




Posibles soluciones a los ejercicios del parcial

Estas son posibles soluciones, no significa que sean la única forma, traté de hacerlas simples y siguiendo las ideas que hemos visto en las teorías y/o explicaciones prácticas para no confundir.



Resolver con **SEMÁFOROS** el siguiente problema. En una empresa hay UN Coordinador y 30 Empleados que formarán 3 grupos de 10 empleados cada uno. Cada grupo trabaja en una sección diferente y debe realizar 345 unidades de un producto. Cada empleado al llegar se dirige al coordinador para que le indique el número de grupo al que pertenece y una vez que conoce este dato comienza a trabajar hasta que se han terminado de hacer las 345 unidades correspondientes al grupo (cada unidad es hecha por un único empleado). Al terminar de hacer las 345 unidades los 10 empleados del grupo se deben juntar para retirarse todos juntos. El coordinador debe atender a los empleados de acuerdo al orden de llegada para darle el número de grupo (a los 10 primeros que lleguen se le asigna el grupo 1, a los 10 del medio el 2, y a los 10 últimos el 3). Cuando todos los grupos terminaron de trabajar el coordinador debe informar (imprimir en pantalla) el empleado que más unidades ha realizado (si hubiese más de uno con la misma cantidad máxima debe informarlos a todos ellos). **Nota:** maximizar la concurrencia; suponga que existe una función Generar que simula la elaboración de una unidad de un producto.

```

sem mutexCola = 1;
sem hayEmpleado = 0;
sem esperaGrupo[30] = ([30] 0);
sem finalGrupo = 0;
sem mutexGrupo[3] = ([3] 1);
sem esperaSalir[3] = ([3] 0);

queue cola;
int grupo[30];
int cantidades[30] = ([30] 0);
int cantProd[3] = ([3] 0);
int cantFin[3] = ([3] 0);

```

```

Process Coordinador{
    int i, idE, max, numG = 0, cant = 0;

    for (i=0; i<30; i++) {
        P(hayEmpleado);
        P(mutexCola);
        pop(cola, idE);
        V(mutexCola);
        cant++;
        if (cant>10) {
            numG++;
            cant = 1;
        };
        grupo[idE] = numG;
        V(esperaGrupo[idE]);
    };
    for (i=0; i< 3; i++) P(finalGrupo);
    max = CalculaMaximo(cantidades);
    for (i=0; i< 30; i++)
        if (cantidades[i] == max) print(i);
};


```

```

Process Empleado[id: 0..29] {
    int miG;

    P(mutexCola);
    push(cola, id);
    V(mutexCola);
    P(hayEmpleado);
    P(esperaGrupo[id]);
    miG = grupo[id];
    P(mutexGrupo[miG]);
    while (cantProd[miG] < 345) {
        cantProd[miG]++;
        V(mutexGrupo[miG]);
        GenerarProducto
        cantidades[id]++;
        P(mutexGrupo[miG]);
    };
    cantFin[miG]++;
    if (cantFin[miG] == 10) {
        for (i=0; i<9; i++) V(esperaSalir[miG]);
        V(finalGrupo);
    }
    else {
        V(mutexGrupo[miG]);
        P(esperaSalir[miG]);
    };
};

```



Resolver con **MONITORES** la siguiente situación. En la guardia de traumatología de un hospital trabajan 5 médicos y una enfermera. A la guardia acuden P Pacientes que al llegar se dirigen a la enfermera para que le indique a que médico se debe dirigir y cuál es su gravedad (entero entre 1 y 10); cuando tiene estos datos se dirige al médico correspondiente y espera hasta que lo termine de atender para retirarse. Cada médico atiende a sus pacientes en orden de acuerdo a la gravedad de cada uno. **Nota:** maximizar la concurrencia.

```

Process Paciente[id:0..P-1] {
    int gravedad, numMed;
    Enfermera.Solicitud(gravedad, numMed);
    SalaEspera[numMed].EsperarTurno(id, gravedad);
    Consultorio[numMed].ConsultaPaciente;
};

```

```

Process Medico[id: 0..4] {
    int idP;

    while (true) {
        SalaEspera[id].ProximoPaciente(idP);
        Consultorio[id].Atender;
    };
};

```

Monitor Enfermera {

```

    Procedure Solicitud (int *grav, int *idMedico) {
        idMedico = randon (0..4);
        grav = determinaGravedad;
    }
}

```

Monitor SalaEspera[id:0..4] {

```

    queue cola;
    cond esperaMedico, esperaPaciente[P];

```

```

    Procedure EsperarTurno (int idP, int g) {
        insertar_ordenado (cola, (idP, g));
        signal (esperaMedico);
        wait (esperaPaciente[idP]);
    }

```

```

    Procedure ProximoPaciente (int *pac) {
        int g;

        if (empty(cola)) wait (esperaMedico);
        pop (cola, (pac, g));
        signal (esperaPaciente[pac]);
    }

```

Monitor Consultorio[id: 0..4] {

```

    bool llegoPaciente;
    cond esperaMedico, esperaFin

```

```

    Procedure ConsultaPaciente {
        llegoPaciente = true;
        signal (esperaMedico);
        wait (esperaFin);
    }

```

```

    Procedure Atender {
        if (not llegoPaciente)
            wait (esperaMedico);
        // atiende al paciente
        signal (esperaFin);
        llegoPaciente = false;
    }
}

```