender y le dan el comprobante de pago. Las cajas atienden a las personas que van a ella de acuerdo al orden de llegada pero dando prioridad a los clientes especiales; cuando terminan de atender a un cliente le debe entregar un comprobante de pago. Nota: maximizar la concurrencia.

Si no hay clientes, servidores hacen otra cosa

- Ejercicios en los que te dicen “Si no hay clientes, se hace otra cosa”
- Se necesita un coordinador que mande “vacio” a los servidores
- Siempre hay más de un servidor

Ejercicio 1c práctica: Suponga que N clientes llegan a la cola de un banco y que serán atendidos por sus empleados. Considerando que hay 2 empleados y si no hay clientes para atender, los empleados realizan tareas administrativas durante 15 minutos.
Ejercicio 3 práctica: Se debe modelar el funcionamiento de una casa de comida rápida, en la cual trabajan 2 cocineros y 3 vendedores, y que debe atender a C clientes. El modelado debe considerar que:
- Cada cliente realiza un pedido y luego espera a que se lo entreguen.
- Los pedidos que hacen los clientes son tomados por cualquiera de los vendedores y se lo pasan a los cocineros para que realicen el plato. Cuando no hay pedidos para atender, los vendedores aprovechan para reponer un pack de bebidas de la heladera (tardan entre 1 y 3 minutos para hacer esto).
- Repetidamente cada cocinero toma un pedido pendiente dejado por los vendedores, lo cocina y se lo entrega directamente al cliente correspondiente.
Ejercicio 1 examenes adicionales: Resolver con PMA el siguiente problema. Se debe modelar el funcionamiento de una casa de venta de repuestos automotores, en la que trabajan V vendedores y que debe atender a C clientes. El modelado debe considerar que: (a) cada cliente realiza un pedido y luego espera a que se lo entreguen; y (b) los pedidos que hacen los clientes son tomados por cualquiera de los vendedores. Cuando no hay pedidos para atender, los vendedores aprovechan para controlar el stock de los repuestos (tardan entre 2 y 4 minutos para hacer esto).
Ejercicio 2 examenes adicionales: Resolver el siguiente problema con Pasaje de Mensajes Asincrónicos (PMA). En una empresa de software hay 3 programadores que deben arreglar errores informados por N clientes. Los clientes continuamente están trabajando, y cuando encuentran un error envían un reporte a la empresa para que lo corrija (no tienen que esperar a que se resuelva). Los programadores resuelven los reclamos de acuerdo al orden de llegada, y si no hay reclamos pendientes trabajan durante una hora en otros programas. Nota: los procesos no deben terminar (trabajan en un loop infinito); suponga que hay una función ResolverError que simula que un programador está resolviendo un reporte de un cliente, y otra Programar que simula que está trabajando en otro programa

Tipos de ejercicios PMS

Los ejercicios más comunes en los parciales son exclusión mutua y aquellos que son cliente-servidor

Exclusion mutua

- Hay uno o más recursos compartidos.
- Suelen decir algo como “hay un .... que debe ser usado por....”
- Se manejan muy parecido a como usábamos los monitores

Ejercicio 4 de la práctica: En una exposición aeronáutica **hay un simulador de vuelo (que debe ser usado con exclusión mutua)** y un empleado encargado de administrar su uso. Hay P personas que esperan a que el empleado lo deje acceder al simulador, lo usa por un rato y se retira. a) Implemente una solución donde el empleado sólo se ocupa de garantizar la exclusión mutua. b) Modifique la solución anterior para que el empleado considere el orden de llegada para dar acceso al simulador.

Ejercicio 5 de la práctica: En un estadio de fútbol hay una **máquina expendedora de gaseosas que debe ser usada por** E Espectadores de acuerdo con el orden de llegada. Cuando el espectador accede a la máquina en su turno usa la máquina y luego se retira para dejar al siguiente. Nota: cada Espectador una sólo una vez la máquina.

Ejercicio 2 práctica de repaso: Resolver el siguiente problema con PMS. En la estación de trenes hay una **terminal de SUBE que debe ser usada** por P personas de acuerdo con el orden de llegada. Cuando la persona accede a la terminal, la usa y luego se retira para dejar al siguiente. Nota: cada Persona usa sólo una vez la terminal.

Ejercicio horno: Resolver con Pasaje de Mensajes Sincrónicos (PMS) el siguiente problema. En un comedor estudiantil **hay un horno microondas que debe ser usado** por E estudiantes de acuerdo con el orden de llegada. Cuando el estudiante accede al horno, lo usa y luego se retira para dejar al siguiente. Nota: cada Estudiante una sólo una vez el horno.

Ejercicio aeronáutica: En una exposición aeronáutica hay un **simulador de vuelo (que debe ser usado con exclusión mutua)** y un empleado encargado de

administrar el uso del mismo. A su vez hay P personas que van a la exposición y solicitan usar el simulador, cada una de ellas espera a que el empleado lo deje acceder, lo usa por un rato y se retira para que el empleado deje pasar a otra persona. El empleado deja usar el simulador a las personas respetando el orden en que hicieron la solicitud. Nota: cada persona usa sólo una vez el simulador.

Ejercicio servidor: Resolver con PMS el siguiente problema: Se debe administrar el acceso para usar en determinado Servidor donde no se permite más de 10 usuarios trabajando al mismo tiempo, por cuestiones de rendimiento. Existen N usuarios **que solicitan acceder al Servidor, esperan hasta que se les de acceso para trabajar en él y luego salen del mismo**. Nota: suponga que existe una función TrabajarEnServidor() que llaman los usuarios para representar que están trabajando en el Servidor

Cliente-servidor

- Se nombra a un servidor que hace una tarea específica (más fácil de darte cuenta si es un empleado, profesor, etc)
- Los procesos servidor deben mandar señal de que están libres
- Hay un coordinador que recibe pedidos del clientes y mensajes de los servidores indicando si están libres

Ejercicio 3ayb de la práctica: En un examen final hay N alumnos y P profesores. Cada alumno resuelve su examen, lo entrega y **espera a que alguno de los profesores lo corrija** y le indique la nota. Los profesores corrigen los exámenes respetando el orden en que los alumnos van entregando.
a) Considerando que P=1. b) Considerando que P>1.

Ejercicio torneo de programación: Resolver con Pasaje de Mensajes Sincrónicos (PMS) el siguiente problema. En un torneo de programación hay 1 organizador, N competidores y S supervisores. El organizador comunica el desafío a resolver a cada competidor. Cuando un competidor cuenta con el desafío a resolver, lo hace y lo entrega para ser evaluado. A continuación, espera a que **alguno de los supervisores lo corrija y le indique si está bien**. En caso de tener errores, el competidor debe corregirlo y volver a entregar, repitiendo la misma metodología hasta que llegue a la solución esperada. Los supervisores corrigen las entregas respetando el orden en que los competidores van entregando. Nota: maximizar la concurrencia y no generar demora innecesaria.

Ejercicio carrera: En una carrera hay C corredores y 3 Coordinadores. Al llegar los corredores deben dirigirse a los **coordinadores para que cualquiera de ellos le dé el número de "chaleco"** con el que van a correr y luego se va. Los coordinadores atienden a los corredores de acuerdo al orden de llegada (cuando un coordinador está libre atiende al primer corredor que está esperando).

Ejercicio matrices: Existe un sitio de algebra lineal especializado en resolver multiplicaciones de matrices que atiende pedidos de N clientes, el sitio tiene un **Servidor encargado de resolver los pedidos de los clientes** de acuerdo con el orden en que se hacen los mismos. Cada cliente hace un solo pedido. NOTA: todas las tareas deben terminar.

Diferenciar ejercicios de exclusión mutua y los cliente-servidor

- En los ejercicios cliente-servidor suelen nombrar al servidor como un actor que "hace algo" mientras en los de exclusión mutua es un recurso que los clientes deben pedir y liberar. Por ej: en los C/S suele haber un empleado que atiende clientes, un profesor que corrige exámenes, etc. En cambio con exclusión mutua suele ser una cabina, un juego, una máquina que debe ser usada por los clientes pero que no realiza ninguna tarea por si misma.

- Ejercicio polémico: Resolver este ejercicio con PMS. En un banco se tiene un **sistema que administra el uso de una sala de reuniones** por parte de N clientes. Los clientes se clasifican en habituales o temporales. La sala puede ser usada por un unico cliente a la vez Y cuando esta libre se debe determinar a quien permitirle su uso siempre priorizando a los clientes habituales Dentro de cada clase de cliente se debe respetar el orden de llegada.
    - Argumento exclusión mutua: dice que el sistema administra, se entiende como que es quien le da el acceso pero luego el cliente es quien usa el recurso y lo libera
    - Argumento c/s: el sistema debería ser un proceso que se encarga de "administrar la sala" y le van llegando clientes

Barrera

Ejercicio 3c de la práctica: En un examen final hay N alumnos y P profesores. Cada alumno resuelve su examen, lo entrega y espera a que alguno de los profesores lo corrija y le indique la nota. Los profesores corrigen los exámenes respetando el orden en que los alumnos van entregando. c) Ídem b) pero considerando que **los alumnos no comienzan a realizar su examen hasta que todos hayan llegado al aula.** Nota: maximizar la concurrencia; no generar demora innecesaria; todos los procesos deben terminar su ejecución

Ejercicio guias: Resolver el siguiente problema con Pasaje de Mensajes Sincrónicos (PMS). En una excursión hay una tirolesa que debe ser usada por 20 turistas. Para esto hay un g uía y un empleado. **El empelado espera a que todos los turistas hayan llegado para darles una charla explicando** las medidas de seguridad. Cuando termina la charla los turistas piden usar la tirolesa y esperan a que el guía les vaya dando el permiso de tirarse. El guía deja usar la tirolesa a un cliente a la vez y de acuerdo al orden en que lo van solicitando. Nota: todos los procesos deben terminar; suponga que el empleado tienen una función DarCharla() que simula que el empleado está dando la charla, y los turistas tienen una función UsarTitolesa() que simula que está usando la tirolesa.

ADA

Prioridades

Ejercicio 1b práctica: Se requiere modelar un puente de un único sentido que soporta hasta 5 unidades de peso. El peso de los vehículos depende del tipo: cada auto pesa 1 unidad, cada camioneta pesa 2 unidades y cada camión 3 unidades. Suponga que hay una cantidad innumerable de vehículos (A autos, B camionetas y C camiones). Analice el problema y defina qué tareas, recursos y sincronizaciones serán necesarios/convenientes para resolverlo. a. Realice la solución suponiendo que todos los vehículos tienen la misma prioridad. b. Modifique la solución para **que tengan mayor prioridad los camiones que el resto de los vehículos.**

Ejercicio 4 práctica: En una clínica existe un médico de guardia que recibe continuamente peticiones de atención de las E enfermeras que trabajan en su piso y de las P personas que llegan a la clínica ser atendidos. Cuando una persona necesita que la atiendan espera a lo sumo 5 minutos a que el médico lo haga, si

pasado ese tiempo no lo hace, espera 10 minutos y vuelve a requerir la atención del médico. Si no es atendida tres veces, se enoja y se retira de la clínica. Cuando una enfermera requiere la atención del médico, si este no lo atiende inmediatamente le hace una nota y se la deja en el consultorio para que esta resuelva su pedido en el momento que pueda (el pedido puede ser que el médico le firme algún papel). Cuando la petición ha sido recibida por el médico o la nota ha sido dejada en el escritorio, continúa trabajando y haciendo más peticiones. El médico atiende los pedidos **dándole prioridad a los enfermos que llegan para ser atendidos**. Cuando atiende un pedido, recibe la solicitud y la procesa durante un cierto tiempo. Cuando está libre aprovecha a procesar las notas dejadas por las enfermeras.

Ejercicio 2 práctica repaso: Resolver el siguiente problema. En un negocio de cobros digitales hay P personas que deben pasar por la única caja de cobros para realizar el pago de sus boletas. Las personas son atendidas de acuerdo con el orden de llegada, teniendo prioridad aquellos que deben pagar menos de 5 boletas de los que pagan más. Adicionalmente, las personas ancianas tienen prioridad sobre los dos casos anteriores. Las personas entregan sus boletas al cajero y el dinero de pago; el cajero les devuelve el vuelto y los recibos de pago.

Ejercicio lectores/escritores: Resolver con ADA el siguiente problema. Se debe controlar el acceso a una base de datos. Existen L procesos Lectores y E procesos Escritores que trabajan indefinidamente de la siguiente manera:
Escritor: intenta acceder para escribir, si no lo logra inmediatamente, espera 1 minuto y vuelve a intentarlo de la misma manera.
Lector: intenta acceder para leer, si no lo logro en 2 minutos, espera 5 minutos y vuelve a intentarlo de la misma manera.
Un proceso Escritor podrá acceder si no hay ningún otro proceso usando la base de datos; al acceder escribe y sale de la BD. Un proceso Lector podrá acceder si no hay procesos Escritores usando la base de datos; al acceder lee y sale de la BD. Siempre se le debe dar prioridad al pedido de acceso para escribir sobre el pedido de acceso para leer.

Debo conocer mi id

- Los ejercicios en los que los procesos deben conocer su id pueden ser aquellos en los que hay procesos intermedios para que el servidor sepa a que cliente mandarle resultados

- También pueden ser ejercicios en los que se deba conocer por ej el proceso con el nro mas grande. En este caso el coordinador debe recibir de cada proceso su id para después determinar cual de esos id es el "ganador".

Ejercicio 5 práctica: En una playa hay 5 equipos de 4 personas cada uno (en total son 20 personas donde cada una conoce previamente a que equipo pertenece). Cuando las personas van llegando esperan con los de su equipo hasta que el mismo esté completo (hayan llegado los 4 integrantes), a partir de ese momento el equipo comienza a jugar. El juego consiste en que cada integrante del grupo junta 15 monedas de a una en una playa (las monedas pueden ser de 1, 2 o 5 pesos) y se suman los montos de las 60 monedas conseguidas en el grupo. Al finalizar cada persona debe conocer el grupo que más dinero junto. Nota: maximizar la concurrencia. Suponga que para simular la búsqueda de una moneda por parte de una persona existe una función Moneda() que retorna el valor de la moneda encontrada.

Ejercicio 8 práctica: Una empresa de limpieza se encarga de recolectar residuos en una ciudad por medio de 3 camiones. Hay P personas que hacen reclamos continuamente hasta que uno de los camiones pase por su casa. Cada persona hace un reclamo y espera a lo sumo 15 minutos a que llegue un camión; si no pasa, vuelve a hacer el reclamo y a esperar a lo sumo 15 minutos a que llegue un camión; y así sucesivamente hasta que el camión llegue y recolecte los residuos. Sólo cuando un camión llega, es cuando deja de hacer reclamos y se retira. Cuando un camión está libre la empresa lo envía a la casa de la persona que más reclamos ha hecho sin ser atendido. Nota: maximizar la concurrencia.

Ejercicio genética: Resolver el siguiente problema con ADA. Hay un sitio web para identificación genética que resuelve pedidos de N clientes. Cada cliente trabaja continuamente de la siguiente manera: genera la secuencia de ADN, la envía al sitio web para evaluar y espera el resultado; después de esto puede comenzar a generar la siguiente secuencia de ADN. Para resolver estos pedidos el sitio web cuenta con 5 servidores idénticos que atienden los pedidos de acuerdo al orden de llegada (cada pedido es atendido por un único servidor).

Barrera

Administrador indica cuándo arrancar

- Son aquellos ejercicios en que el servidor le indica a los clientes cuando arrancar a trabajar con un recurso y los clientes luego deben devolver resultados

Ejercicio 6 práctica: se debe calcular el valor promedio de un vector de 1 millón de números enteros que se encuentra distribuido entre 10 procesos Worker (es decir, cada Worker tiene un vector de 100 mil números). Para ello, existe un **Coordinador que determina el momento en que se debe realizar el cálculo** de este promedio y que, además, se queda con el resultado. Nota: maximizar la concurrencia; este cálculo se hace una sola vez.

Ejercicio 7 práctica: Hay un sistema de reconocimiento de huellas dactilares de la policía que tiene 8 Servidores para realizar el reconocimiento, cada uno de ellos trabajando con una Base de Datos propia; a su vez hay un Especialista que utiliza indefinidamente. El sistema funciona de la siguiente manera: el **Especialista toma una imagen de una huella (TEST) y se la envía a los servidores** para que cada uno de ellos le devuelva el código y el valor de similitud de la huella que más se asemeja a TEST en su BD; al final del procesamiento, el especialista debe conocer el código de la huella con mayor valor de similitud entre las devueltas por los 8 servidores. Cuando ha terminado de procesar una huella comienza nuevamente todo el ciclo.

Ejercicio 3 práctica repaso: Resolver el siguiente problema. La oficina central de una empresa de venta de indumentaria debe calcular cuántas veces fue vendido cada uno de los artículos de su catálogo. La empresa se compone de 100 sucursales y cada una de ellas maneja su propia base de datos de ventas. La oficina central cuenta con una herramienta que funciona de la siguiente manera: ante la consulta realizada para un artículo determinado, **la herramienta envía el identificador del artículo a las sucursales, para que cada una calcule** cuántas veces fue vendido en ella. Al final del procesamiento, la herramienta debe conocer cuántas veces fue vendido en total, considerando todas las sucursales. Cuando ha terminado de procesar un artículo comienza con el siguiente (suponga que la herramienta tiene una función generarArtículo() que retorna el siguiente ID a consultar).

Se debe esperar que todos los procesos hayan llegado

Ejercicio 5 práctica: En una playa hay 5 equipos de 4 personas cada uno (en total son 20 personas donde cada una conoce previamente a que equipo pertenece). **Cuando las personas van llegando esperan con los de su equipo hasta que el mismo esté completo (hayan llegado los 4 integrantes)**, a partir de ese momento el equipo comienza a jugar. El juego consiste en que cada integrante del grupo junta 15 monedas de a una en una playa (las monedas pueden ser de 1, 2 o 5 pesos) y se suman los montos de las 60 monedas conseguidas en el grupo. Al finalizar cada persona debe conocer el grupo que más dinero junto. Nota: maximizar la concurrencia. Suponga que para simular la búsqueda de una moneda por parte de una persona existe una función Moneda() que retorna el valor de la moneda encontrada.

Proceso intermedio

- Se necesita un proceso intermedio cuando hay más de un proceso cliente, más de un proceso servidor y al cliente se le debe asignar un servidor. El cliente no sabe a que servidor debe hablarle por eso surge un coordinador.
- La lógica es parecida a la de c/s de PMS ya que el cliente manda un pedido y el servidor pide que le manden un pedido(generalemente con un out del id del cliente que debe atender)

Ejercicio 8 práctica: Una empresa de limpieza se encarga de recolectar residuos en una ciudad por medio de 3 camiones. Hay P personas que hacen reclamos continuamente hasta que uno de los camiones pase por su casa. Cada persona hace un reclamo y espera a lo sumo 15 minutos a que llegue un camión; si no pasa, vuelve a hacer el reclamo y a esperar a lo sumo 15 minutos a que llegue un camión; y así sucesivamente hasta que el camión llegue y recolecte los residuos. Sólo cuando un camión llega, es cuando deja de hacer reclamos y se retira. Cuando un camión está libre la empresa lo envía a la casa de la persona que más reclamos ha hecho sin ser atendido. Nota: maximizar la concurrencia.

Ejercicio genética: Resolver el siguiente problema con ADA. Hay un sitio web para identificación genética que resuelve pedidos de N clientes. Cada cliente trabaja continuamente de la siguiente manera: genera la secuencia de ADN, la envía al sitio web para evaluar y espera el resultado; después de esto puede comenzar a generar la siguiente secuencia de ADN. Para resolver estos pedidos el sitio web cuenta con 5 servidores idénticos que atienden los pedidos de acuerdo al orden de llegada (cada pedido es atendido por un único servidor).

Resoluciones

PMA

Prioridades

```
91   chan menorPrioridad(int, text)
92   chan mediaPrioridad(int, text)
93   chan altaPrioridad(int, text)
94   chan pedido()
95   chan docs[P](text, text)
96   Process Persona[id:1..P]{
97       int prioridad
98       if(prioridad == 3)
99           send menorPrioridad(id, boleta)
100      elif(prioridad == 2)
101          send mediaPrioridad(id, boleta)
102      else
103          send altaPrioridad(id, boleta)
104      send pedido()
105      receive docs[id](vuelto, recibo)
106  }
107
108  Process Caja{
109      while(true)
110          receive pedido()
111          if(!altaPrioridad.empty())
112              receive altaPrioridad(idP, boleta)
113          elif(!mediaPrioridad.empty())
114              receive mediaPrioridad(idP, boleta)
115          else
116              receive menorPrioridad(idP, boleta)
117
118          vuelto, recibo = resolver(boleta)
119          send docs[idP](vuelto, recibo)
120  }
```

Si no hay clientes servidores hacen otra cosa

```
46  chan canal(texto)
47  chan pedido(int)
48  chan siguiente[3](texto)
49
50  Process Persona[id:1..N]{
51      send canal(id)
52  }
53
54  Process Coordinador{
55      while true{
56          receive pedido(idE)
57          if(empty(canal))
58              rep = "vacio"
59          else
60              receive canal(rep)
61          send siguiente[idE](rep)
62      }
63  }
64
65  Process Empleado[id:1..2]{
66      texto soli
67      while true{
68          send pedido(id)
69          receive siguiente[id](respuesta)
70          if(respuesta != "vacio")
71              atender(respuesta)
72          else
73              delay()
74      }
```

PMS

Exclusión mutua

```cpp
Process Persona[id:1..P]{
    Terminal!pedidos(id)
    Terminal?acceso()
    //usar
    Terminal!liberar()
}

Process Terminal{
    bool libre = true
    do Persona[*]?pedidos(idP) ->
        if libre
            libre = false
            Persona[idP]!acceso()
        else
            cola.push(idP)

    [] Persona[*]?liberar(); ->
        if(cola.empty)
            libre = true
        else
            Persona[cola.pop()]!acceso()
}
```

Cliente/servidor

```
91  Process Alumno[id:1..N]{
92      //Hace el examen
93      Coordinador!examenes(examen,id);
94      Profesor?notas(nota);
95  }
96  Process Coordinador{
97      cola examenes
98      int contadorExamenes = 0
99      do
100         contadorExamenes != N; Alumno[*]?examenes(examen,idA) -> examenes.push(examen,idA)
101         [] !examenes.empty; Profesor?recibir() -> Profesor!pedidos(examenes.pop())
102     od
103 }
104 Process Profesor{
105     for (int i =0; i< N; i++){
106         Coordinador!recibir();
107         Coordinador?pedidos(examen,idA);
108         int nota = corregirExamen(examen);
109         Alumno[idA]!notas(nota);
110     }
111 }
```

Barrera

```
145 Process Alumno[id:1..N]{
146     //Hace el examen
147     Barrera!llegue()
148     Barrera?activar()
149     Coordinador!examenes(examen,id);
150     Profesor[*]?notas(nota);
151 }
152 Process Barrera{
153     for(i=1, i<= N, i++)
154         Alumno[*]?llegue()
155     for(i=1, i<= N, i++)
156         Alumno[i]!activar()
157 }
```

ADA

Prioridades

```
178  Procedure Negocio is
179  TASK Caja is
180      ENTRY bajaprioridad(pedido: IN text, vuelto: OUT text, recibo: OUT text)
181      ENTRY mediaprioridad(pedido: IN text, vuelto: OUT text, recibo: OUT text)
182      ENTRY altaprioridad(pedido: IN text, vuelto: OUT text, recibo: OUT text)
183  end caja;
184  TASK BODY Caja is
185  Begin
186      loop
187          SELECT
188              when(altaprioridad.count = 0 && mediaprioridad.count = 0) -> ACCEPT bajaprioridad(pedido: IN text, vuelto: OUT text, recibo: OUT text
189                  vuelto, recibo:= procesar(pedido)
190              end bajaprioridad;
191          OR
192              when(altaprioridad.count = 0) -> ACCEPT mediaprioridad(pedido: IN text, vuelto: OUT text, recibo: OUT text)
193              do
194                  vuelto, recibo:= procesar(pedido)
195              end mediaprioridad;
196          OR
197              ACCEPT altaprioridad(pedido: IN text, vuelto: OUT text, recibo: OUT text)
198              do
199                  vuelto, recibo:= procesar(pedido)
200              end altaprioridad;
201          end select;
202      end loop;
203  end caja;
204
205  TASK TYPE Persona;
206  personas: array(1..P) of Persona
207  TASK BODY Persona is
208  prioridad: int
209  Begin
210      if(prioridad == 1)
211          Caja.altaprioridad(pedido, vuelto, recibo)
212      elif(prioridad == 2)
213          Caja.mediaprioridad(pedido, vuelto, recibo)
214      else
215          Caja.bajaprioridad(pedido, vuelto, recibo)
216      end if;
217  end persona;
218  Begin
219      null
220  end negocio;
```

Debo conocer mi id

```
Procedure sitioWeb is
TASK TYPE Servidor
servidores: array(1..5) of Servidor
TASK BODY Servidor is
    resultado: int
Begin
    loop
        coordinador.pedido(p, idC)
        resultado:= ResolverAnálisis(p)
        clientes[idC].recibirResultado(resultado)
    end loop;
end servidor;

TASK TYPE Cliente is
    ENTRY recibirResultado(resultado: IN text)
    ENTRY recibirId(id: IN int)
end cliente;
clientes: array(1..N) of Cliente
TASK BODY Cliente is
id: int
Begin
    ACCEPT recibirId(id: IN int) do
        id:= id
    end recibirId;
    loop
        adn: generaradn()
        coordinador.recibirADN(adn, id)
        ACCEPT recibirResultado(resultado: IN text)
    end loop;
end cliente;
```

```
TASK Coordinador is
    ENTRY recibirADN(adn: IN text; id: IN int)
    ENTRY pedido(p: OUT text; idC: OUT int)
end coordinador;
TASK BODY Coordinador is
cola
Begin
    loop
        SELECT
            when cola.lenght not 0 -> ACCEPT pedido(p: OUT text; idC: OUT int) do
                p, idC:= cola.pop()
            end pedido;
        OR
            ACCEPT recibirADN(adn: IN text; id: IN int) do
                cola.push(adn, id)
            end recibiradn;
        end select;
    end loop;

end coordiandor;

Begin
    FOR i in 1..N loop
        clientes[i].recibirId(i)
    end loop;
end sitioWeb;
```

Barrera

Administrador indica cuando arrancar

```
590  Procedure contador is
591  Task type Worker is
592      ENTRY comenzar()
593  end worker;
594  workers: array[1..10] of Worker
595  Task body Worker is
596      vector: array (1..100000) of Float;
597      res: Integer := 0;
598  Begin
599      ACCEPT comenzar()
600      res = sacarpromedio(vector)
601      coordinador.resultado(res)
602  end worker;
603
604  Task Coordinador is
605     ENTRY resultado(res: IN int)
606  end coordinador;
607  Task body Coordinador is
608  resultados: array (1..10) of Float;
609  Begin
610      FOR i IN 1..10 LOOP
611          workers[i].comenzar()
612      end loop;
613      FOR i IN 1..10 LOOP
614          ACCEPT resultado(res: IN int) do
615              resultados[i] = res
616          end;
617      end loop;
618
619  end coordinador;
620
621  Begin
622      null
623  end contador;
```

Se debe esperar que todos los procesos hayan llegado

```
504  Procedure juego is
505  Task type Persona is
506      ENTRY cantidadTotal(total: IN int)
507      ENTRY gruposuperior(idgrupo: IN int)
508  end persona;
509  personas: array[1..20] of Persona
510  Task body Persona is
511      int equipo = equipo()
512      int monedas = 0
513      int T
514  Begin
515      equipos[equipo].llegada()
516      equipos[equipo].salida()
517      FOR i IN 1..15 LOOP
518          monedas:= juntarmoneda()
519      end loop;
520      equipos[equipo].cantidadMonedas(monedas)
521      ACCEPT cantidadTotal(total) do
522          T = total
523      ACCEPT gruposuperior(idgrupo)
524      end;
525  end persona;
```

```
527   Task type Equipo is
528       ENTRY conseguirid(id: IN int)
529       ENTRY llegada()
530       ENTRY salida()
531       ENTRY cantidadmonedas(monedas: IN int)
532   end equipo;
533   equipos: array[1..5] od Equipo
534   Task body equipo is
535       int idE
536   Begin
537       ACCEPT conseguirid(id) do
538           idE = id
539       end;
540       FOR i IN 1..4 LOOP
541           ACCEPT llegada()
542       end loop;
543       FOR i IN 1..4 LOOP
544           ACCEPT salida()
545       end loop;
546       FOR i IN 1..4 LOOP
547           ACCEPT cantidadmonedas(monedas) do
548               total = total + monedas
549           end;
550       end loop;
551       organizador.totalEquipo(total, idE)
552   end equipo;
```

```
554  Task Organizador is
555      ENTRY totalequipo(monedas: IN int)
556  end organizador;
557  Task body organizador is
558  int maximo = 0
559  int idmaximo
560  Begin
561      FOR i IN 1..4 LOOP
562          ACCEPT totalEquipo(monedas, idE) do
563              if(monedas > maximo)
564                  maximo = monedas
565                  idmaximo = idE
566              end if;
567          end;
568      end loop;
569      FOR i IN 1..20 LOOP
570          personas[i].gruposuperior(idmaximo)
571          end;
572      end loop;
573  end organizador;
574
575  Begin
576      FOR i IN 1..4 LOOP
577          equipos[i].conseguirid(i)
578      end loop;
579  end juego;
```

Proceso intermedio

El mismo que puse para el de “debo conocer mi id”