Practica de repaso 3

1. Resolver el siguiente problema. En una elección estudiantil, se utiliza una máquina para voto
   electrónico. Existen N Personas que votan y una Autoridad de Mesa que les da acceso a la máquina de acuerdo con el orden de llegada, aunque ancianos y embarazadas tienen prioridad sobre el resto.
   La máquina de voto sólo puede ser usada por una persona a la vez. Nota: la función Votar() permite usar la máquina.

```
Monitor Mesa {
    cond cola[N],termine, llegue;
    cola filaOrdenada;
    int dormidos= 0;
    boolean libre= true;

    procedure llegar(id: int in, edad: int in, embarazada: bo
        dormidos+=1;
        filaOrdenada.insertar(id, edad, embarazada);
        signal(llegue);
        wait(cola[id]);

    }

    procedure darAcceso(){
        if (dormidos == 0 ){
             wait(llegue);
        }
        signal(cola[filaOrdenada.pop(id)]);
      wait (termine);
        dormidos-=1;
  }

  procedure salir(){
```

```
      signal(termine);
    }

}

process Persona [id: 0.. N-1]{
    Mesa.llegar(id, edad, embarazada);
    Votar();
    Mesa.salir();
}

process Autoridad(){
    for i:= 1 to N
        Tarea.darAcceso();
}
```

2. Resolver el siguiente problema. En una empresa trabajan 20 vendedores ambulantes que forman 5 equipos de 4 personas cada uno (cada vendedor conoce previamente a qué equipo pertenece). Cada equipo se encarga de vender un producto diferente. Las personas de un equipo se deben juntar antes de comenzar a trabajar. Luego cada integrante del equipo trabaja independientemente del resto vendiendo ejemplares del producto correspondiente. Al terminar cada integrante del grupo debe conocer la cantidad de ejemplares vendidos por el grupo. Nota: maximizar la concurrencia

```
Monitor Equipo[id: 0.. 4]{
    int llegaron=0;
    cond cola, esperar;
    int fin=0;
    int cantidadVendida=0;

    procedure llegar(){
        llegaron+=1;
        if (llegaron == 4 )
            signal_all(cola);
        else
            wait (cola);
```

```
    }

    procedure terminar(vendi: int in; vendidos: int out){
        fin+=1;
        cantidadVendida+=vendi
        if (fin == 4 ){
            vendidos:= cantidadVendida;
            signal_all(esperar);
        }
        else
            wait(esperar);
    }
}

process Trabajador (id: 0.. 49){
    int cantidadVendidaEquipo;
    int miCantidad
    int idEquipo;
    Equipo[idEquipo].llegar();
    miCantidad= vender();
    Equipo[idEquipo].terminar(miCantidad, cantidadVendidaEqui
```

3. Resolver el siguiente problema. En una montaña hay 30 escaladores que en una parte de la subida deben utilizar un único paso de a uno a la vez y de acuerdo con el orden de llegada al mismo. Nota: sólo se pueden utilizar procesos que representen a los escaladores; cada escalador usa sólo una vez el paso.

```
Monitor Paso{
    cond espera;
    int dormidos=0;
    boolean libre=true;

    process darPaso(){
        if (!libre){
            dormidos+=1;
            wait(espera);
```

```
        }
        else
            libre:=false;
    }

    procedure terminarPaso(){
        if (dormidos > 0 ){
            dormidos -=1;
            signal(espera);
        }
        else
            libre:= true;
    }
}

process Escalador [id: 0.. 29]{
    Paso.darPaso();
    //da el paso
    Paso.terminarPaso();
}
```