

RESUMEN PROGRAMACIÓN CONCURRENTE – 2DO PARCIAL

ADA:

Clientes piden atención. Si pasan 10 minutos se retiran. Con RENDEVOUZ.

b) Implemente una solución donde los clientes se retiran si esperan más de 10 minutos para realizar el pago.

```
Procedure Banco is
```

```
Task Empleado is
```

```
  entry Atender (C: OUT texto);
```

```
End Empleado
```

```
Task type Cliente is
```

```
  arrClientes: array (1..C) of Cliente;
```

```
End Cliente
```

```
Task body Cliente is
```

```
  C: texto;
```

```
Begin
```

```
  SELECT
```

```
    Empleado.Atender(C: OUT texto);
```

```
  OR DELAY 600
```

```
    null
```

```
  END SELECT
```

```
End Cliente
```

```
Task body Empleado is
```

```
  loop
```

```
    accept Atender(C: OUT texto) do
```

```
      C := darComprobante();
```

```
    end Atender
```

```
  end loop
```

```
End Empleado
```

```
End Banco
```

Idem anterior. Pero pasado ese tiempo piden atención de nuevo

d) Implemente una solución donde los clientes esperan a lo sumo 10 minutos para ser atendidos. Si pasado ese lapso no fueron atendidos, entonces solicitan atención una vez más y se retiran si no son atendidos inmediatamente.

```
Procedure Banco is
```

```
Task Empleado is
```

```
|   entry Atender (C: OUT texto);
```

```
End Empleado
```

```
Task type Cliente is
```

```
|   arrClientes: array (1..C) of Cliente;
```

```
End Cliente
```

```
Task body Cliente is
```

```
|   C: texto;
```

```
Begin
```

```
  SELECT
```

```
  |   Empleado.Atender(C: OUT texto);
```

```
OR DELAY 600
```

```
  SELECT
```

```
  |   Empleado.Atender(C: OUT texto);
```

```
  ELSE
```

```
  |   null
```

```
  END SELECT
```

```
End Cliente
```

```
Task body Empleado is
```

```
  loop
```

```
  |   accept Atender(C: OUT texto) do
```

```
  |   |   C := darComprobante();
```

```
  |   end Atender
```

```
  end loop
```

```
End Empleado
```

```
End Banco
```

Procesos Worker que piden un número y luego mandan el resultado.

```
Procedure Trabajo is
```

```
Task Admin is
```

```
    equipos: array(1..5) of integer;  
    entry Resultado (suma: OUT integer);  
    entry empezar ();
```

```
End Admin
```

```
Task Type Worker is
```

```
    arrWorker: array (1..10) of Worker;  
    entry recibirID (id: OUT integer);
```

```
End Worker
```

```
Task Type Equipo is
```

```
End Equipo
```

```
Task body Admin is
```

```
    total: integer; i: integer;
```

```
Begin
```

```
    SELECT
```

```
    | accept empezar();
```

```
    OR
```

```
    | accept Resultado (suma: OUT integer) do
```

```
        total += suma;
```

```
    End Resultado
```

```
    prom := total/1000000;
```

```
End Admin
```

```
Task body Worker is
```

```
    suma: integer; VC: array (1..100000) of integer;
```

```
Begin
```

```
    Admin.empezar();
```

```
    for i := 1..100000 do
```

```
        suma += VC[i];
```

```
    end for
```

```
    Admin.Resultado(suma);
```

```
End Worker
```

Enviar algo para buscar y a la vez recibir en loop. Con ENTRY Terminar para avisar cuando hay una nueva opción para buscar.

```
Procedure Huellas is

Task Especialista is
|   entry Resultado (cod: OUT integer; valor: OUT integer);
End Especialista

Task Type Servidor is
|   arrServidor: array (1..8) of Servidor;
|   entry recibirHuella (h: OUT TEST);
End Servidor

Task body Especialista is
|   h: TEST; i,max,codMax,cod,valor: integer;
Begin
|   max := -1; codMax := 0;
|   loop
|       tomarImagen(h);
|       for i := 1..16 do
|           SELECT
|               |   accept recibirHuella (h: OUT TEST) do
|                   |   test := h;
|                   |   end recibirHuella
|               OR
|               |   accept Resultado (cod: OUT integer; valor: OUT integer) do
|                   |   if (valor > max) do
|                       |   max := valor;
|                       |   codMax := cod;
|                   |   end if
|               |   end Resultado
|           end for
|           for i := 1..8 do
|               |   accept terminar();
|           end for
|       end loop
End Especialista
```

```

Task body Servidor is
|   test: TEST; c,v: integer;
Begin
|   loop
|       Especialista.recibirHuella(test);
|       Buscar (h,v,c)
|       Especialista.Resultado(c,v);
|       Especialista.terminar();
|   end loop
End Servidor

End Huellas

```

Pedir solicitud y esperar cierto tiempo a una cantidad de servidores

```

Task body Tarea is
|   i: integer;
Begin
|   loop
|       for i := 1..20
|           SELECT
|               Admin.Solicitar(i);
|           OR DELAY 5
|               write "Información del banco", i, "se encuentra vacía"
|           END SELECT
|       end for
|   end loop
End Tarea

```

Solicitudes con Prioridad. Se necesitan dos ENTRY.

Parcial 2022)

Procedure Banco is

Task Empleado is

```
    entry pedidoRegular (pago: IN texto; comprobante: OUT texto);  
    entry pedidoPremiun (pago: IN texto; comprobante: OUT texto);  
end Empleado
```

Task Type Premiun is

```
    arrPremiun: array (1..P) of Premiun;  
End Premiun
```

Task Type Regular is

```
    arrRegular: array (1..R) of Regular;  
End Regular
```

Task body Empleado is

```
    i: integer;
```

Begin

```
    loop  
        SELECT  
            accept pedidoPremiun (pago: IN texto; comprobante: OUT texto) do  
                comprobante := generarComprobante (pago);  
            end pedidoPremiun  
        OR  
            (when (pedidoPremiun'COUNT = 0)) accept pedidoRegular (pago: IN texto; comprobante: OUT texto) do  
                comprobante := generarComprobante (pago);  
            end pedidoRegular  
    end loop  
End Empleado
```

Task body Premiun is

```
    pago: texto;  
    c: texto;
```

Begin

```
    Empleado.pedidoPremiun (pago,c);  
End Premiun
```

Task body Regular is

```
    pago: texto;  
    c: texto;
```

Begin

```
    SELECT  
        Empleado.pedidoRegular (pago,c)  
    OR DELAY 1800  
        null  
End Regular  
end Banco
```